Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN
INFORMATICA
Ciclo XXVIII

Settore concorsuale di afferenza: 09/H1
Settore scientifico disciplinare: ING-INF/05

DATA AND TEXT MINING TECHNIQUES FOR IN-DOMAIN AND CROSS-DOMAIN APPLICATIONS

Presentata da: Giacomo Domeniconi

Coordinatore Dottorato	Relatore
Prof. Paolo Ciaccia	Prof. Gianluca Moro
	Tutor
	Prof. Claudio Sartori

Esame finale anno 2016

Abstract

In the big data era, a wide amount of data has been generated in different domains, from social media to news feeds, from health care to genomic functionalities. When addressing a problem, we usually need to harness multiple disparate datasets (Zheng, 2015). Humans can intelligently face any problem and domain, even applying previously learned knowledge to solve new problems faster or with better solutions. For example, we may find that having knowledge of C++ can help studying Java or other programming languages and playing piano can help learning the guitar. Transfer learning aims at transferring knowledge from some source domains to a target domain (Hu et al., 2011). A branch of transfer learning is the cross-domain classification, where knowledge must be transferred across two domains which are generally different in spite of having the same labels for data. In general, data from different domains may follow different modalities, each of which has a different representation, distribution, scale and density. For example, text is usually represented as discrete sparse word count vectors, whereas an image is represented by pixel intensities or outputs of feature extractors which are real-valued and dense.

Nowadays plenty of Data Mining and Machine Learning techniques are proposed in literature, which have already achieved significant success in many knowledge engineering areas, including classification, regression and clustering. Anyway some challenging issues remain when tackling a new problem: how to represent the problem? What approach is better to use among the huge quantity of possibilities? What is the information to be used in the Machine Learning task and how to represent it? There exist any different domains from which borrow knowledge?

This dissertation proposes some possible representation approaches for problems in different domains, from text mining to genomic analysis. In particular, one of the major contributions is a different way to represent a classical classification problem: instead of using an instance related to each object (a document, or a gene, or a social post, etc.) to be classified, it is proposed to use a pair of objects or a pair object-class, using the relationship between them as label. The application of this approach is tested on both flat and hierarchical text categorization datasets, where it potentially allows the efficient addition of new categories during classification. Furthermore, the same idea is used to extract conversational threads from an unregulated pool of messages and also to classify the biomedical literature based on the genomic features treated.

Several other contributions are here proposed. For instance a method for cross domain text classification based on nearest centroid classification, where profiles of categories are generated from the known domain and then iteratively adapted to the unknown one. Despite being conceptually simple and having easily tuned parameters, this method achieves the state-of-the-art accuracy in most benchmark datasets with fast running times. Another proposed method involves using Markov Chains to represent and handle the in and cross domain sentiment classification.

Finally, a last contribution is to reliably predict new functionalities for the genes or proteins of an organism. The idea is to leverage a new representation of the annotation discovery problem and a random perturbation of the available controlled annotations, to allow the application of supervised algorithms to predict with good accuracy unknown gene or protein annotations. This approach is tested both using the same organism as training and test, and in a cross domain task, using a well-studied organism to transfer knowledge in order to predict new gene annotations of a target, less studied, organism.

To everyone who somehow supported me

Contents

Chapter 1

Introduction

Nowadays, across many contexts, information of all kinds is produced at fast rates. However, the extraction of useful knowledge from very large amounts of data is not always trivial, especially to be performed manually. Data mining research is dedicated to developing processes for automated extraction of useful, high-level information hidden within large amounts of data; it has many applications in business, finance, science, society and so on. Machine learning is the general base of data mining, it provides algorithms to analyze a set of raw data and extract from it a knowledge model which encapsulates recurring patterns within it and allows to make predictions on future data.

A plenty of machine learning algorithms were proposed in literature in the years, reaching high performances in terms of efficacy and efficiency. Although the research of new more performing algorithms is always active, nowadays most of the practical difficulties to resolve a data mining problem consist in the correct representation of it. Generally, a machine learning algorithm works with data expressed in a matrix, where rows refer to records (or objects, or instances, or observations, etc.) and columns refer to characteristics that describe each record. One of the main issues when facing a data mining problem, is to define an effective representation of it through a matrix, allowing the use of machine learning algorithms.

A classic example of non-trivial problem representation is the automatic analysis of unstructured textual data, which is easily interpreted by humans in small amounts, but not directly understandable by computers. Such text data can contain valuable information, but automatic analysis is necessary to efficiently extract such knowledge in an usable form. Text mining is a branch of data mining studying techniques for automatic treatment of free text. This is generally achieved by processing text to represent it with structured forms, which can then be analyzed by usual learning algorithms. A very common form to efficiently represent every document of a large collection is the *bag of words*, a multiset of relevant words or other related features contained in the text. In this context, a quite common task is text categorization, consisting in the automatic organization of a possibly very large set of documents into distinct, meaningful categories. Categorization is usually carried out according to separate different topics discussed in document, but the general description involves also tasks like spam filtering in e-mail and identification of positive and negative reviews of a product. Organization of documents by topics can be useful in contexts like dividing books in sections, separating news articles by categories and organizing Web pages in hierarchical directories.

As already discussed, most data mining methods are based on machine learning: given a dataset a knowledge model is inferred from them, which is able to handle further data within the same domain. This approach automates the learning process, but requires a consistent set of data, which may not be priorly available and thus requires considerable effort to be built. To overcome this problem, at least in some circumstances, *transfer learning* methods were proposed in literature. The goal is to transferring knowledge from known source domains to target domains (Hu et al., 2011), thus inferring the knowledge models from a known source dataset and use that knowledge to handle unknown data coming from different domains.

Cross-domain classification is the branch of transfer learning where knowledge must be transferred across two domains, which are generally different in spite of having the same labels for data. In this task a knowledge model to classify a set of instances within a target domain is obtained by leveraging a set of pre-labeled instances of a source domain which is allowed to have some differences from the former. Most methods to perform this task are based on advanced statistical techniques, which transform source data to adapt them to the target or convert data of both domains to a common representation.

Here are proposed several approaches for problems in different domains, applying the methods in both in-domain and cross-domain applications. A recurring idea used in different proposed approaches, is to represent a machine learning problem in a different way: instead of using an instance related to each object (a document, or a gene, or a social post, etc.) to be classified, where each feature is related to a certain characteristic of it, it is proposed to use a pair of objects or a pair object-class as instance (row) of the model, using features (columns) that express relations - in different points of views - between the paired objects. The application of this approach is tested on both flat and hierarchical text categorization datasets, where it potentially allows the efficient addition of new categories during classification. Furthermore, the same idea is used to extract conversational threads from an unregulated pool of messages and also to classify the biomedical literature based on the genomic features treated. In the latter, a combination of document-centric and topic-centric strategies are used to annotate (i.e. classify) biomedical literature.

Another proposal of this dissertation is a simple method for cross-domain text categorization based on explicit document-like representations of categories, likely to some existing methods for standard classification, which are created from the source domain and then iteratively refined to adapt them to the target domain. From experimental evaluations, this approach appears to be effective in finding good representations of categories of target domain and consequently in accurately classifying documents within them.

A specific application of text categorization is the classification of textual opinions in positive, negative or neutral polarity, generally called sentiment classification. This task is useful to understand people thoughts about products, services, persons, organisations, and so on. Interpreting and labelling opportunely text data polarity is a costly activity if performed by human experts. To cut this labelling cost, cross domain approaches can be helpful, inferring models from a known source dataset and using them to classify unknown target set. Here is proposed a Markov chain-based method to accomplish sentiment classification in both in-domain and cross-domain tasks. The proposed technique is based on the Markov chain theory, using it both to represent the problem and to directly classify the opinions, allowing a knowledge flow from source to target documents.

Finally, a last contribution is to reliably predict new functionalities for the genes or proteins of an organism. The knowledge of gene and protein functions is key to understand the complex biological processes; particularly when such knowledge is expressed in a controlled and computable way through the use of terminologies and ontologies, i.e. through controlled biomedical annotations, it can be profitably leveraged by computational algorithms for new biomedical information and knowledge discovery. Here is proposed a new learning method to automatically infer novel unknown gene functional annotations according to the available annotations; it is based on a novel representation of the annotation discovery problem and a random perturbation of the available annotations that permit to use supervised algorithms despite the unsupervised nature of the task. The method is also applied in a cross-domain view, to reliably predict new functionalities for the genes of an organism based on the controlled annotations available for the genes of another organism; in this way, the available biological knowledge about a better studied organism can be leveraged to improve the knowledge about a more limitedly studied one.

1.1 Contributions

Here are summarized the main contributions presented in this dissertation. In the sub-points, references to related publications and works currently under review are given.

- Firstly a general data mining and machine learning introduction is done, introducing the concept of transfer learning and in-domain and cross-domain applications.

- An high level discussion of text mining tasks and issues is given, focusing on representation techniques that allow the application of automated algorithms to unstructured textual data. Starting from the Vector Space Model, to the feature selection task and the term weighting. Regarding the latter, a novel term weighting method is presented: based on computing the well-known *tf.idf*, without considering documents of the category to be recognized, so that importance of terms frequently appearing only within it is not underestimated. A further proposed variant is additionally based on *relevance frequency*, considering occurrences of words within the category itself. In extensive experiments on two recurring text collections with several unsupervised and supervised weighting schemes, is shown that the proposed ones generally perform better than or comparably to other ones in terms of accuracy.

  - G. Domeniconi, G. Moro, R. Pasolini, C. Sartori. A Study on Term Weighting for Text Categorization: a Novel Supervised Variant of tf.idf. *4th International Conference on Data Management Technologies and Applications (DATA)*, 2015 (Domeniconi et al., 2015c).
  - G. Domeniconi, G. Moro, R. Pasolini, C. Sartori. A Comparison of Term Weighting Schemes for Text Classification and Sentiment Analysis with a Supervised Variant of tf.idf. Submitted to *Data Management Technologies and Applications* (*Communications in Computer and Information Science* series, Springer).

- An overview of text categorization state of the art, on both In-Domain and Cross-domain applications, is done. A novel cross-domain text categorization method based on nearest centroid classification is presented: the source domain is used to extract initial category profiles, which are then iteratively refined according to most similar documents in the target domain, until a stable configuration is reached. Compared to the state of the art, the method yields better or comparable results with a more conceptually simple algorithm, which can be implemented easily, exposes few parameters to be tuned and runs fast. A couple of variants are separately discussed to further improve robustness with respect to parameters and running times.
  - G. Domeniconi, G. Moro, R. Pasolini, C. Sartori. Cross-domain text categorization through iterative refining of target categories representations. *6th International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, 2014, (awarded as Best Student Paper) (Domeniconi et al., 2014b).
  - G. Domeniconi, G. Moro, R. Pasolini, C. Sartori. Iterative refining of category profiles for nearest centroid cross-domain text classification. *Knowledge Discovery, Knowledge Engineering and Knowledge Management* (*Communications in Computer and Information Science* series, Springer), 2015. (Domeniconi et al., 2015d).

- The introduction af a novel idea for a domain-independent text categorization model, that predicts whether and how documents and category profiles are related from the analysis of semantic relationships held between respective representative words. A couple of non-trivial algorithms are proposed to search indirect semantic relationships from the primitive links given by the WordNet database. Specific methods are proposed for both flat and hierarchical classification, along with results of experiments to evaluate the classification accuracy and the generality of the model.

  - G. Domeniconi, G. Moro, R. Pasolini, C. Sartori. Domain-independent text categorization. *2nd Italian Workshop on Machine Learning and Data Mining (MLDM.it)* at the *XIII Conference of the Italian Association for Artificial Intelligence*, 2013.
  - (undergoing review process) G. Domeniconi, G. Moro, R. Pasolini, C. Sartori. Towards topic-independent text classification: a novel semantic learning method for hierarchical corpora. Submitted to *Machine Learning*.

- A specific application of text categorization is the sentiment analysis.. Here is proposed a new method based on the Markov chain theory to accomplish both transfer learning and sentiment classification tasks. Experiments on popular text sets show that the proposed approach achieves performance comparable with other works but with a lower parameter calibration effort.

  - G. Domeniconi, G. Moro, A. Pagliarani, R. Pasolini. Markov Chain Based Method for In-Domain and Cross-Domain Sentiment Classification. *7th International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, 2015 (Domeniconi et al., 2015b).

- After an extensive overview of conversation threads detection techniques, a novel general approach to detect topic threads from any type of conversational text data is proposed. To determine the relevant features of messages each message is mapped into a three dimensional representation based on its semantic content, the social interactions in terms of sender/recipients and its issue time. Then a clustering algorithm is used to detect conversation threads. In addition, is proposed a supervised approach that builds a classification model which combines the above extracted features for predicting whether a pair of messages belongs to the same thread or not. Extensive tests on seven datasets demonstrate the effectiveness of the proposed method. This work is done in cooperation with the IBM Dublin Research Labs.

  – (undergoing review process) G. Domeniconi, K. Semertzidis, V. Lopez, E. M. Daly, S. Kotoulas, G. Moro. Combining Clustering and Classification for Thread Detection in Conversational Messages. *25rd International World Wide Web Conference (WWW)*, 2016.

- Job recommendation systems have been proposed in literature in order to automate and simplify the job seeking process. Here, a novel approach is proposed to find out relationships, through the use of the *Latent Semantic Analysis* (LSA), between jobs and people skills making use of data from LinkedIn users' public profiles. A hierarchical clustering of job positions is carried out and it is used to create a job recommendation system. The outcome of experimental test on $\sim 50,000$ real LinkedIn profiles proves the effectiveness of the method in recommending job positions. Noteworthy, both the hierarchical clustering and the recommendation system do not require parameters to be tuned.

  – G. Domeniconi, G. Moro, A. Pagliarani, R. Pasolini. Job Recommendation From Semantic Similarity of LinkedIn Users' Skills. *5th International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 2016.

- After an extensive overview of computational techniques able to reliably predict new gene or protein functionalities, a novel representation of the annotation discovery problem is proposed to enable applying supervised algorithms to predict Gene Ontology annotations of different organism genes. In order to use supervised algorithms despite labeled data to train the prediction model are not available, the proposal is to use a random perturbation method of the training set, which creates a new annotation matrix to be used to train the model to recognize new annotations. This approach is extensively tested on both In-Domain and Cross-Domain applications, results show both the usefulness of the perturbation method and a great improvement of the cross-organism models with respect to the single-organism (i.e.

In-Domain) ones, without influence of the evolutionary distance between the considered organisms.

- G. Domeniconi, M. Masseroli, G. Moro, P. Pinoli. Discovering New Gene Functionalities from Random Perturbations of Known Gene Ontological Annotations. *6th International Conference on Knowledge Discovery and Information Retrieval (KDIR)*, 2014, (Domeniconi et al., 2014a).
- G. Domeniconi, M. Masseroli, G. Moro, P. Pinoli. Random Perturbations of Term Weighted Gene Ontology Annotations for Discovering Gene Unknown Functionalities. *Knowledge Discovery, Knowledge Engineering and Knowledge Management* (*Communications in Computer and Information Science* series, Springer), 2015. (Domeniconi et al., 2015a).
- G. Domeniconi, M. Masseroli, G. Moro, P. Pinoli. Cross-organism learning method to discover new gene functionalities. *Computer Methods and Programs in Biomedicine.*

• Gene function curation is largely based on manual assignment of Gene Ontology (GO) annotations to genes by using published literature. The annotation task is extremely time-consuming, therefore there is an increasing interest in automated tools that can assist human experts. Here is introduced GOTA[1], a GO term annotator for biomedical literature. The proposed approach makes use only of information that is readily available from public repositories and it is easily expandable to handle novel sources of information. The classification capabilities of GOTA is extensively assessed on a large benchmark set of publications. The overall performances are encouraging in comparison to the state of the art in multi-label classification over large taxonomies. Furthermore, the experimental tests provide some interesting insights into the potential improvement of automated annotation tools.

- P. Di Lena, G. Domeniconi, L. Margara, G. Moro. GOTA: GO term annotation of biomedical literature. *BMC Bioinformatics*, 2015 (Lena et al., 2015).

[1]`http://gota.apice.unibo.it`

1.2 Structure of the Thesis

Chapter 2 gives a general introduction of automated data analysis along with its motivations, summarizing the data mining and machine learning high level task, finally introducing the concept of transfer learning. In Chapter 3, the most recurring techniques employed in text mining are described in detail, along with novel contributions presented in this dissertation for each of them. Chapter 4 focuses on recommender systems, particularly on the Job Recommendation, providing a new approach based on social network information about the users. Chapter 5 treats the application of data mining on genomics. It firstly focuses on the prediction of the biomolecular annotations of genes, proposing a new method based on random perturbations of the knowledge, on both In-Domain and Cross-Domain tasks. Then, the focus of the chapter moves to the automatic association of genomic annotations to the scientific literature. Finally, Chapter 6 briefly sums up the work and indicates potential directions for further research.

1.3 Technical Remarks

All the experimental evaluations which are part the work presented in the thesis have been performed on virtualized hardware, with the use of up to 8 parallel processing cores.

Tests have been run within a dedicated software framework implemented in Java. The following third-party tools and libraries have been extensively used:

- the *WEKA* software suite for machine learning (Hall et al., 2009) (`http://www.cs.waikato.ac.nz/ml/weka/`),
- the *Java WordNet Interface* (JWI) (Finlayson, 2014) (`http://projects.csail.mit.edu/jwi/`).

Where cited, version 3.1 of the WordNet database has been used.

Chapter 2

Introduction to Data Mining

This chapter gives a general overview about data mining techniques, including the motivations for this research, the general tasks which are usually tackled and an Introduction to transfer learning problems.

2.1 Data Mining

The analysis of *data* is a key part of many types of task in many context. Often, however, there exists a *gap* between data and valuable information, generally meaning that possessed data is not enough to have a sufficiently accurate knowledge of the phenomenon under analysis. Traditional approaches for deriving knowledge from data rely strongly on manual analysis and interpretation. For any domain (scientific, business, marketing, finance, health, etc.) the success of a traditional analysis depends on the capabilities of one/more specialists to read into the data: scientists go through remote images of planets and asteroids to mark interest objects, such as impact craters; bank analysts go through credit applications to determine which are prone to end in defaults. Such an approach is slow, expensive and with limited results, relying strongly on experience, state of mind and specialist know-how. Furthermore, with the continuous development of processing, storage and communication means, a situation of *excess* of information can be easily reached. While a limited quantity of information can be feasibly analyzed by human workers, difficulties can be met in extracting useful information from an overly abundant quantity of data.

At this end, the *data mining* and *knowledge discovery* are generally focused on the development of methods and techniques for the non-trivial extraction of high-level information from potentially enormous quantities of *raw* data (Frawley et al., 1992). A definition of data mining (DM) is given by Fayyad et al. (1996): "*Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*".

2.2 Machine Learning

Data mining or knowledge discovery process generally involves different steps, ranging from the collection and processing of data to be analyzed to the consumption of final results. From these steps, the key one is the conversion of the great and not easily understandable amount of "raw" data into compact, high-level information which can be easily interpreted.

This task is usually based on *machine learning* techniques, which are generally used to infer *knowledge models* from data of heterogeneous form and nature: these models can highlight interesting features of the analyzed data and can be used to make predictions regarding data which arrives in the future.

In general, input data is composed of *records* or *instances*, each corresponding to one item, or event, of a set to be analyzed: these can be for example transactions (purchases) of a store. Observations are distinguishable from each other by a set of *features*, indicating which are the properties of each record. This can be viewed as a table of a relational database, or also a $n \times m$ matrix, in which records correspond to rows, while features correspond to columns.

To be used as input for machine learning algorithms, available data generally needs to undergo some pre-processing steps to obtain a set of observations with convenient features. This can involve joining data from multiple tables of a relational database, in order to obtain in practice a single table where all information is condensed with no or minimal loss. This can lead to a consistent number of features, which can make the analysis cumbersome (*dimensionality problem*): specific methods exist to perform selection of these features and discard those least useful in the analysis. Learning methods are generally able to work only with specific types of data: some of them in their basic versions can only deal with discrete values, such as integer numbers or categorical data (elements from a finite set), while others are specialized on continuous ranges of values, which can anyway be discretized in order to be treated by other algorithms.

Different machine learning algorithms exist, differing mainly for the type of knowledge which is extracted. A rather common need is to extract knowledge from known items or past events in order to somehow assign labels to them or to new items or events according to some specific important characteristics of them. This generic description can be applied to different tasks.

- ***Classification*** is the task of labeling new items with a set of predefined set of *classes*, which represent categories in which is convenient to subdivide such items. In general, this process involves building a *training* set of observation which are already correctly labeled with the same classes; a machine learning algorithm then analyzes data to find relevant correlations between features and class labels and extracts from them a *classifier*, which is ideally able to correctly label any observation represented with the same features. As data provided for training must be priorly labeled, algorithms of this type are said to be *supervised*. An example of applications of this general task is insurance fraud detection (indemnity requests are classified as either suspect or not).

- ***Clustering*** is the task of partitioning data in some groups, said *clusters*, about which no prior knowledge is given. The goal of a clustering algorithm is, given a set of data, to identify clusters within this set so that each contains observations which are very similar to each other and significantly different from those of other clusters; possibly, also new observations can then be organized in the same clusters. Differently from classification, clustering algorithms are *unsupervised*, as they analyze data with no priorly assigned group labels. Among concrete applications described above, customers segmentation is an example of clustering problem.

- ***Recommendation*** is the task of provide novel useful information, called *recommendations*, related to an object (generally an user). Recommender systems study patterns of behavior to know what someone will prefer from among a collection of things he has never experienced.

2.3 Transfer Learning

Broadly speaking, the goal of a transfer algorithm is to use supervised training data from related domain (*source*) to improve performance in *target* data. This is usually achieved by training classifiers for related tasks jointly. What is meant by *joint training* depends on the transfer learning algorithms; each of them develops a particular notion of relatedness and the corresponding transfer algorithms designed to exploit it.

Three are the main lines of research in transfer learning:

- **learning hidden representations**: the main assumption is that there exists a mapping from the original input space to an underlying shared feature representation. This latent representation captures the information necessary for training classifiers for the target domain. The aim of a transfer model is to simultaneously uncover the underlying shared representation and the parameters of the domain-specific classifiers.
- **feature sharing**: the main assumption is that source and target domains share relevant features. The goal of a transfer model is to simultaneously discover the subset of relevant features and the parameter values for the domain-specific classifiers. More specifically, there exist two types of feature sharing approaches: parameter tying and joint sparsity approaches.
- **hierarchical bayesian learning**: domains are assumed to be related by means of sharing a common prior distribution over classifierss parameters. This shared prior distribution can be learned in a classical hierarchical bayesian using data from the source domain.

2.3.1 In-Domain and Cross-Domain Classification

Typical approaches to automatic classification in predefined categories require the availability of a training set, from which necessary knowledge is inferred. A training set generally must have some characteristics in order to be usable and to yield optimal results: it must contain an adequate amount of data, which must be as exactly as possible representative of those which will be classified and must be coherently labeled with the same categories which are going to be assigned to new instances. In other words, can be said that training instances must be within the same *domain* of the target ones. This task is also called *in-domain categorization*.

Retrieving a set of labeled instances of the exact same domain of the target set can often be difficult. However, in some cases, a set of labeled data of a slightly different domain may be available. The difference between the domains may consists in the use of some different features or in the organization within categories representing slightly different concepts. Considering the general setting of the problem, the in-domain categorization techniques might be applied to infer a classifier from available training data and apply it to the target set.

However, due to the possible differences between the two domains, this would likely not result in an accurate classification, as many of the features known by the classifier would not be found within the target documents. Cases like these would require specific methods to somehow transfer the knowledge extracted from the available training data to the target domain. *Cross-domain categorization* is a branch of transfer learning that refers to the specific task of classifying a set of target instances in predefined categories, using as support a set of pre-labeled data of a slightly different domain.

Contrarily to in-domain categorization problems, where target instances are generally assumed to be not known in advance, typical cross-domain methods imply that unlabeled target set must be given in advance, at least in part, as some knowledge of the target domain is necessary. Also, the majority of cross-domain methods consider a single-label setting (i.e. one and only one category label to each instance), which is assumed by default for the rest of this dissertation.

Chapter 3

Text Mining

Data mining has been introduced as the extraction of information from large amounts of data. Such data can come in different forms, each requiring specific techniques to be handled. Machine learning algorithms, as discussed in the previous chapter, are suitable to handle *structured* data, where single pieces of information are organized according to some model and each of them is easily distinguishable from the others, as usually happens for example in a relational database. Textual data, written in natural language, can't be handled directly by these algorithms. *Text mining* (or *text analysis*) is the research field which deals with extracting useful, high-level information from text: it can be seen in practice as data mining applied on natural language text. Most text mining-related literature has its roots in data mining, machine learning and related fields as well as in *information retrieval*, which generally refers to automatic retrieval from a vast data collection of information relevant to some need. Most research on this field is focused on retrieval from text collections and proposed many of the techniques used in text mining.

Likely to data mining, a text mining process generally involves a preliminary pre-processing phase, which in this case is needed to translate free text into a structured form, allowing the use of statistical analysis and machine learning algorithms. The typical choice is to represent documents as vectors (also called *bag-of-words*), according to a variety of techniques spawned from information retrieval research, which can then be processed with many known methods.

3.1 Brief History

Research on automatic analysis of text dates back to several decades ago: its need was generally driven by the general necessity to efficiently retrieve specific knowledge from consistent amounts of information. Storage of information, primarily in written forms, is performed since hundreds and even thousands of years. As the amount of information grows, retrieving specific needed pieces of it naturally becomes more difficult, so any method to improve the efficiency of search across information sources has primary importance. Since shortly after their advent, the great potential support of computers in storage and retrieval of very large amount of information had started to be explored.

Information retrieval (IR) is the research field dedicated to this issue, which can be dated back to the 1950s: in (Luhn, 1958) was first proposed the general idea of a statistical approach based on keywords to automatize the search of pertinent information. The typical problem studied in IR is the selection of documents from a possibly large collection which are pertinent to some query given by a user. Important advances in this field were made in the last of 1950s, when the ranked retrieval of documents took root. In the first IR retrieval task query was a logical combination of terms - i.e. words -, which resulted in a set of those documents that exactly matched the query. Luhn (1957) proposed an approach where at each document in the collection was assigned a score indicating its relevance to a given query. The documents were then sorted and those at the top ranks were returned to the user. Another key step was the development of the SMART Retrieval System (Salton, 1971), from which the vector space model (see next section) and other important concepts emerged. In those years were proposed two of the major and commonly used ways to give importance to the words with a statistical basis, i.e. the so called term weighting,. The first was G. K. Zipf with the law appointed as with his name (Zipf, 1949), then K. Jones Sparck, in (Sparck Jones, 1972) introduced the concept of inverse document frequency (idf); we will treat in depth this task in Section 3.2.4.

One of the need within IR systems was the indexing of documents, to improve the query response time. In (Maron, 1961) a concrete probabilistic technique for indexing of documents in categories is presented along with experimental evaluation of its accuracy: this and successive similar works (Borko and Bernick, 1963; Field, 1975) can be seen as early examples of text categorization methods.

Until 1990s, experimental evaluation of information retrieval methods was usually performed on relatively small datasets, due to the lack of availability of large collections. To support this research field, in 1992 has been launched the Text Retrieval Conference (TREC) (Harman, 1993), whose goal is to encourage large-scale information retrieval by providing the necessary infrastructure for its evaluation.

3.2 Text Representation

One of the major problems in working with text, is the intrinsic unstructured form of documents: a suitable derived representation must be used in an automated classification process. In this section are introduced the most common representation techniques for allow the application of automated algorithms to textual data.

3.2.1 Vector Space Model

The most widely used approach for the structured representation of textual data is the Vector Space Model (VSM) (Salton et al., 1975), where each document is represented by a vector in a high-dimensional space, also known as *bag-of-words*. Each dimension of this space represents a word, or *term* equivalently. In practice, the VSM representation of a textual documents dataset, creates a $m \times n$ matrix where each row represents a document and each column represent a term.

In a vector space $\mathbb{R}^n$, it is generally possible to define a *similarity function* $\mathbb{R}^n \times \mathbb{R}^n \to [0, 1]$ which maps to two vectors a numeric measure of how much they are similar. Generally, the commonly used distance measure is the *euclidean distance*. However, considering vectors representing text documents where values represent the frequency of each term, the similarity between two documents should depends on the proportions between the different values rather than on their absolute values. For instance, two documents dealing with the same topic but with very different lengths (say 100 words versus 10,000) would generally be represented with two vectors with similar orientation but different length (or *magnitude*), thus their euclidean distance would be erroneously high.

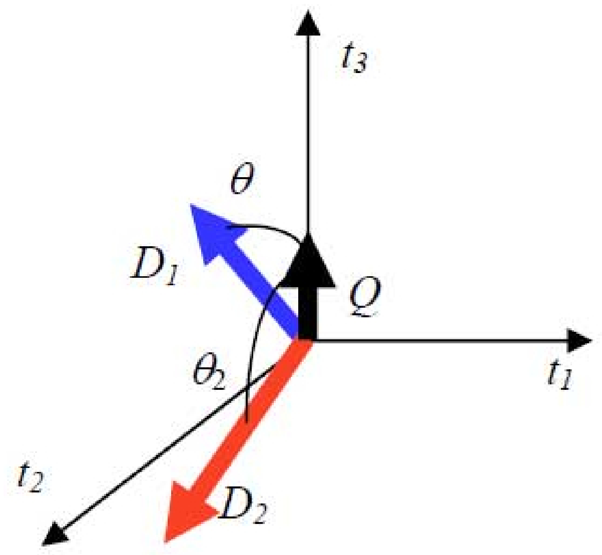


Figure 3.1 – *Example of Vector Space Model representation.*

For this reason, a suitable solution is to evaluate the *angle* between two vectors without considering their length (Θ in the example in Fig. 3.1): the smaller is the angle, the more similar are the documents. To obtain a similarity measure constrained in the range between 0 and 1, the *cosine* of such angle is considered, which is equal to 1 for a null angle and is non-negative if all the vectors are so. This measure, called *cosine similarity*, can be computed from the components of two generic vectors $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ as follows.

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{3.1}$$

As shown in the formula, the cosine similarity is in practice the dot product of the two vectors normalized by their lengths; it effectively expresses how much two bags of words are similar in the distribution of the words they contain, thus constituting a good estimation of much the topics they discuss overlap.

3.2.2 Extraction of Features

In the need of reducing text documents to vectors, is necessary to define a set of predictive features which are effective in representing the original contents of the document. A trivial choice is to consider words as they appear in the document, without considering their form or meaning. Variants of this approach aim to overcome this limitation by exploiting semantics of words, possibly using external knowledge. Other approaches consider other features than single words, such as n-grams.

Lemmas

Given a dictionary of a certain language (e.g. English), at each entry corresponds one *headword* or *lemma*, which is the canonical form representing a possibly wider set of words corresponding to the same entry. For many of such lemmas, *inflected* forms also exist, which express the same concept with differences in number, gender, tense or other categories. For example, waiter and waiters refer to the same dictionary entry, having waiter as its lemma; similarly, served and serving are both inflected forms of (to) serve.

In the basic case presented above, where words are considered as meaningless tokens, different inflected forms of a same lemma are considered as distinct words. A different approach would be to represent each word by its lemma, without distinguishing its different forms. While this theoretically implies a loss of information, the advantage is a noteworthy reduction of the number of features which, in many cases, does not lead to a practical loss of accuracy. Taking again text classification as an example, it generally matters to know which lemmas appear and their frequency, while the specific form in which they appear (e.g. how many singular and plural forms of each noun are there) has negligible importance.

The application of this approach, however, is not straightforward, as a computer must be able to infer, for each distinct word encountered in a set of documents, its corresponding lemma. This process, known as *lemmatisation*, requires prior knowledge of the specific language under analysis. Such knowledge can be very complex, including a complete set of general rules addressing common cases (e.g. -s termination for plurals in many nouns and for singular third person in verbs) and enumerations of specific cases and exceptions to such rules (e.g. mice being plural of mouse).

Stems

In the most general sense, a *stem* of a word is a part of it. Here, *stem* is used to indicate the part of any word which is common to all its inflected variants. Some words have their stem equal to their lemma, but does not apply to any word. Recalling the examples above, waiter and waiters have waiter as their stem, which is also their lemma; however, served and serving have serv as stem, which is a truncation of the lemma (to) serve. In many cases, such as this latter example, the stem of a word, contrarily to the lemma, is not itself a word of the language (although a human reading a stem should generally be able to infer the lemma of the original word).

Likely to lemmatisation, *stemming* is the process of extracting the stem of an arbitrary word. Also in this case, the process is dependent from the language and requires proper knowledge of it. However, stemming a word is usually more simple than extracting its lemma: the stem is always contained in the word itself and can usually be extracted just relying on a not too many complex set of rules.

Given the efficiency of stemming algorithms, also known as *stemmers*, in the many situations where it is not necessary to have complete words as features, stems are instead used to efficiently identify groups of similar words.

Some stemming algorithms have been proposed, even since decades ago. The most commonly used among such algorithms is the *Porter stemmer* (Porter, 1980). This algorithm, likely to others, is based on suffix stripping: a number of rules divided into some steps is defined so that each input word has its suffix removed or substituted according to rules whose preconditions match the word. For example, the -ing suffix is removed only if the resulting stem contains at least one vowel, so that serving is stemmed into serv, but sing is instead left as is.

N-grams and Phrases

An *n-gram* is in general a sequence of n consecutive elements extracted from a wider sequence: common specific cases are for $n = 2$ (*bigrams*) and $n = 3$ (*trigrams*). Elements grouped in n-grams are generally either letters or words. For example, considering letters, from the word example can be extracted the bigrams ex, xa, am, mp, pl, le. Some works use n-grams as features in place of or in addition to single words, sometimes mixing different lengths.

N-grams of letters are not practical to use in place of words, as these serve usually to identify the topic discussed in a documents and groups of letters (especially if short) would generally instead be poorly indicative of what the document discusses about. Instead, sequences of letters are usually employed in particular tasks where they can actually be effective as predictive features, such as classifying documents by language (Cavnar and Trenkle, 1994): for example, English texts can be recognized by the high frequency of some bigrams such as th and er.

N-grams of words as features are instead more similar to words, as can be informative of the topic discussed in the text. The extraction of bigrams and trigrams can be useful to represent recurring compound expressions, such as text categorization or word sense disambiguation. A field where n-grams of words are currently particularly used is sentiment analysis, where some sequences of words expressing a specific polarity are present which would be not accurately represented as single words (Pak and Paroubek, 2010).

Some works also deal with the use of *phrases* as features, which may refer either to generic sequences of words which often recur in text – which reflects the above definition of word n-grams – or to *syntactic* phrases, which are those defined by the language grammar, having a meaning on their own. In (Lewis, 1992) the use of syntactic phrases for text categorization is evaluated; it is observed anyway that phrases "have inferior statistical qualities" with respect to words, because distinct phrases are in higher number and many of them having similar meaning end up being split across many features.

Concepts

Types of features described above are extracted in a relatively trivial way from words, this with minimal processing of them. While words can generally give a good indication of the contents of the document, they anyway are not a perfect representation. A cause of this are the properties of synonymy and polysemy of natural language, implying that a meaning may be expressed by more than one word and a word, taken out of its context, may have more possible meanings. An ideal solution would be to represent a document with the *concepts* expressed in them, rather than (or in addition to) with the possibly ambiguous words.

Different methods exist to extract features of this type. Two general types of approaches can be distinguished: existing concepts can be extracted *statistically* from the documents or obtained from an external knowledge base. In the first case, the documents to be processed (or in some cases an external set of different documents) are analyzed in order to identify potential concepts according to co-occurrences between words in documents. For example, a set of words which (almost) always appear together in documents are likely to represent a unique meaning or very related ones: this set could be reduced to a single feature with no significant loss of information about the contents of documents. Some methods to extract *latent* semantic knowledge from collections of documents, the most common is the latent semantic analysis (LSA) (Deerwester et al., 1990).

The other possible approach is to use some sort of external knowledge base to obtain information about all the existing concepts and to correctly map words found within documents to them. Different sorts of knowledge bases can be used for this purpose, in Section 3.5 is proposed a novel method for text categorization making use of external knowledge bases for extracting semantic features from documents.

3.2.3 Feature Selection

As the dimensionality problem affects data analysis in various fields, among which text mining is a prominent example (a set of textual documents can create a set of thousands and more features), general techniques to automatically select an optimal subset of features are paramount; both to allow the use of machine learning algorithms, and to improve the accuracy, discarding the non-useful terms.

Standard *feature selection* algorithms are usually based on statistical observation of the data, regardless of what features actually represent. A trivial selection technique of this type is based on *document frequency*: features appearing in less than a set number N of documents are removed.

$$df(t) = |\{d : t \in d\}| \tag{3.2}$$

This serves to ignore some very rare features which can be usually ignored without significant loss of information, like rather uncommon terms or words with typos. This threshold N is usually very small, roughly near to the 0.1% of the number of documents, but even with such values a consistent number of unimportant features can be usually removed. Some extensive analysis on the impact of different feature selection metrics are provided in (Yang and Pedersen, 1997; Forman, 2003).

3.2.4 Term Weighting

Table 3.1 – *Term frequency factors.*

Term Frequency Factor	Notation	Description
1/0	BIN	Presence or absence of terms in the document
term frequency	TF	Number of times the term occurs in the document
$\log(1+tf)$	log TF	Logarithm of *tf*
$1-\frac{r}{r+tf}$	ITF	Inverse term frequency, usually $r=1$
$S(V_a)=(1-d)+d\cdot\sum_{V_b\in Conn(V_A)}\frac{S(V_b)}{\lvert Conn(V_b)\rvert}$	RW	Given a graph $G=(V,E)$, let $Conn(V)$ be the set of vertices connected to V. Typical value for d is 0.85.

In the VSM, the content of a document d_j is represented as a vector $\mathbf{w}_j=\{w_1^j,w_2^j,\ldots,w_n^j\}$ in a n-dimensional vector space $\mathbb{R}^n$, where w_i is a weight that indicates the importance of a term t_i in d_j. Terms $t_1,t_2,\ldots,t_n$ constitute a set of features, shared across all documents. In other words, each weight w_i^j indicates how much the term t_i contributes to the semantic content of d_j. Weights for each term-document couple are assigned according to a predefined term weighting scheme, which must meaningfully estimate the importance of each term within each document.

Three are the considerations discussed in the years regarding the correct assignment of weights in text categorization (Debole and Sebastiani, 2003):

1. the multiple occurrence of a term in a document appears to be related to the content of the document itself (*term frequency* factor);

2. terms uncommon throughout a collection better discriminate the content of the documents (*collection frequency* factor);

3. long documents are not more important than the short ones, normalization is used to equalize the length of documents.

Referring to these considerations, most term weighting schemes can be broken into a *local* (*term frequency*) factor and a *global* (*collection frequency*) factor. Normalization is applied on a per-document basis after computing these factors for all terms, usually by means of *cosine normalization* (eq. 3.3).

$$\mathbf{w}_j^{\text{normalized}} = \frac{1}{\sqrt{\sum_{i=1}^{n}(w_i^j)^2}} \cdot \mathbf{w}_j \qquad (3.3)$$

There are several ways to calculate the local term frequency factor, which are summarized in Table 3.1. The simplest one is binary weighting, which only considers the presence (1) or absence (0) of a term in a document, ignoring its frequency. The perhaps most obvious possibility is the number of occurrencies of the term in the document, which is often the intended meaning of "term frequency" (*tf*). Other variants have been proposed, for example the *logarithmic tf*, computed as log(1 + *tf*), is now practically the standard local factor used in literature (Debole and Sebastiani, 2003). Another possible scheme is the inverse term frequency, proposed by (**?**). Another way to assign the term frequency factor was proposed by (Hassan and Banea, 2006), inspired by the PageRank algorithm: they weight terms using a random walk model applied to a graph encoding words and dependencies between them in a document. Each word of the document is modeled in the graph as a node and an edge (bidirectional, unlike PageRank) connects two words if they co-occur in the document within a certain windows size,. In this thesis, the logarithmic term frequency (log(1 + *tf*)) has been chosen as the local factor for all experiments.

The global collection frequency factor can be *supervised* or *unsupervised*, depending whether it leverages or not the knowledge of membership of documents to categories. In the following, are summarized some of the most used and recent methods proposed in the literature of both types.

Unsupervised Term Weighting Methods

Generally, unsupervised term weighting schemes, not considering category labels of documents, derive from IR research. The most widely unsupervised method used is *tf.idf* (Sparck Jones, 1972), which (with normalization) perfectly embodies the three assumptions previously seen. The basic idea is that terms appearing in many documents are not good for discrimination, and therefore they will weight less than terms occurring in few documents. Over the years, researchers have proposed several variations in the way they calculate and combine the three basic assumptions (*tf*, *idf* and normalization), the result is the now standard variant "ltc", where $tf(t_i, d_j)$ is the *tf* factor described above denoting the importance of t_i within document d_j. In the following, the form "$t_i \in d_x$" is used to indicate that term t_i appears at least once in document d_x.

$$tf.idf(t_i, d_j) = tf(t_i, d_j) \cdot \log\left(\frac{|\mathcal{D}_{Tr}|}{|d_x \in \mathcal{D}_{Tr} : t_i \in d_x|}\right) \tag{3.4}$$

The *idf* factor multiplies the *tf* for a value that is greater when the term is rare in the collection of training documents $\mathcal{D}_{Tr}$. The weights obtained by the formula above are then normalized according to the third assumption by means of cosine normalization (Eq. 3.3).

(Tokunaga and Makoto, 1994) propose an extension of the *idf* called *Weighted Inverse Document Frequency* (*widf*), given by dividing the $tf(t_i, d_j)$ by the sum of all the frequencies of t_i in all the documents of the collection:

$$widf(t_i) = \frac{1}{\sum_{d_x \in \mathcal{D}_{Tr}} tf(t_i, d_x)} \tag{3.5}$$

(Deisy et al., 2010) propose a combination of *idf* and *widf*, called *Modified Inverse Document Frequency* (*midf*) that is defined as follows:

$$midf(t_i) = \frac{|d_x \in \mathcal{D}_{Tr} : t_i \in d_x|}{\sum_{d_x \in \mathcal{D}_{Tr}} tf(t_i, d_x)} \tag{3.6}$$

Of course the simplest choice, sometimes used, is to not use a global factor at all, setting it to 1 for all terms and only considering term frequency.

Supervised Term Weighting Methods

Since text categorization is a supervised learning task, where the knowledge of category labels of training documents is necessarily available, many term weighting methods use this information to supervise the assignment of weights to each term.

A basic example of supervised global factor is *inverse category frequency* (*icf*):

$$icf(t_i) = \log\left(\frac{|\mathcal{C}|}{|c_x \in \mathcal{C} : t_i \in c_x|}\right) \tag{3.7}$$

where "$t_i \in c_x$" denotes that t_i appears in at least one document labeled with c_x. The idea of the *icf* factor is similar to that of *idf*, but using the categories instead of the documents: the fewer are the categories in which a term occurs, the greater is the discriminating power of the term.

Within text categorization, especially in the multi-label case where each document can be labeled with an arbitrary number of categories, it is common to train one binary classifier for each one of the possible categories. For each category c_k, the corresponding model must separate its *positive examples*, i.e. documents actually labeled with c_k, from all other documents, the *negative examples*. In this case, it is allowed to compute for each term t_i a distinct collection frequency factor for each category c_k, used to represent documents in the VSM only in the context of that category.

In order to summarize the various methods of supervised term weighting, Table 3.2 shows the fundamental elements usually considered by these schemes and used in the following formulas to compute the global importance of a term t_i for a category c_k.

- A denotes the number of documents belonging to category c_k where the term t_i occurs at least once;
- C denotes the number of documents not belonging to category c_k where the term t_i occurs at least once;
- dually, B denotes the number of documents belonging to c_k where t_i does not occur;
- D denotes the number of documents not belonging to c_k where t_i does not occur.

The total number of training documents is denoted with $N = A + B + C + D = |\mathcal{D}_{Tr}|$. In this notation, the *ltc-idf* factor is expressed as:

$$idf = \log\left(\frac{N}{A + C}\right) \tag{3.8}$$

Table 3.2 – *Fundamental elements of supervised term weighting.*

	c_k	$\overline{c_k}$
t_i	A	C
$\overline{t_i}$	B	D

As suggested by (Debole and Sebastiani, 2003), an intuitive approach to supervised term weighting is to employ common techniques for feature selection, such as χ^2, *information gain*, *odds ratio* and so on. (Deng et al., 2004) uses the χ^2 factor to weigh terms, replacing the *idf* factor, and the results show that the *tf*.χ^2 scheme is more effective than *tf.idf* using a SVM classifier. Similarly (Debole and Sebastiani, 2003) apply feature selection schemes multiplied by the *tf* factor, by calling them "supervised term weighting". In this work they use the same scheme for feature selection and term weighting, in contrast to (Deng et al., 2004) where different measures are used. The results of the two however are in contradiction: (Debole and Sebastiani, 2003) shows that the *tf.idf* always outperforms χ^2, and in general the supervised methods not give substantial improvements compared to unsupervised *tf.idf*. The widely-used collection frequency factors χ^2, information gain (ig), odds ratio (or) and mutual information (mi) are described as follows:

$$\chi^2 = N \cdot \frac{(A \cdot D - B \cdot C)^2}{(A+C) \cdot (B+D) \cdot (A+B) \cdot (C+D)} \tag{3.9}$$

$$ig = -\frac{A+B}{N} \cdot \log \frac{A+B}{N} + \frac{A}{N} \cdot \log \frac{A}{A+C} + \frac{B}{N} \cdot \log \frac{B}{B+D} \tag{3.10}$$

$$or = \log \left(\frac{A \cdot D}{B \cdot C} \right) \tag{3.11}$$

$$mi = \log \left(\frac{A \cdot N}{(A+B) \cdot (A+C)} \right) \tag{3.12}$$

Any supervised feature selection scheme can be used for the term weighting. For example, the *gss* extension of the χ^2 proposed by (Galavotti et al., 2000) eliminates N at numerator and the emphasis to rare features and categories at the denominator.

$$gss = \frac{A \cdot D - B \cdot C}{N^2} \tag{3.13}$$

(Largeron et al., 2011) propose a scheme called *Entropy-based Category Coverage Difference* (*eccd*) based on the distribution of the documents containing the term and its categories, taking into account the entropy of the term.

$$eccd = \frac{A \cdot D - B \cdot C}{(A+B) \cdot (C+D)} \cdot \frac{E_{max} - E(t_i)}{E_{max}} \tag{3.14}$$

$$E(t_i) = \textit{Shannon Entropy} = -\sum_{c_k \in \mathcal{C}} tf_i^k \cdot \log_2 tf_i^k$$

where tf_i^k is the term frequency of the term t_i in the category c_k.

(Liu et al., 2009) propose a *prob-based* scheme, combining the ratios A/C measuring the relevance of a term t_i for the category c_k and A/B, since a term with this ratio high means that it is often present in the documents of c_k and thus highly representative.

$$\textit{prob-based} = \log\left(1 + \frac{A}{B} \cdot \frac{A}{C}\right) \tag{3.15}$$

Another similar scheme is *tf.rf*, proposed by (Lan et al., 2009): it takes into account the terms distribution in the positive and negative examples, stating that, for a multi-label text categorization task, the higher the concentration of high-frequency terms in the positive examples than in the negative ones, the greater the contribution to categorization.

$$rf = \log\left(2 + \frac{A}{\max(1, C)}\right) \tag{3.16}$$

Combining this idea with the *icf* factor, (Wang and Zhang, 2013) propose a variant of *tf.icf* called *icf-based*.

$$\textit{icf-based} = \log\left(2 + \frac{A}{\max(1, C)} \cdot \frac{|\mathcal{C}|}{|c_x \in \mathcal{C} : t_i \in c_x|}\right) \tag{3.17}$$

(Ren and Sohrab, 2013) implement a category indexing-based *tf.idf.icf* observational term weighting scheme, where the inverse category frequency is incorporated in the standard *tf.idf.icf* to favor the rare terms and is biased against frequent terms. Therefore, they revised the *icf* function implementing a new inverse category space density frequency ($ics_\delta f$), generating the $tf.idf.ics_\delta f$ scheme that provides a positive discrimination on infrequent and frequent terms. The inverse category space density frequency is denoted as:

$$ics_\delta f(t_i) = \log\left(\frac{|\mathcal{C}|}{\sum_{c_x \in \mathcal{C}} C_\delta(t_i, c_x)}\right) \tag{3.18}$$

$$C_\delta(t_i, c_x) = \frac{A}{A + C}$$

(Song and Myaeng, 2012) proposes a term weighting scheme that leverages availability of past retrieval results, consisting of queries that contain a particular term, retrieved documents, and their relevance judgments. They assign a term weight depending on the degree to which the mean frequency values for the past distributions of relevant and non-relevant documents are different. More precisely, it takes into account the rankings and similarity values of the relevant and non-relevant documents. (Ropero et al., 2012) introduce a novel fuzzy logic-based term weighting scheme for information extraction.

Another different approach to supervise term weighting is proposed by (Luo et al., 2011): they do not use the statistical information of terms in documents like methods mentioned above, but a term weighting scheme that exploits the semantics of categories and terms. Specifically, a category is represented by the semantic sense, given by the lexical database WordNet, of the terms contained in its own label; while the weight of each term is correlated to its semantic similarity with the category. (Bloehdorn and Hotho, 2006) propose a hybrid approach for document representation based on the common term stem representation enhanced with concepts extracted from ontologies.

3.2.5 A Supervised Variant of Inverse Document Frequency

Here is introduced a supervised variant of the widely used *tf.idf*, presented in (Domeniconi et al., 2015c). The basic idea is to avoid decreasing the weight of terms contained in documents belonging to the same category, so that words that appear in several documents of the same category are not disadvantaged, as instead happens in the standard formulation of *idf*. We refer to this variant with the name *idfec* (Inverse Document Frequency Excluding Category). Therefore, the proposed category frequency factor scheme is formulated as:

$$\textit{idfec}\ (t_i, c_k) = \log \left(\frac{|\mathcal{D}_{Tr} \setminus c_k| + 1}{\max\left(1, |d \in \mathcal{D}_{Tr} \setminus c_k : t_i \in d|\right)} \right) \tag{3.19}$$

where "$\mathcal{D}_{Tr} \setminus c_k$" denotes training documents not labeled with c_k. Using the previous notation, the formula becomes:

$$\textit{tf.idfec}\ (t_i, d_j) = tf(t_i, d_j) \cdot \log \left(\frac{C + D}{\max\left(1, C\right)} \right) \tag{3.20}$$

Note that with this variant of *idf* we can have particular cases. If the i-th word is only contained in j-th document, or only in documents belonging to c_k, the denominator becomes 0. To prevent division by 0, the denominator is replaced by 1 in this particular case.

The *tf.idfec* scheme is expected to improve classification effectiveness over *tf.idf* because it discriminates where each term appears. For any category c_k, the importance of a term appearing in many documents outside of it is penalized as in *tf.idf*. On the other side, the importance is not reduced by appearances in the positive examples of c_k, so that any term appearing mostly within the category itself retains an high global weight. This scheme is similar to *tf.rf* (Lan et al., 2009), as both penalize weights of a term t_i according to the number of negative examples where the t_i appears. The difference is in the numerator of the fraction, which values positive examples with the term in *rf* and negative ones without it in *idfec*.

To illustrate these properties, we use the following numerical example. Considering the notation shown in Table 3.2, suppose we have a corpus of 100 training documents divided as shown in Table 3.3, for two terms t_1 and t_2 and a category c_k. We can easily note how the term t_1 is very representative, and then discriminant, for the category c_k since it is very frequent within it $(A/(A + B)) = 27/30)$ and not in the rest of the documents $(C/(C + D) = 5/70)$. Similarly we can see that t_2 does not seem to be a particularly discriminating term for c_k. In the standard formulation, the *idf* is

$$\textit{idf}(t_1) = \log(100/(27 + 5)) = \log(3.125)$$

Table 3.3 – *Example of document distribution for two terms.*

	c_k	$\overline{c_k}$		c_k	$\overline{c_k}$
t_1	27	5	t_2	10	25
$\overline{t_1}$	3	65	$\overline{t_2}$	20	45

and for our best competitor *rf* is

$$rf(t_1) = log(2 + 27/5) = log(7.4)$$

while with the *idfec* we obtain

$$\textit{idfec}\ (t_1) = \log((65 + 5)/5) = \log(14)$$

For t_2 we have instead:

$$\textit{idf}(t_2) = \log(100/(10 + 25)) = \log(2.857)$$

$$rf(t_2) = log(2 + 10/25) = log(2.4)$$

$$\textit{idfec}\ (t_2) = \log((45 + 25)/25) = \log(2.8)$$

We can see that our supervised version of *idf* can separate the weights of the two terms according to the frequency of terms in documents belonging to c_k or not. In fact, while with the standard *idf* the weights of t_1 and t_2 are very similar, with *idfec* t_1 has a weight much greater than t_2 since t_1 is more frequent and discriminative for the category c_k. This kind of behavior is also exhibited by *rf*, but our method yields an even higher weight for the relevant term t_1.

In its base version, *tf.idfec* takes into account only the negative examples (C and D in Table 3.2). Instead it could be helpful, especially for the classification task, also to take into account how many documents belonging to c_k contain the term, i.e. how much the term occurs within the category more than in the rest of the collection. Considering this, in a way similar to (Wang and Zhang, 2013), we propose to mix our idea with that of the *rf* in a new version of our weighting scheme, called *tf.idfec-based* (*tf.idfec-b.*

for short) and expressed by the following formula:

$$tf.idfec\text{-}b.(t_i, d_j) = tf(t_i, d_j) \cdot \log\left(2 + \frac{A + C + D}{\max(1, C)}\right) \tag{3.21}$$

Using the example in the Table 3.3, the new term weighting scheme becomes for t_1 and t_2 respectively:

$$idfec\text{-}b.(t_1) = log(2 + ((27 + 5 + 65)/5)) = log(21.4)$$

$$idfec\text{-}b.(t_2) = log(2 + ((10 + 25 + 45)/25)) = log(5.2)$$

With this term weighting scheme, the difference in weight between a very common term (t_2) and a very discriminative one (t_1) is even more pronounced.

Experimental Setup

We performed extensive experimental evaluation to compare the effectiveness of the proposed term weighting approach with other schemes in text categorization (this task will be further analysed in Section 3.3). In the following, we describe in detail the organization of these experiments.

Benchmark Datasets

We used two commonly employed text collections as benchmarks in our experiments.

The **Reuters-21578** corpus[1] consists in 21,578 articles collected from Reuters. According to the ModApté split, 9,100 news stories are used: 6,532 for the training set and 2,568 for the test set. One intrinsic problem of the Reuters corpus is the skewed category distribution. In the top 52 categories, the two most common categories (*earn* and *acq*) contain, respectively, 43% and 25% of the documents in the dataset, while the average document frequency of all categories is less than 2%. In literature, this dataset is used considering a various number of categories: we considered two views of this corpus, *Reuters-10* and *Reuters-52* where only the 10 and the 52 most frequent categories are considered, respectively.

[1]`http://www.daviddlewis.com/resources/testcollections/reuters21578/`.

The **20 Newsgroups** corpus[2] is a collection of 18,828 Usenet posts partitioned across 20 discussion groups. Some newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc. hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). Likely to (Lan et al., 2009) we randomly selected 60% of documents as training instances and the remaining 40% make up the test set. Contrarily to Reuters, documents of 20 Newsgroups are distributed rather uniformly across categories.

Classification Process

For each dataset, all documents were pre-processed by removing punctuation, numbers and stopwords from a predefined list, then by applying the common SnowballStemmer to remaining words. In this way, we obtained a total of 16,145 distinct terms in the Reuters-21578 corpus and 61,483 in 20 Newsgroups.

We performed feature selection on these terms to keep only a useful subset of them. Specifically, we extracted for each category the p terms appearing in most of its documents, where for p all the following values were tested: 25, 50, 75, 150, 300, 600, 900, 1200, 1800, 2400, 4800, 9600. This feature selection method may be considered counterproductive since we selected the most common terms, but it is actually correct considering the use of the VSM as the terms result to be as less scattered as possible. The task of term weighting is therefore crucial to increase the categorization effectiveness, giving a weight to each term according to the category to which the documents belong.

Since we tested both supervised and unsupervised term weighting methods, we used two different procedures. For unsupervised methods we processed the training set in order to calculate the collection frequency factor for each term, which was then multiplied by the logarithmic term frequency factor (referred to as *tf* in the following) for each term in training and test set. Finally, cosine normalization (Eq. 3.3) was applied to normalize the term weights.

For supervised methods we used the multi-label categorization approach, where a binary classifier is created for each category. That is, for each category c_k, training documents labeled with it are tagged as positive examples,

[2]`http://people.csail.mit.edu/jrennie/20Newsgroups/`.

while the remaining one constitute negative examples. We computed statistical information related to c_k (as described in Table 3.2) for each term of training documents. The weight of each term was calculated multiplying its *tf* with the global factor computed on the training set; finally cosine normalization was performed.

Learning Algorithms

We chose to use support vector machines (SVM), which are usually the best learning approach in text categorization (Lan et al., 2009; Sebastiani, 2002). We used the well known SVM^{Light} implementation[3] (Joachims, 1998), testing both the linear kernel and the radial basis function (RBF) kernel.

Furthermore, to test the effectiveness of classification by varying the term weighting scheme with another algorithm, we used the Weka implementation of *Random Forest* (Breiman, 2001), chosen for both its effectiveness and its speed. As parameters we set the number of trees to $I = 10$ and the number of features to $K = 50$.

Performance Measures

We measured the effectiveness in terms of precision (p) and recall (r), defined in the usual way for text categorization (Lewis, 1995). As a measure of effectiveness that combines p and r we used the well-known F_1 measure, defined as:

$$F_1 = \frac{2 \cdot p \cdot r}{p + r} \tag{3.22}$$

For multi-label problems, the F_1 is estimated in two ways: micro-averaged F_1 and macro-averaged F_1 (Sebastiani, 2002). The *micro-*F_1 sums up the individual true positives, false positives and false negatives of the different classifiers and applies them to get the F_1. The *macro-*F_1 is instead the average of the F_1 related to each category.

Significance Tests

To evaluate the difference of performances between term weighting methods, we employed the McNemar's significance test (Dietterich, 1998; Lan et al.,

[3]`http://svmlight.joachims.org/`

Table 3.4 – *McNemar's test contingency table.*

n_{00}: Num. of istances misclassified by both classifiers f_a and f_b	n_{01}: Num. of istances misclassified by f_a but not by f_b
n_{10}: Num. of istances misclassified by f_b but not by f_a	n_{11}: Num. of istances correctly classified by both classifiers f_a and f_b

2005, 2009), used to compare the distribution of counts expected under the null hypotesis to the observed counts.

Let's consider two classifiers f_a and f_b trained from the same documents but with two different term weighting methods and evaluated using the same test set. The outcome of categorization for all test instances is summarized in a contingency table, as shown in Table 3.4.

The null hypotesis for the McNemar's significance test states that on the same test instances, two classifiers f_a and f_b will have the same prediction errors, which means that $n_{01} = n_{10}$. So the χ statistic is defined as:

$$\chi = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \tag{3.23}$$

χ is approximately distributed as a χ^2 distribution with 1 degree of freedom, where the significance levels 0.01 and 0.001 correspond respectively to the thresholds $\chi_0 = 6.64$ and $\chi_1 = 10.83$. If the null hypotesis is correct, than the probability that χ is greater than 6.64 is less than 0.01, and similarly 0.001 for a value greater than 10.83. Otherwise we may reject the null hypotesis in favor of the hypotesis that f_a and f_b have different performance and therefore the two term weighting schemes have different impact when used on the particular training set.

Experimental Results

We tested the effectiveness of classification varying the term weighting scheme on several datasets. For each dataset we tested the classification varying the number of features p selected for each category. For ease of reporting, in tables 3.5 and 3.6 we show the best result for each dataset obtained by both SVM^{Light}, in linear and radial basis function and Ran-

Table 3.5 – *Micro-averaged F_1 (in thousandths) best results obtained with each term weighting scheme ("b." is short for "based") on the three dataset with different learning algorithms. The best result for each dataset and algorithm is marked in bold.*

	tf.idfec	*tf.idfec-b.*	*tf.rf*	*tf.icf-b.*	*tf.idf*	*tf.χ^2*	*eccd*	*tf.gss*	*tf.ig*	*midf*	*tf.oddsR*	*rw*	*tf*	*bin*
Reuters-10														
SVM(LIN)	.929	**.933**	**.933**	.930	.930	.847	.839	.863	.852	.920	.918	.914	.932	.916
SVM(RBF)	.932	**.937**	**.937**	.935	.933	.879	.853	.895	.882	.923	.926	.922	.934	.924
RandomForest	**.904**	.902	.903	.899	.903	.901	.885	.901	.903	.898	.903	.899	.902	.897
Reuters-52														
SVM(LIN)	.920	**.925**	.922	.916	.917	.828	.811	.848	.822	.882	.890	.881	.912	.881
SVM(RBF)	.925	**.927**	.926	.924	.922	.848	.828	.873	.848	.886	.895	.887	.915	.887
RandomForest	.855	**.868**	.867	.853	.858	.863	.847	.861	.864	.858	.863	.856	.866	.861
20 Newsgroups														
SVM(LIN)	.754	**.759**	**.759**	.747	.741	.567	.450	.512	.520	.606	.666	.702	.709	.702
SVM(RBF)	.712	**.713**	.712	**.713**	.702	.587	.490	.555	.560	.609	.664	.674	.677	.675
RandomForest	.536	**.570**	.563	.537	.543	.568	.511	**.570**	.565	.536	.562	.556	.566	.555

domForest classifiers using each term weighting scheme.

Tables 3.5 and 3.6 show the performance of 14 different term weighting methods: *tf.idfec*, *tf.idfec-based*, *tf.rf*, *tf.icf-based*, *tf.idf*, *tf.*χ^2, *eccd*, *tf.gss*, *tf.ig*, *midf*, *tf.oddsR*, *rw*, *tf* and *bin*, in terms of *micro-*F_1 and *macro-*F_1 on the three datasets previously described. In general, our second proposed scheme *tf.idfec-based* achieved top results in all datasets and with each classifier.

In particular, on *Reuters-52*, our *tf.idfec-based* outperforms every other scheme with all classifiers in terms of *micro-*F_1: using a SVM with linear kernel, compared with the standard *tf.idf* we have an improvement of 0.8% of *micro-*F_1 and 2.2% of *macro-*F_1. Compared with standard supervised schemes such as *tf.ig* and *tf.*χ^2 we have an improvement respectively of 10.3% and 9.7% of *micro-*F_1 and 31.4% and 28% of *macro-*F_1. Furthermore, *tf.idfec-based* outperforms albeit slightly the *tf.rf* and also the *tf.icf-based* in terms of *micro-*F_1. It is possible to note a marked difference between the *micro* and *macro-*F_1 values, this is due to the strong unbalancing of the classes in this dataset; the *macro-*F_1 is strongly negatively biased by classes with few documents, for which the classification has low effectiveness, thus in this dataset the *micro-*F_1 is much more meaningful.

On *Reuters-10* we note that the results obtained with our proposed methods are very close to *tf.rf*, *tf.icf-based* and also to the standard *tf.idf* and *tf* schemes. These results show how in this dataset, consisting of only 10 categories, using a supervised term weighting method is not relevant to

Table 3.6 – *Macro-averaged F_1 (in thousandths) best results obtained with each term weighting scheme ("b." is short for "based") on the three dataset with different learning algorithms. The best result for each dataset and algorithm is marked in bold.*

	tf.idfec	*tf.idfec-b.*	*tf.rf*	*tf.icf-b.*	*tf.idf*	*tf.*χ^2	*eccd*	*tf.gss*	*tf.ig*	*midf*	*tf.oddsR*	*rw*	*tf*	*bin*
Reuters-10														
SVM(LIN)	.880	.886	**.892**	.889	.878	.692	.732	.740	.679	.841	.863	.858	.880	.863
SVM(RBF)	.888	.889	**.902**	.899	.883	.816	.755	.835	.801	.846	.876	.872	.883	.874
RandomForest	.843	.850	.853	.849	.847	.855	.825	.847	.846	.843	.849	.838	**.856**	.839
Reuters-52														
SVM(LIN)	.581	**.596**	.583	.593	.574	.316	.293	.320	.282	.244	.520	.348	.465	.348
SVM(RBF)	.611	.623	.595	**.643**	.613	.411	.335	.367	.341	.271	.550	.341	.471	.341
RandomForest	.291	**.363**	.355	.310	.308	.337	.283	.302	.341	.311	.329	.327	.361	.335
20 Newsgroups														
SVM(LIN)	.739	**.744**	**.744**	.733	.724	.530	.405	.467	.476	.578	.650	.681	.689	.681
SVM(RBF)	.691	**.692**	.688	**.692**	.682	.552	.453	.517	.521	.582	.650	.650	.654	.651
RandomForest	.496	**.537**	.532	.497	.504	.535	.470	.536	.531	.498	.527	.521	.532	.519

the effectiveness of classification: this can be deduced from the difference between the effectiveness of standard *tf.idf* and our supervised versions on *Reuters-52*, which contains the same documents of *Reuters-10* but labeled with more categories. However, our schemes outperform standard supervised term weighting by more than 10%.

The results on *20 Newsgroups* show that *tf.idfec-based* obtains the best *micro-*F_1, in parity with *tf.rf* using linear SVM, with *tf.icf-based* using radial kernel SVM and with *tf.gss* using RandomForest. Using linear SVM, the best *micro-*F_1 of *tf.idfec-based* is higher by 1.8% compared to that obtained from *tf.idf* and of 23.9% compared with a standard supervised method like *tf.ig*.

Observing all the results, we can see that our first proposed scheme *tf.idfec* obtains results always in line but slightly lower than the *tf.idfec-based* variant. This evidently means that considering only the information about the negative categories of the terms is not enough to achieve an optimal accuracy. Conversely, adding information about the ratio between A and C (from notation of Table 3.2), it is obtained an optimal mixture that leads to better classification results, using either SVM or RandomForest classifiers.

We employ the McNemar's significance test to verify the statistical difference between performances of the term weighting schemes. We report these results in Table 3.7, where each column is related to a dataset and a classifier and the term weighting schemes are shown in decreasing order of effectiveness, separated according to the significance of difference between

Table 3.7 – *McNemar's significance test results. Each column is related to a dataset and a supervised classifier, the term weighting schemes are shown in decreasing order of effectiveness, separating groups of schemes by significance of differences in their perfomances. Results for RandomForest on Reuters-10 are omitted as no significant difference is observed between them.*

Reuters-10		*Reuters-52*			*20 Newsgroups*		
SVM(LIN)	SVM(RBF)	SVM(LIN)	SVM(RBF)	RandomFor.	SVM(LIN)	SVM(RBF)	RandomFor.
tf.idfec-b.	***tf.idfec-b.***	***tf.idfec-b.***	***tf.idfec-b.***	***tf.idfec-b.***	***tf.idfec-b.***	***tf.idfec-b.***	***tf.idfec-b.***
tf.rf	*tf.rf*	*tf.rf*	*tf.rf*	*tf.rf*	*tf.rf*	*tf.icf-b.*	*tf.gss*
tf	*tf.icf-b.*	***tf.idfec***	***tf.idfec***	*tf*	***tf.idfec***	***tf.idfec***	*tf.*χ^2
tf.icf-b.	*tf*	*tf.idf*	*tf.icf-b.*	*tf.ig*	*tf.icf-based*	*tf.rf*	*tf*
tf.idf	*tf.idf*	*tf.icf-b.*	*tf.idf*	*tf.oddsR*	*tf.idf*	*tf.idf*	*tf.ig*
tf.idfec	***tf.idfec***	*tf*	*tf*	*tf.*χ^2	*tf*	*tf*	*tf.rf*
midf	*tf.oddsR*	*tf.oddsR*	*tf.oddsR*	*tf.gss*	*bin*	*bin*	*tf.oddsR*
tf.oddsR	*bin*	*bin*	*bin*	*bin*	*rw*	*rw*	*rw*
bin	*midf*	*rw*	*rw*	*tf.idf*	*tf.oddsR*	*tf.oddsR*	*bin*
rw	*rw*	*midf*	*midf*	*midf*	*midf*	*midf*	*tf.idf*
tf.gss	*tf.gss*	*tf.gss*	*tf.gss*	*rw*	*tf.*χ^2	*tf.*χ^2	*tf.icf-b.*
tf.ig	*tf.ig*	*tf.ig*	*tf.*χ^2	***tf.idfec***	*tf.ig*	*tf.ig*	***tf.idfec***
tf.χ^2	*tf.*χ^2)	*tf.*χ^2	*tf.ig*	*tf.icf-b.*	*tf.gss*	*tf.gss*	*midf*
eccd	*eccd*	*eccd*	*eccd*	*eccd*	*eccd*	*eccd*	*eccd*

them. Schemes not separated with lines do not give significantly different results, a single line denotes that the schemes above perform better than the schemes below with a significance level between 0.01 and 0.001 (commonly indicated as "$A > B$"), while a double line denotes a significance level better than 0.001 ("$A >> B$"). To save space, we do not report the results on the *Reuters-10* corpus with RandomForest, because there are no significant statistical differences between them. From Table 3.7 we can note that our proposed *tf.idfec-based* scheme always provides top effectiveness. In addition, this table shows that with SVM classifiers, either linear or RBF kernel, some term weighting methods are more efficient than others. The best methods in general seem to be the latest supervised methods, such as *tf.idfec-based*, *tf.rf*, *tf.icf-based* and *tf.idfec*. Instead, using RandomForest, the classic supervised methods seem to work better, with results comparable or slightly below with respect to *tf.idfec-based*.

Let's now observe how classification effectiveness varies by changing the number of features considered to create the dictionary of the VSM. For reasons of readability of the graph, we do not show all the 14 term weighting

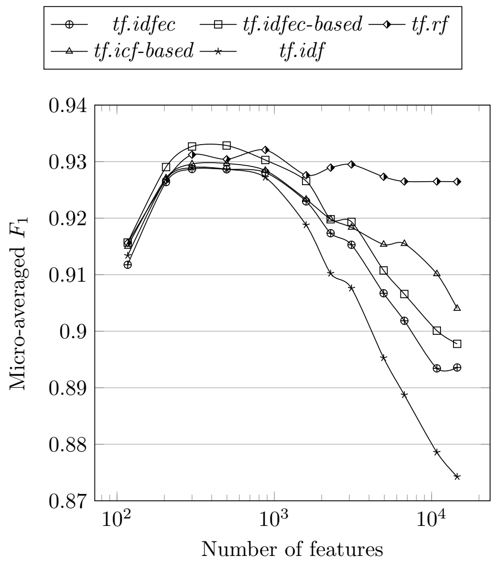


Figure 3.2 – *Results obtained on Reuters-10 varying the number of top p features selected per category using linear SVM classifier. The X axis (in logarithmic scale) indicates the resulting total number of features.*

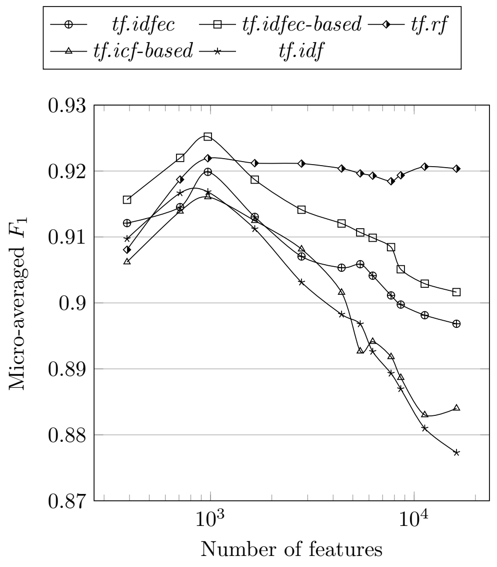


Figure 3.3 – *Results obtained on Reuters-52 varying the number of top p features selected per category using linear SVM classifier. The X axis (in logarithmic scale) indicates the resulting total number of features.*

methods investigated, but only the five most recent and with better results, i.e. *tf.idfec*, *tf.idfec-based*, *tf.rf*, *tf.icf-based*, *tf.idf*. For each dataset, we show the results obtained using an SVM classifier with a linear kernel.

Figures 3.2 and 3.3 show that on both views of Reuters corpus, when using *tf.idfec-based* we obtain the best results by using few features per category, considering the variations of p described in Section 3.2.5. We note that the best results on *Reuters-10* are obtained with 150 features per category and on *Reuters-52* with 75 features, corresponding respectively to overall dictionary sizes of 498 and 970. From these plots, however, we can see as the effectiveness of the classification using *tf.idfec-based* deteriorates by increasing the number of features considered, therefore introducing terms less frequent and discriminative. Analysing the behavior of the schemes from which *idfec-based* takes its cue, i.e. standard *tf.idf* and *tf.rf*, we note that this performance degradation is probably due to the *idf* factor of the

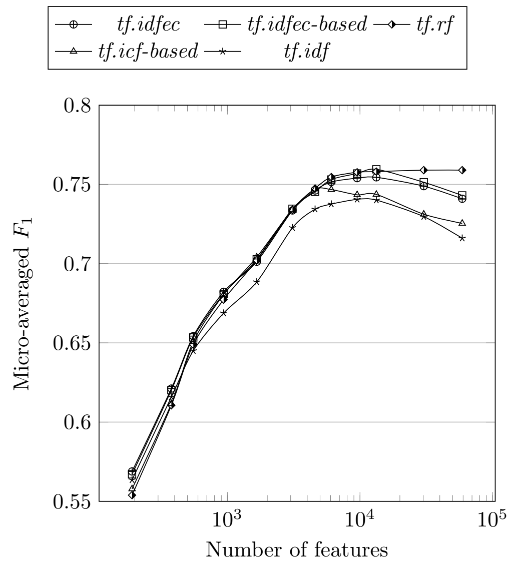


Figure 3.4 – *Results obtained on 20 Newsgroups varying the number of top p features selected per category using linear SVM classifier. The X axis (in logarithmic scale) indicates the resulting total number of features.*

weight, as even the *tf.idf* has the same type of trend results, while *tf.rf* seems to remain stable at values comparable to the best results also by increasing the dictionary size.

Figure 3.4 shows how to obtain the best results with the *20 Newsgroups* corpus is necessary a greater number of features. Using *tf.idfec-based* we obtain the best result with a dictionary of about 10000 features; after that the efficiency shows a slight decrease, but more moderate than that shown in the Reuters dataset.

Discussions

In this section two novel supervised variations of the *tf.idf* term weighting approach are proposed. In the first one, we employ the basic idea of *tf.idf*, but considering only documents not belonging to the modeled category: this prevents having a low *idf* factor if a term is largely present in the documents of the category under consideration. The second variant uses a mixture of the previous idea and the relevance frequency *rf* in order to consider also the amount of documents belonging to the category in which the term occurs.

We performed extensive experimental studies on two datasets, i.e. *Reuters* corpus with either 10 or 52 categories and *20 Newsgroups*, and three different classification methods, i.e. SVM classifier with linear and RBF kernel functions and RandomForest. The obtained results show that the *tf.idfec-based* method that combines *idfec* and *rf* generally gets top results on all datasets and with all classifiers. Through statistical significance tests, we showed that the proposed scheme always achieves top effectiveness and is never worse than other methods. The results put in evidence a close competition between our *tf.idfec-based* and the *tf.rf* schemes; in particular, the best results obtained with the different datasets and algorithms, varying the amount of feature selection, are very similar, but with some differences: *tf-rf* seems to be more stable when the number of features is high, while our *tf.idfec-based* gives excellent results with few features and shows some decay (less than 4%) when the number of features increases.

3.3 Text Categorization

Generally speaking, *text categorization* (or *classification*, hence abbreviated in TC) is the task of labeling each text *document* of a set with *categories* (or *classes*, as commonly named in general data classification). Categories assigned to each document must be a subset of a priorly defined set of known categories, in contrast e.g. to *text clustering*, where no such set is given.

Throughout the thesis, the global set of documents of a particular collection is usually denoted with $\mathcal{D}$, while $\mathcal{C}$ denotes the set of known categories with which documents can be labeled. Given this notation, the general goal of text classification can be formalized as giving in output a *labeling* $\hat{\mathcal{L}} : \mathcal{D} \times \mathcal{C} \to \{0, 1\}$, indicating for each couple $(d, c) \in \mathcal{D} \times \mathcal{C}$ whether d should be labeled with c (1) or not (0). The set $\mathcal{C}$ of categories is decided according to some practical criterion. In the most common case, each category represents a *topic* which is discussed in some documents in the collection: the need is to extract the subset of topics significantly treated by each of the documents.

The problem of TC has been extensively investigated in past years. While earlier works were based on manually defining simple rules to infer the topic of each document on the base of prominent keywords, subsequent literature is mostly based on machine learning (ML): pre-labeled training documents are used to automatically build a knowledge model, used to classify further documents. The most known methods and techniques for TC have their roots in information retrieval (IR), the general activity of efficiently retrieving resources from a collection which are relevant to specific needs. The use of IR and ML techniques to classify documents is extensively surveyed by (Sebastiani, 2002).

Text documents are usually classified according to their topic: for example, a news site may define categories like “business”, “world”, “science” and so on. Other possibilities exist, such as classification by author (De Vel et al., 2001) or by language (Cavnar and Trenkle, 1994), for which ad hoc techniques are often employed. A research branch currently of high interest is *sentiment analysis* (further described in Section 3.6), where text documents like reviews, forum posts or even short messages (from Twitter, for example) must be sorted in positive and negative opinions regarding a specific subject, which might be an individual, a brand, a political party, etc. (Pang and Lee, 2004; Pan et al., 2010).

Earlier literature deals with *flat classification*, using a flat set of classes, without any structural relationship upon it (see (Apté et al., 1994; Freund and Schapire, 1997; Dumais et al., 1998; Joachims, 1998; Platt, 1999; Guo et al., 2006; Sebastiani, 2002) and references therein). Most of these works treat *multi-label* classification, where any document may be labeled with any number of categories, although *single-label* cases exist where to each document must be assigned exactly one label.

To apply standard ML methods, a data pre-processing step is generally needed to extract suitable features from the unstructured text documents. As above described, *bag of words* is a widely used representation. Different supervised learning methods can be applied to train a model to classify documents represented as bags of words: notable approaches which have proven to be effective for TC include decision rules (Apté et al., 1994), Naïve Bayes (McCallum et al., 1998) and Support Vector Machines (Joachims, 1998).

An approach somewhat more distant from ML is the *Rocchio method*, also known as *nearest centroid* classification: it is based on building for each category a *prototype*, which can be represented like regular documents and compared to them, typically by means of cosine similarity, so that the right category for a new document is the one having the representation most similar to it (Hull, 1994; Joachims, 1997). In practice, a document-like representation of each category is built by averaging representations for all documents relevant to it and optionally subtracting the corresponding average for unrelated documents: different weights can be set for the two components. By just averaging related documents, a centroid for each category is obtained. In the following sections, will be proposed some novel classification techniques using a similar approach.

3.3.1 Hierarchical Categorization

In earlier works, categories were considered as a *flat* set, with no relationships between them. In the last decade, to better address real use cases and to achieve higher effectiveness and efficiency, increasing attention is being given to *hierarchical* classification of documents: categories are arranged in a usually rooted tree-like taxonomy, where each of them treats a slightly more specific topic with respect to its parent category and may have any number of subcategories dealing in turn with even more specific arguments. A hierarchical taxonomy can be expressed formally as a partially ordered set $\langle \mathcal{C}, \prec \rangle$, where $\mathcal{C}$ is the set of categories and $\prec \subset \mathcal{C} \times \mathcal{C}$ is the asymmetric, transitive *is-a* relationship. In practice, $c_d \prec c_a$ means that c_d represents a specific topic within the wider discussion area represented by c_a: in this case, c_a is said to be an *ancestor* of c_d, which is in turn a *descendant* of c_a. In this formalization, the (direct) *parent* of a non-root category c_d is the only category c_p satisfying $c_d \prec c_p \wedge \nexists \gamma \in \mathcal{C} : c_d \prec \gamma \prec c_p$, while *children* of a category c_p are those categories whose c_p is parent.

The use of a hierarchical taxonomy of categories is often useful to better organize documents, allowing to find specific ones starting from general discussion areas and progressively narrowing down the domain to the topic of interest. A typical example of this organization are web directories, where great numbers of websites are organized in a fine-grained taxonomy of categories which can be browsed by the user, from the home page presenting the top-level categories to the sub-pages of specific topics listing general related websites and possibly even more specific sub-categories where other websites are distributed.

(Silla and Freitas, 2011) give a general review of hierarchical TC literature, while (Qi and Davison, 2009) focus on web pages classification, which is a common application.

The hierarchical setting of categories has some variants: (Sun and Lim, 2001) distinguish between taxonomies organized specifically as trees or more generally as directed acyclic graphs (where nodes may have multiple parents) and between allowing or denying documents to be classified in intermediate (non-leaf) nodes of the taxonomy. The same work discerns two major approaches to hierarchical text classification. In the *big-bang* approach a global feature selection is performed and a single classifier is used on the whole hierarchy, while in the most common *top-down* approach a different classifier is used for each node: the computational effort to build classifiers is therefore higher, but the most suitable features can be selected for each node.

To classify a document with a *top-down* approach, an iterative process is generally used starting from the root of the hierarchy and progressively descending to more specific categories of the tree to find the potentially correct one. Typical issues in this process are correctly classifying very specific documents at high levels of the hierarchy and, if documents can be classified in intermediate nodes, choosing at each level whether to stop at the current node or keep descending the tree.

Some earlier experiments were performed on flat collections like Reuters-21578, using categories as leafs in a small hierarchy built ad hoc. For instance, (D'Alessio et al., 2000) use a local classifier for each node of the hierarchy, using two feature sets created from positive and negative example documents.

Among the first experiments on real hierarchical collections, (Dumais and Chen, 2000) create one SVM classifier for each node in a two-levels hierarchy of summaries of web documents, using feature sets created with documents and categories of the same node, assigning multiple leaf nodes to the test documents. (Liu et al., 2005) also use multiple SVMs trained for each category. (Cai and Hofmann, 2004) leverage knowledge of categories relationships in the whole SVM classification architecture. In (Cesa-Bianchi et al., 2006b,a), the authors propose improvements in the classification process, like an incremental classifier (Cesa-Bianchi et al., 2006b) and a refined evaluation scheme (Cesa-Bianchi et al., 2006a). (Xue et al., 2008b) propose a strategy based on pruning the original hierarchy which first computes the similarity between the test document and all other documents, and then classifies the document in a pruned hierarchy. (Sun et al., 2004) tested three methods to limit the blocking problem in top-down approaches, that is the problem of documents wrongly rejected by the classifiers at higher-levels of the taxonomy. (Bennett and Nguyen, 2009) propose a *expert refinements* technique that uses cross-validation in the training phase to obtain a better estimation of the true probabilities of the predicted categories. (Ceci and Malerba, 2007) make a comparison between using a flat classifier and a local hierarchy classifier created in each parent node, using both SVM and Naïve Bayes approaches. (Ruiz and Srinivasan, 2002) propose a variant of the Hierarchical Mixture of Experts model, making use of two types of neural networks for intermediate nodes and for leaves.

Also as regards the use of global classifiers the literature offers various reference works. (Cai and Hofmann, 2003) investigate the use of concept-based document representations to supplement word- or phrase-based features. Weak hypotheses are created based on terms and concepts features, which are combined using Adaboost. (Vens et al., 2008) transform the classification output into a vector with boolean components corresponding to the possible category. They also use a distance-based metric to measure the similarity of training examples in the classification tree.

(Li et al., 2012c) propose an active learning method to significantly reduce the number of training documents needed: this is based on iteratively picking a limited number of informative unlabeled documents from a pool of available data and getting a "yes" or "no" answer for each category from a specialized oracle (human expert).

While the basic bag-of-words approach considers words only in their lexical form, many works are based on leveraging their semantic content to improve the accuracy of text classification. Some works use statistical techniques like Latent Semantic Indexing (LSI; or Analysis, LSA) to capture the underlying correlations between words from training documents. (Hull, 1994) applied LSI in conjunction with the Rocchio method for classification. (Yang, 1995) used Singular Value Decomposition to reduce noise in latent semantic structures. (Zelikovitz and Hirsh, 2001) test the benefits of leveraging additional unlabeled documents when applying LSI. Newer techniques like Probabilistic Latent Semantic Analysis (Hofmann, 1999) and Latent Dirichlet Allocation (Blei et al., 2003) model the presence of different topics in the documents, each represented by different words.

While these statistical techniques learn approximated semantic knowledge from the training documents themselves, another possibility is to use external knowledge bases. Generally, an external knowledge base may be any kind of resource and may be used in different ways. Structured or semi-structured resources are often used to obtain accurate semantic information enabling to spot relationships between words which might not be learned from the documents themselves. A widely used resource is WordNet, a lexical database for the English language which indicates concepts as sets of synonyms and mutual semantic relationships between them (Miller, 1995): we'll return on it later when describing how it is used in our method.

Many works make use of semantic knowledge by substituting or enriching the representations of documents with concepts expressed by terms. For example, (Scott and Matwin, 1998) use WordNet synsets (concepts) as features, weighting them also by respective hypernyms found in the text. (Gabrilovich and Markovitch, 2005) tested the introduction of additional features extracting hundreds of thousands of concepts from domain-specific and common-sense world knowledge sources, like DMoz (Gabrilovich and Markovitch, 2005) and Wikipedia (Gabrilovich and Markovitch, 2007). In a similar way, (Tao et al., 2012) perform unsupervised classification by extracting categories from generalized world knowledge. (Peng and Choi, 2005) generate further features from relationships between WordNet synsets and also create representations for categories which are tested for similarity with those for documents: our method resembles this approach for these aspects.

There also are other approaches based on the creation of structured representations for each category, other than for single documents. In a base case, each topic may be represented with a set of manually-picked keywords. The method presented by (McCallum, 1999) assigns few keywords for each category in a small hierarchy as a preliminary step to train a Naïve Bayes classifier from a set of initially unlabeled documents. (Barak et al., 2009) build topics representations by using the name of each category and finding terms semantically related to it; documents are then classified in a flat context by comparing them to categories by cosine similarity, as also done by (Peng and Choi, 2005).

3.3.2 Cross-Domain Text Classification

Typical approaches of text categorization require the availability of a training set of documents, from which necessary knowledge is inferred. A training set generally must have some characteristics in order to be usable and to yield optimal results: it must contain an adequate amount of documents, which must be as exactly as possible representative of the documents which will be classified and must be coherently labeled with the same categories which are going to be assigned to new documents. To be representative, training documents should intuitively contain the same words or, more generally, the same features which will be extracted to predict categories of subsequent documents to be classified, which are refered to as *target* documents. In other words, can be said that training documents must be within the same *domain* of the target documents. Intuitively, the domain of the documents is the context within they are produced and/or consumed and dictates the words which are used and the categories under which are or must be filed.

Retrieving a set of documents of the exact same domain of the target documents can often be difficult. As discussed previously, a solution would be to manually label some target documents, but creating a suitable training set may require to label a great amount of documents, thus implying a significant amount of human effort. However, in some cases, a set of labeled documents of a slightly different domain may be available. The difference between the domains may consist in the use of some different terms or in the organization within categories representing slightly different concepts. Considering the general setting of the problem, the traditional techniques seen so far for text categorization might be applied to infer a classifier from available training documents and apply it to target documents. However, due to the differences between the two domains, this would likely not result in an accurate classification, as many of the features known by the classifier would not be found within the target documents. Cases like these would require specific methods to somehow transfer the knowledge extracted from the available training data to the target domain. Throughout the last decade, techniques for *transfer learning* have been devised to address these cases (Pan and Yang, 2010). Transfer learning generally involves solving a problem in a *target domain* by leveraging knowledge from a *source domain* whose data is fully available. *Cross-domain text categorization* refers to the specific task of classifying a set of target documents in predefined categories using as support a set of pre-labeled documents of a slightly different domain.

Contrarily to traditional text categorization problems, where target documents are generally assumed to be not known in advance, typical cross-domain methods imply that unlabeled target documents must be given in advance, at least in part, as some knowledge of the target domain is necessary. Also, the majority of cross-domain methods consider a single-label setting (i.e. one and only one category label to each document), which is assumed by default for the rest of this section.

Formally, an algorithm for cross-domain text classification has as its input a set $\mathcal{D}_S$ of *source documents* constituting the *source domain*, a set $\mathcal{D}_T$ of *target documents* making up the *target domain*, we denote with $\mathcal{D} = \mathcal{D}_\mathrm{S} \cup \mathcal{D}_\mathrm{T}$ their union, a set $\mathcal{C}$ of *categories* and a labeling $C_S : \mathcal{D}_S \rightarrow \mathcal{C}$ associating a single category to each source document. The required output is a predicted labeling $C_T : \mathcal{D}_T \rightarrow \mathcal{C}$ for documents of the target domain.

For what concerns the relationship between the two domains, in the general case of transductive transfer learning where cross-domain text categorization falls in (see below), they must share the same feature space $\mathcal{X}$ and the same class labels $\mathcal{Y}$: in the case of text documents, the first condition can be satisfied simply by selecting the same terms for source and target domain. The common assumption on the difference between the two domains is that the labels are equally conditioned by the input data, which though is distributed differently in them. Denoting with X_S and Y_S data and labels for the source domain and with X_T and Y_T those for the target domain, we have $P(Y_S|X_S) = P(Y_T|X_S)$, but $P(X_S) \neq P(X_T)$: this condition is known as *covariate shift* (Shimodaira, 2000).

Often, two major approaches to transductive transfer learning are distinguished: (Pan et al., 2010) and other works refer to them as *instance-transfer* and *feature-representation-transfer*.

Instance-transfer-based approaches generally work by re-weighting instances (data samples) from the source domain to adapt them to the target domain, in order to compensate the discrepancy between $P(X_S)$ and $P(X_T)$: this generally involves estimating an *importance* $\frac{P(x_S)}{P(x_T)}$ for each source instance x_S to reuse it as a training instance x_T under the target domain.

Some works mainly address the related problem of sample selection bias, where a classifier must be learned from a training set with a biased data distribution. (Zadrozny, 2004) analyzes the bias impact on various learning methods and proposes a correction method using knowledge of selection probabilities.

The *kernel mean matching* method (Huang et al., 2007) learns re-weighting factors by matching the means between the domains data in a reproducing kernel Hilbert space (RKHS); this is done without estimating $P(X_S)$ and $P(X_T)$ from a possibly limited quantity of samples. Among other works operating under this restriction there is the *Kullback-Liebler importance estimation procedure* (Sugiyama et al., 2007), a model to estimate importance based on minimization of the Kullback-Liebler divergence between real and expected $P(X_T)$.

Among works specifically considering text classification, (Dai et al., 2007b) trains a Naïve Bayes classifier on the source domain and transfers it to the target domain through an iterative Expectation-Maximization algorithm. In (Gao et al., 2008) multiple classifiers are trained on possibly multiple source domains and combined in a *locally weighted ensemble* based on similarity to a clustering of the target documents to classify them.

On the other side, feature-representation-transfer-based approaches generally work by finding a new feature space to represent instances of both source and target domains, where their differences are reduced and standard learning methods can be applied.

The *structural correspondence learning* method (Blitzer et al., 2006) works by introducing *pivot* features and learning linear predictors for them, whose resulting weights are transformed through Singular Value Decomposition and then used to augment training data instances. The paper (Daumé III, 2007) presents a simple method based on augmenting instances with features differentiating source and target domains, possibly improvable through nonlinear kernel mapping. In (Ling et al., 2008a) a spectral classification-based framework is introduced, using an objective function which balances the source domain supervision and the target domain structure. With the *Maximum Mean Discrepancy (MMD) Embedding* method (Pan et al., 2008), source and target instances are brought to a common low-dimensional space where differences between data distributions are reduced; *transfer component analysis* (Pan et al., 2011) improves this approach in terms of efficiency and generalization to unseen target data.

The following works are focused on text classification. In (Dai et al., 2007a) an approach based on co-clustering of words and documents is used, where labels are transferred across domain using word clusters as a bridge. The *topic-bridged PLSA* method (Xue et al., 2008a) is instead based on Probabilistic Latent Semantic Analysis, which is extended to accept unlabeled data. In (Zhuang et al., 2011) is proposed a framework for joint non-negative matrix tri-factorization of both domains. *Topic correlation analysis* (Li et al., 2012b) extracts both shared and domain-specific latent features and groups them, to support higher distribution gaps between domains.

Likely to traditional text classification, some methods leverage external knowledge bases: these can be helpful to link knowledge across domains. The method presented in (Wang et al., 2008a) improves the cited co-clustering-based approach (Dai et al., 2007a) by representing documents with concepts extracted from Wikipedia. The *bridging information gap* method (Xiang et al., 2010) exploits instead an auxiliary domain acting as a bridge between source and target, using Wikipedia articles as a practical example. These methods usually offer very high performances, but need a suitable knowledge base for the context of the analyzed documents, which might not be easily available for overly specialized domains.

Beyond the presented works where domains differ only in the distribution of terms, methods for *cross-language* text classification exist, where source and target documents are written in different languages, so that there are few or no common words between the two domains. This scenario generally requires either some labeled documents for the target domain or an external knowledge base to be available: a dictionary for translation of single terms is often used. As examples, in (Ling et al., 2008b) is presented an approach based on *information bottleneck* where Chinese texts are translated into English to be classified, while the method in (Prettenhofer and Stein, 2010) is based on the structural correspondence learning cited above (Blitzer et al., 2006).

3.4 Cross-Domain Text Classification through Iterative Refining of Target Categories Representations

In the following, a novel approach to cross-domain text categorization is proposed (Domeniconi et al., 2014b, 2015d), summarily based on creating representations of categories from the source domain and adapting them to the target domain by iteratively retrieving highly related documents and using them to refine the representations.

3.4.1 Method

Basically, the method is based on *nearest centroid* classification, where explicit *representations* or *profiles* are built for each category in the same form of bags of words for documents, so that each new document is assigned to the category having the profile most similar to its bag of words. These profiles are often simply constituted by the mean point (centroid) of bags for known documents belonging to respective categories. As a measure to compare vector-like representations of documents and categories, *cosine similarity* (Eq. 3.1) is usually employed.

In standard text categorization, these profiles can be created from documents of the training set and this potentially leads to optimal representations of categories. In the cross-domain case, profiles can be created from documents of the source domain, whose labels for each are known, but these are presumably not optimal to classify documents under the target domain, so some sort of adaptation is needed to use them.

The proposed idea is to use profiles extracted from the source domain as a starting point, expecting that at least some documents of the target domain will be significantly similar to them. Considering these documents as correctly classified, updated profiles for categories can be computed by averaging them, assuming them to constitute better representations of the categories in the target domain. By iteratively repeating this step, somewhat similarly to what happens in the *k-means* clustering algorithm, category profiles can be furtherly improved, as they are progressively extracted from more documents. After a number of iterations, the final profiles are used as a reference to classify documents.

Document Pre-Processing and Term Weighting

As a first step, as typically happens in text categorization, a pre-processing phase is run where all documents in $\mathcal{D}$ are reduced to bags of words.

For each document, single words are extracted, then those shorter than three letters or found in a list of *stopwords* (articles, prepositions, etc.) are removed. Among all distinct words found within documents, likely to many other cross-domain methods, only those appearing in at least three of them are used as features or *terms*: the global set of selected features is denoted with $\mathcal{W}$. Each document d is then represented with a vector $\mathbf{w}_d \in \mathbb{R}^{|\mathcal{W}|}$, containing for each term t a weight $w_{t,d}$ of its relevance within the document.

As described in Section 3.2.4, each weight $w_{t,d}$ is generally obtained by product of a *local* factor denoting the relevance of t in d itself and a *global* factor equal for all documents, indicating how important is t across them. These schemes can be combined in different ways: Sect. 3.4.3 will initially present various composite schemes before establishing one of them to be used.

Cross-Domain Learning Algorithm

After processing documents, a profile $\mathbf{w}_c^0$ for each category $c \in \mathcal{C}$ is computed as the centroid of source documents labeled with c, whose set is denoted with R_c^0.

$$R_c^0 = \{d \in \mathcal{D}_\mathrm{S} : C_S(d) = c\} \tag{3.24}$$

$$\mathbf{w}_c^0 = \frac{1}{|R_c^0|} \sum_{d \in R_c^0} \mathbf{w}_d \tag{3.25}$$

The "0" index denotes that these are *initial* profiles, which constitute the starting point for the iterative phase, which is explained in the following. The i index in the following is the iterations counter, which starts from 0.

Firstly the similarity score $s^i(d,c)$ between each target document $d \in \mathcal{D}_\mathrm{T}$ and each category $c \in \mathcal{C}$ is computed. In the base method, it is simply the cosine similarity between the bag of words for d and the current profile for c.

$$s^i(d,c) = \cos(\mathbf{w}_d, \mathbf{w}_c^i) \tag{3.26}$$

This similarity is considered as the *absolute* likelihood of c being related to d, i.e. the probability that it is the correct category for d. In order to be confident in the assignment of a category c to a document d, the relevant score $s^i(d,c)$ must be significantly higher than those for the same document and other categories. To evaluate where this is the case, *relative* scores are computed by normalizing those of each document for all categories: this makes them constitute in practice a probability distribution among categories.

$$p^i(d,c) = \frac{s^i(d,c)}{\sum_{\gamma \in \mathcal{C}} s^i(d,\gamma)} \tag{3.27}$$

The value of $p^i(d,c)$ indicates the estimated probability of c being the correct category for d, considering similarities between d and all categories. For each document, the most likely category is obviously that with the highest score: we denote with $A_c^i \subseteq \mathcal{D}_\mathrm{T}$ the set of target documents for which c is the predicted category. However, the score could indicate more or less certainty about the prediction. Setting a threshold ρ, we can define for each category $c \in \mathcal{C}$ a set $R_c^{i+1} \subseteq A_c^i$ of documents for which the assignment of the c label is "sure enough".

$$A_c^i = \{d \in \mathcal{D}_\mathrm{T} : c = \operatorname*{argmax}_{\gamma \in \mathcal{C}} p^i(d, \gamma)\} \tag{3.28}$$

$$R_c^{i+1} = \{d \in A_c^i : p^i(d, c) > \rho\} \tag{3.29}$$

R_c^{i+1} is a set of *representative* documents for the category c in the target domain. A new profile for the category is built by averaging these documents.

$$\mathbf{w}_c^{i+1} = \frac{1}{|R_c^{i+1}|} \sum_{d \in R_c^{i+1}} \mathbf{w}_d \tag{3.30}$$

At this point, conditions for the termination of the iterative phase are checked. A maximum number N_I of iterations is set to ensure that the algorithm does not run for an excessively long time. However, after a limited number of iterations, category profiles usually tend to cease to change from one iteration to another. At a certain iteration, if category profiles are identical to those of the previous one, the same representative documents as before will be selected and the same profiles will keep to be computed, so in this case the iterative phase can be safely terminated. This leads to the following termination condition.

$$\forall c \in \mathcal{C} : \mathbf{w}_c^{i+1} = \mathbf{w}_c^i \tag{3.31}$$

If this condition does not hold and the number of finished iterations $(i + 1)$ is below N_I, all steps from (3.26) up to here are repeated with the iteration counter i incremented by 1. Otherwise, the iterative phase terminates with an iteration count $n_\mathrm{I} = i + 1$. When this happens, the final predicted category for each target document d is computed as the one whose latest computed profile is most similar to the bag of words for d.

$$\hat{C}_T(d) = \operatorname*{argmax}_{c \in \mathcal{C}} \cos(\mathbf{w}_d, \mathbf{w}_c^{n_\mathrm{I}}) \tag{3.32}$$

Other than to documents of the target domain known and used in the iterative phase, this formula can be applied to any previously unseen document of the same domain, comparing its bag of words to final category profiles.

Computational Complexity

The process performs many operations on vectors of length $|\mathcal{W}|$: using suitable data structures, both storage space and computation time can be bound linearly w.r.t. the mean number of non-zero elements, which will be denoted with l_D and l_C for bags of words for documents and categories, respectively. By definition, we have $l_\mathrm{D} \leq |\mathcal{W}|$ and $l_\mathrm{C} \leq |\mathcal{W}|$; from our experiments we also generally observed $l_\mathrm{D} \ll l_\mathrm{C} < |\mathcal{W}|$.

Initial profiles for categories are built in $O(|\mathcal{D}_\mathrm{S}| \cdot l_\mathrm{D})$ time, as all values of all bags of words for documents must be summed up. Cosine similarity between vectors with l_D and l_C non-zero elements respectively can be computed in $O(l_\mathrm{D} + l_\mathrm{C})$ time, which can be written as $O(l_\mathrm{C})$ given that $l_D < l_C$.

In each iteration of the refining phase, the method computes cosine similarity for $N_\mathrm{T} = |\mathcal{D}_\mathrm{T}| \cdot |\mathcal{C}|$ document-category pairs and normalizes them to obtain distribution probabilities in $O(N_\mathrm{T} \cdot l_\mathrm{C})$ time; then, to build new bags of words for categories, up to $|\mathcal{D}_\mathrm{T}|$ document bags must be summed up, which is done in $O(|\mathcal{D}_\mathrm{T}| \cdot l_\mathrm{D})$ time. The sum of these two steps, always considering $l_\mathrm{D} < l_\mathrm{C}$, is $O(|\mathcal{D}_\mathrm{T}| \cdot |\mathcal{C}| \cdot l_\mathrm{C})$, which must be multiplied by the final number n_I of iterations.

Summing up, the overall complexity of the method is $O(|\mathcal{D}_\mathrm{S}| \cdot l_\mathrm{D} + n_\mathrm{I} \cdot |\mathcal{D}_\mathrm{T}| \cdot |\mathcal{C}| \cdot l_\mathrm{C})$, which can be simplified to $O(n_\mathrm{I} \cdot |\mathcal{D}| \cdot |\mathcal{C}| \cdot l_\mathrm{C})$, with $l_\mathrm{C} \leq |\mathcal{W}|$. The complexity is therefore linear in the number $|\mathcal{D}|$ of documents, the number $|\mathcal{C}|$ of top categories (usually very small), the mean number l_C of mean terms per category (having $|\mathcal{W}|$ as an upper bound) and the number n_I of iterations in the final phase, which in our experiments is almost always within 20. This complexity is comparable to many other cross-domain classification methods.

3.4.2 Variants

The section above describes the base method which, as will be shown in the experiments, already ensures optimal accuracy with correct setting of parameters. In the following, two variants are proposed to make the method more robust to variation of parameters and to set a tradeoff between accuracy and running times. The two proposed modifications are effective in two different points of the base algorithm and can therefore also be applied at the same time.

Logistic Regression on Similarity

Normally, the cosine similarity between vectors of a document d and a category c directly constitutes the absolute likelihood for them of being related, i.e. of c being the correct label for d. However, in practice, such cosine similarity is generally largely below 1 even if d and c are related and slightly above 0 otherwise. In an ideal situation, the similarity score $s^x(d,c)$ should be very close to 1 if d and c are related, so that also the relative score $p^x(d,c)$ is significantly high.

A solution is to apply a correction function $[0,1] \to [0,1]$ to the "raw" cosine similarity, in order to boost values generally indicating high relatedness to be close to 1. The function to be used must be somehow infered from known data about similarity and relatedness of documents and categories: such data can be obtained from the source domain, whose labeling of documents is known. To extract a function from this data, univariate logistic regression is employed, where a function π with the following form is obtained (Hosmer Jr and Lemeshow, 2004).

$$\pi(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \tag{3.33}$$

In practice, after extracting initial categories representations from the source domain, for each couple $(d,c) \in \mathcal{D}_\mathrm{S} \times \mathcal{C}$, an observation $(x_{d,c}, y_{d,c})$ is extracted.

$$x_{d,c} = \cos(\mathbf{w}_d, \mathbf{w}_c^0) \qquad y_{d,c} = \begin{cases} 1 & \text{if } C_S(d) = c \\ 0 & \text{if } C_S(d) \neq c \end{cases} \tag{3.34}$$

All these observations are used for logistic regression, which computes optimal parameters β_0 and β_1 of function π in order to maximize the following objective function.

$$\prod_{(d,c)\in\mathcal{D}_\mathrm{S}\times\mathcal{C}} \pi(x_{d,c})^{y_{d,c}}(1-\pi(x_{d,c}))^{1-y_{d,c}} \tag{3.35}$$

Following this, the iterative phase is executed as in the base method, but integrating the function π in the calculation of the absolute similarity score between documents and categories. In practice, (3.26) is substituted with the following.

$$s^i(d,c) = \pi(\cos(\mathbf{w}_d, \mathbf{w}_c^i)) \tag{3.36}$$

Regarding the computational complexity to fit the logistic regression model, the cosine similarity for $N_S = |\mathcal{D}_\mathrm{S}| \cdot |\mathcal{C}|$ pairs must be computed to acquire input data, which requires $O(l_c \cdot N_S)$ time; then the model can be fit with one of various optimization methods, which are generally linear in the number N_S of data samples (Minka, 2003).

Relaxed Termination Condition

To terminate the iterative phase before the maximum number N_I of iterations is reached, is normally required that the last computed profile of each category is identical to the one computed in the previous iteration. However, changes of these profiles across later iterations are in practice very small; earlier iterations are by far more important to reach a nearly optimal configuration of the profiles.

Using a substantially low maximum number of iterations could improve the running time of the algorithm with limited losses of classification accuracy, but the number of iterations necessary to obtain a fair accuracy be significantly different according to the specific data under analysis.

The proposed solution is, rather than testing strict equality between current and previous profiles, measuring the mutual cosine similarities and checking that they are all close enough to 1. Formally, fixed a threshold parameter $\beta \in [0, 1]$, the termination condition given in (3.31) is here substituted with the following.

$$\forall c \in \mathcal{C} : \cos(\mathbf{w}_c^{i+1}, \mathbf{w}_c^i) \geq \beta \tag{3.37}$$

This modification ensures a number of iterations not higher than those obtained with the base method, which can be tuned by using different values of β, which should be very close to 1 (such as 0.999). In the experiments section, the effectiveness of this variant on benchmark datasets will be shown.

3.4.3 Experimental Results

Experiments have been conducted to assess the performances of the proposed method, to compare them to the state of the art and to test how they vary for different values of the parameters.

The method has been implemented in Java within a software framework. For logistic regression, we relied upon the WEKA machine learning software (Hall et al., 2009).

Benchmark Datasets

For our experiments, we considered three text collections commonly used in cross-domain classification, allowing to compare our results with those reported by other works for the same collections. Their classes taxonomy exhibits a shallow hierarchical structure, allowing to isolate a small set $\mathcal{C}$ of *top categories*, each including a number of *sub-categories* in which documents are organized. Each input dataset is obtained by picking a small set of top categories of a collection and splitting documents of these categories between a source and a target domain, which contain documents of different branches of the top categories. By labeling each document of the two domains with the correct top category, we obtain suitable benchmark datasets.

The ***20 Newsgroups*** collection[4] (*20NG* for short) is a set of posts from 20 different Usenet discussion groups arranged in a hierarchy, each represented by almost 1,000 posts. We consider the 4 most frequent top categories *comp*, *rec*, *sci* and *talk*, each represented by 4 sub-categories (5 for *comp*). Each test involves two or more top categories, with disjoint source and target domains each composed by documents of 2 or 3 sub-categories for each of them. We performed tests with two, three and all four top categories, considered the sub-categories splits used in (Dai et al., 2007a) and other works for two top categories and those suggested in (Wang et al., 2008a) for less common tests with three or four top categories.

The ***SRAA*** text collection[5] is also drawn from Usenet: it consists of posts from discussion groups about simulated autos, simulated aviation, real autos and real aviation. Tests are performed using two different sets of top categories: {*real*, *simulated*} (with documents about aviation as source and about autos as target) and {*auto*, *aviation*} (simulated as source, real as target). Likely to other works, 4,000 documents are considered for each of the four groups.

[4]`http://qwone.com/~jason/20Newsgroups/` (we used the `bydate` distribution)

[5]`http://people.cs.umass.edu/~mccallum/data/sraa.tar.gz`

The ***Reuters-21578*** collection[6] contains 21,578 newswire stories from year 1992 about economy and finance. Documents are tagged with 5 types of labels, among which *orgs*, *people* and *places* are used as top categories: we considered the three possible pairs of them, using the same split between source and target employed by other works where sub-categories are evenly divided.

Results

In each test run, to evaluate the goodness of the predicted labeling $\hat{C}_T$ with respect to the correct one C_T, likely to other works, we measure the *accuracy* as the ratio of documents in the target domain for which the correct label was predicted: as almost all target domains have evenly distributed documents across categories, this is a fairly valid measure.

$$\text{Accuracy}(C_T, \hat{C}_T) = \frac{|\{d \in \mathcal{D}_\text{T} : \hat{C}_T(d) = C_T(d)\}|}{|\mathcal{D}_\text{T}|} \tag{3.38}$$

Firstly, using the base method, we tested on all the datasets different term weighting schemes and different values of the ρ parameter to check how accuracy varies with them. Plots in Fig. 3.5 show results for these tests on some of the considered test datasets with some of the best performing weighting schemes, namely cosine normalized raw or logarithmic term frequency multiplied by standard or probabilistic idf. The maximum number of iterations (also for subsequent tests) has been set to $N_I = 50$, which has been rarely reached.

From the plots, it can be noticed that optimal accuracy is generally reached with values of ρ near to the minimum possible value, which is $1/|\mathcal{C}|$ (e.g. 0.5 with 2 top categories). This suggests that the selection of representative documents must be large enough in order to obtain good accuracies. By considering all target documents at each iteration to rebulid profiles (i.e. setting $\rho = 1/|\mathcal{C}|$) accuracy is very good, but usually not at its highest possible value.

Given the results, we set $\rho = 1.08/|\mathcal{C}|$ as the default threshold, which is then 0.54, 0.36 and 0.27 for problems with respectively 2, 3 and 4 top-categories: these values are among those performing best over all datasets. Table 3.8 compares results with this setting for a selection of tested term

[6] `http://www.cse.ust.hk/TL/dataset/Reuters.zip`

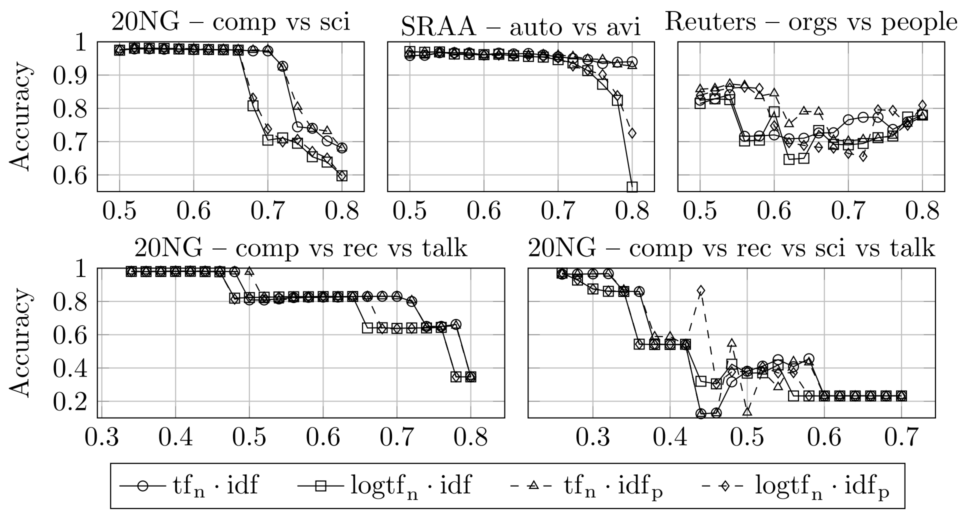


Figure 3.5 – *Classification accuracy (Y axis) of the base cross-domain method for different datasets (plot titles), term weighting schemes (data series, see legend) and values of the threshold parameter ρ (X axis)*

weighting schemes. From now on, given the overall good results it yields, we choose $\mathrm{logtf_n} \cdot \mathrm{idf}$ (cosine normalized logarithmic tf by standard idf) as the default scheme.

Table 3.9 reports the accuracy measures with this configuration for each considered dataset, along with the number of iterations, results reported in other works and also two *baseline* results, which represent expected lower and upper bounds for the accuracy of our method. The "min" accuracy is obtained by classifying target documents directly using the initial category profiles, thus suppressing the iterative phase. The "max" accuracy is instead obtained by classifying target documents using profiles extracted from the target domain itself. Regarding other works, we reported the available results from the following ones: (**CoC**) co-clustering (Dai et al., 2007a), (**TbP**) topic-bridged PLSA (Xue et al., 2008a), (**SC**) spectral classification (Ling et al., 2008a), (**MTr**) matrix trifactorization (Zhuang et al., 2011) and (**TCA**) topic correlation analysis (Li et al., 2012b).

We can see from the table that our approach performs better than reported methods in most cases. Also, the effective accuracy is usually fairly close to the "max" baseline, suggesting that the final category profiles ob-

Table 3.8 – *Comparison of accuracy (A, in thousahdths) and number of iterations (I) with different term weighting schemes, setting* $\rho = 1.08/|\mathcal{C}|$*; best accuracy for each dataset is highlighted in bold*

	$\mathrm{logtf} \cdot \mathrm{idf}$		$\mathrm{logtf} \cdot \mathrm{idf_p}$		$\mathrm{tf_n} \cdot \mathrm{idf}$		$\mathrm{tf_n} \cdot \mathrm{idf_p}$		$\mathrm{logtf_n} \cdot \mathrm{idf}$		$\mathrm{logtf_n} \cdot \mathrm{idf_p}$	
Dataset	A	I	A	I	A	I	A	I	A	I	A	I
20 Newsgroups												
comp vs sci	.981	7	**.981**	7	.979	10	.980	10	.981	8	.980	9
rec vs talk	.993	6	.993	6	.993	5	.994	6	.994	6	**.994**	7
rec vs sci	.980	10	.981	7	.985	8	.985	8	.986	8	**.986**	8
sci vs talk	.973	9	.974	9	**.977**	10	.977	13	.976	11	**.977**	8
comp vs rec	.983	7	**.984**	7	.980	6	.980	7	.983	9	**.984**	7
comp vs talk	.990	6	.991	5	.992	6	**.992**	6	**.992**	6	.992	6
comp v rec v sci	.876	25	.827	48	.932	36	.873	13	**.934**	34	.824	33
rec v sci v talk	.978	11	.978	11	.977	9	.979	12	.979	11	**.980**	11
comp v sci v talk	**.974**	11	.973	11	.964	12	.965	12	.972	18	.972	17
comp v rec v talk	.977	5	.978	6	.980	7	**.980**	7	.979	6	.980	6
comp rec sci talk	.969	8	.969	9	.964	11	.965	12	.970	18	**.970**	17
SRAA												
real v simulated	.940	10	.942	10	.934	10	.937	14	.950	16	**.950**	16
auto v aviation	.963	10	.962	12	**.966**	16	.965	15	.964	14	.965	12
Reuters-21578												
orgs vs places	.729	7	**.730**	6	.722	8	.722	8	.726	7	**.730**	7
orgs vs people	.768	37	.778	40	.841	12	**.873**	9	.826	8	.861	13
people vs places	.513	48	.413	44	**.741**	35	.640	22	.722	23	.652	24

tained by iterative refining are similar enough to the ideal ones which would be extracted directly from the target domain. Within our environment, the running times for single test runs have always ranged between about 10 seconds for tests on Reuters datasets (the smallest ones) and one minute when considering 20 Newsgroups with all four top categories.

We now analyze the effect of the variants, starting from the application of logistic regression. Plots in Fig. 3.6 compare the accuracy obtained either applying or not this variant for some datasets and for different values of the ρ threshold. While with regression the method reaches roughly the same levels of accuracy as does without it, these levels are generally reached for

Table 3.9 – *Results of our method (on rightmost columns) on selected test datasets using* $\text{logtf}_\text{n} \cdot \text{idf}$ *weighting and* $\rho = 1.08/|\mathcal{C}|$*, compared with those reported by other works: the results in bold are the best for each dataset (excluding baselines).*

	Other methods					$\rho = 1.08/\lvert\mathcal{C}\rvert$		Baselines	
Dataset	CoC	TbP	SC	MTr	TCA	Acc.	Iters.	min	max
	20 Newsgroups								
comp vs sci	.870	**.989**	.902	-	.891	.981	8	.803	.986
rec vs talk	.965	.977	.908	*.950*	.962	**.994**	6	.646	.997
rec vs sci	.945	.951	.876	*.955*	.879	**.986**	8	.837	.990
sci vs talk	.946	.962	.956	*.937*	.940	**.976**	11	.781	.989
comp vs rec	.958	.951	.958	-	.940	**.983**	9	.896	.990
comp vs talk	.980	.977	.976	-	.967	**.992**	6	.970	.995
comp v rec v sci	-	-	-	*.932*	-	**.934**	34	.687	.970
rec v sci v talk	-	-	-	*.936*	-	**.979**	11	.503	.988
comp v sci v talk	-	-	-	*.921*	-	**.972**	18	.719	.985
comp v rec v talk	-	-	-	*.955*	-	**.979**	6	.922	.988
comp rec sci talk	-	-	-	-	-	**.970**	18	.618	.981
	SRAA								
real v simulated	.880	.889	.812	-	-	**.950**	16	.618	.969
auto v aviation	.932	.947	.880	-	-	**.964**	14	.807	.979
	Reuters-21578								
orgs vs places	.680	.653	.682	**.768**	.730	.726	7	.732	.915
orgs vs people	.764	.763	.768	.808	.792	**.826**	8	.772	.930
people vs places	**.826**	.805	.798	.690	.626	.722	23	.632	.920

Values of **MTr** for 20NG (in italic) are averages of multiple runs with equal top categories where a baseline classifier trained on source domain and tested on target got 65% or higher accuracy

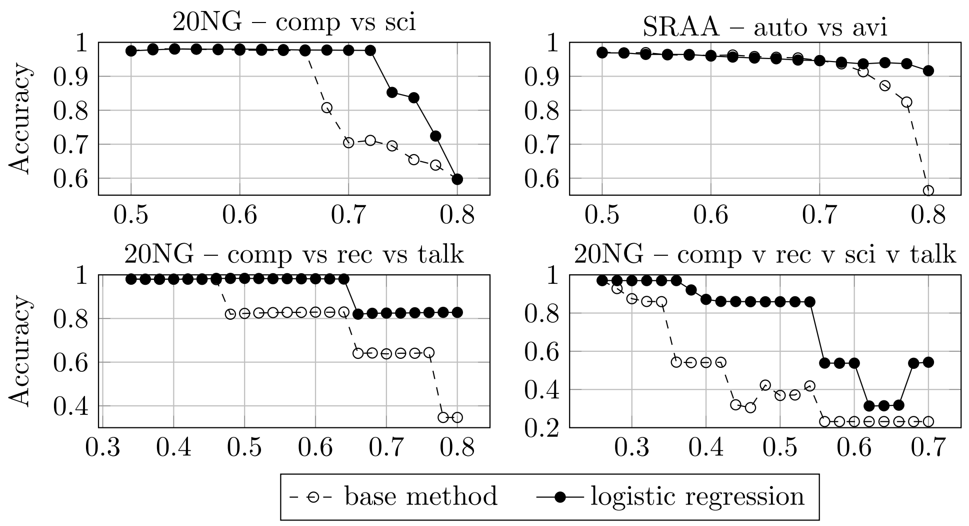


Figure 3.6 – *Comparison of classification accuracy (Y axis) with and without application of logistic regression to similarity on different datasets (plot titles) and for different values of the threshold parameter ρ (X axis)*

a wider range of values of the confidence threshold. We then suppose that the general effect of logistic regression is to make the algorithm more robust with respect to variations of the ρ parameters and then possibly more likely to perform well.

Regarding instead the use of the relaxed termination condition for the iterative phase, Table 3.10 compares the results obtained with the base method (already reported in Table 3.9) with those obtained from this variant with the β parameter set either to 0.9999 or 0.999. It is shown that in the latter cases, the number of iterations is generally sensibly lowered with negligible or acceptable losses of accuracy (with only one exception on "comp vs rec vs sci") or, in some cases, even with slight improvements of it. Interestingly, the final number of iterations with this variant is still different across datasets, but its effects in terms of variation of accuracy is similar in many of them. This suggests that the β parameter could be set to impose a roughly predictable tradeoff between the classification accuracy and the running time, which depends from the number of iterations.

Finally, we show results obtained by simulating the situation where not all documents of the target domain are known in advance. Specifically, we

Table 3.10 – *Accuracy (A, in thousandths) and number of iterations (I) for all dataset with* $\mathrm{logtf_n} \cdot \mathrm{idf}$ *as weighting,* $\rho = 1/|\mathcal{C}|$ *and different settings for termination*

$\beta \rightarrow$	none		0.9999		0.999		$\beta \rightarrow$	none		0.9999		0.999	
Dataset	A	I	A	I	A	I	Dataset	A	I	A	I	A	I
20 Newsgroups – 2 top-categories							20 Newsgroups – 3 and 4 top-categories						
comp vs sci	981	8	980	6	979	4	comp rec sci	934	34	885	11	878	6
rec vs talk	994	6	994	5	994	4	rec sci talk	979	11	979	7	978	6
rec vs sci	986	8	985	5	985	4	comp sci talk	972	18	958	10	955	7
sci vs talk	976	11	977	6	977	5	comp rec talk	979	6	979	4	979	3
comp vs rec	983	9	983	5	983	4	all 4 cats.	970	18	961	11	958	8
comp vs talk	992	6	992	3	992	3	Reuters-21578						
SRAA							orgs places	726	7	726	7	727	5
real vs sim	950	16	950	8	951	5	orgs people	826	8	826	8	830	4
auto vs avi	964	14	966	6	969	4	people places	722	23	731	16	739	12

run tests where only a set ratio of randomly sampled target documents are assumed to be known in the iterative phase and so used to build refined category profiles, but final accuracy evaluation is carried out on all target documents as always. Plots in Fig. 3.7 show the results for some datasets, namely those with the best and the worst accuracy measures among those with 2 or 3 top categories of 20 Newsgroups: results for other datasets of the same groups lie between the two. It is shown that, even if only 10% of the target documents are known (leftmost points in the plots), the obtained accuracy would often be only few percentage points below the one obtained with all documents. This indicates that the method is fairly accurate in classifying even documents of the target domain which are not known while computing category profiles, assuming anyway that a representative enough set of them is known during the iterative phase.

3.4.4 Discussions

This section presented a simple method for cross-domain text categorization based on nearest centroid classification. Profiles for categories are initially

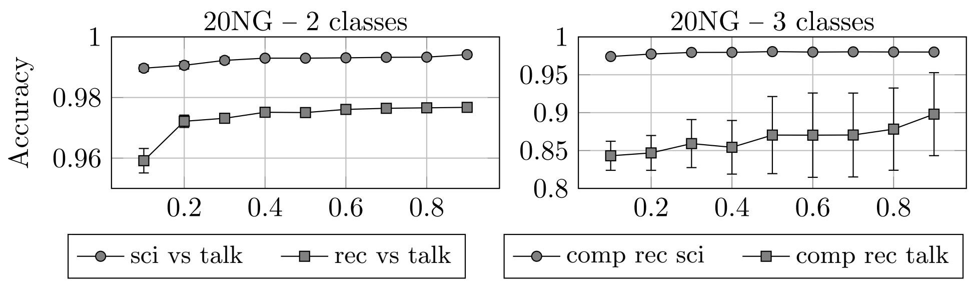


Figure 3.7 – *Classification accuracy (Y axis) of the base cross-domain method for different datasets (data series, see legend) where only a given ratio (X axis) of documents of the target domain is known; each point is an average on 5 runs with different random selections of documents, error bars indicate standard deviation*

built from the source domain, to be then refined by iteratively selecting for each category a set of target documents with high enough classification confidence and using it to build a new profile.

Experiments shown that this method, despite its conceptual simplicity and relative ease of implementation, achieves accuracy results better or comparable to many of those reported throughout the literature for common benchmark datasets. This is obtained with fixed values for the few parameters to be set and with fast running times. By measuring how much category profiles change across successive iterations, it is possible to set a tradeoff between accuracy and running times in a way which guarantees an optimal number of iterations for most datasets.

3.5 Towards Topic-Independent Text Classification: a Novel Semantic Learning Method for Hierarchical Corpora

Most effective approaches in text classification train classifiers from a large number of document terms represented as bags of words, hence their classification efficacy, which inherently depends from terms and topics in the

Training set for classic TC methods

	features (terms)				*label*
doc. id	**"shot"**	**"movie"**	**"song"**	. . .	**category**
doc1	2	1	1	. . .	movies
doc2	0	1	3	. . .	music
doc3	1	2	4	. . .	music
. . .	. . .	. . .	. . .	. . .	. . .

terms are used to predict **categories**

Training set of couples in CHONXI

	features (terms relationships)			*label*
couple	**synonyms**	**hypernyms**	. . .	**relation**
doc1-movies	4	2	. . .	related
doc1-music	1	3	. . .	unrelated
doc2-movies	0	2	. . .	unrelated
. . .	. . .	. . .	. . .	. . .

terms relationships are used to predict **document-category relationships**

Figure 3.8 – *Comparison between explanatory examples of training sets for classic text classification methods (with the VSM) and* CHONXI*: absolute numbers of occurrences of terms in documents and of relationships in document-category couples are used as features, respectively.*

training set, degrades with the increase of previously unknown terms and fails with new topics. This section presents CHONXI (hierar**ch**ical t**o**pic-**in**dependent te**x**t categor**i**zation), a learning method to classify documents in hierarchies of topics capable of dealing even with terms and topics missing in the training set and added to the hierarchy after the training phase. The method learns mutual semantic relationships between document-topic couples from the semantic relationships among their relevant terms, such as counts of WordNet synonymies, antonymes, hypernyms etc. Moreover it learns classification models from a constant number of features derived from the twentynine WordNet semantic relationships, independently from the largeness of the vocabulary and topics of corpora. This kind of features leads to classification models where any couple document-topic is classified as semantically related, *is-a* related or unrelated. This type of classification model, without embedded terms and topics, can potentially classify documents dealing with any new topic, just by providing its relevant terms only in the classification phase: we refer to this classification as *topic-independent*.

3.5.1 Learning Document-Category Relationships from Semantic Terms Relationships

The component discussed here is the core of the CHONXI method and the major novelty with respect to other works.

While many methods work by extracting features (usually words or concepts) from each training document and directly use topics as class labels;

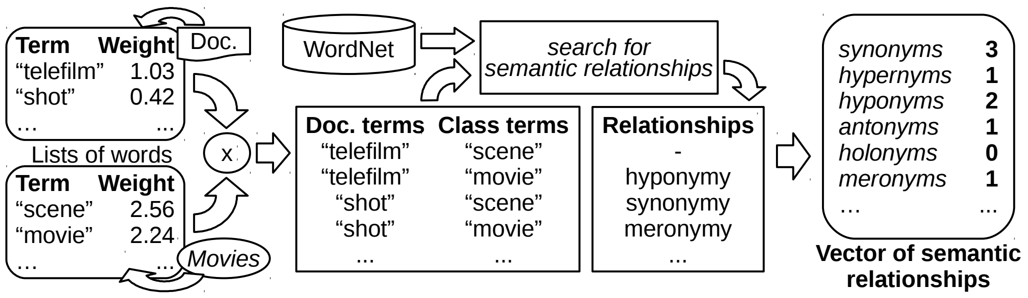


Figure 3.9 – *Illustrative diagram of operations performed by the Semantic Matching Algorithm on a single document-category input couple: for each type of semantic relationship, we only show here the count of its occurrencies.*

Chonxi generates vectors from *document-category couples*, using counts and weights of *semantic relationships* as features and labeling them with the mutual relationship between the document and the category. From a set of training document-category couples, the method learns a knowledge model, which then infers relations between these objects from the semantic relationships between words which compose them.

The resulting model is structurally both *dictionary-independent* and *topic-independent*, as it references neither specific words nor specific categories of documents. We suppose that, if an appropriate training set is used, the resulting model actually presents these traits, making it reusable across different contexts. In a base case, we consider that a document is either *related* to a category if it treats the corresponding topic, or *unrelated* otherwise; we will introduce a further distinction later. This scheme, compared with classic approaches, is sketched in Figure 3.8.

In the rest of this section, we see at an high level how the model is trained, how it is used to classify documents and how its independence from the training set can be exploited.

Pre-processing and Learning

We describe here, at an abstract level, the steps followed to learn the model which infers document-category relationships. These constitute the initial training phase of Chonxi.

As in any text classification problem, is given a collection (or *corpus*) of *training documents* $\mathcal{D}$, labeled in a set of possible categories $\mathcal{C}$. In this section, we do not consider the hierarchical arrangement of categories, which will be recalled in Section 3.5.2, as the considerations below hold regardless of the possible hierarchical structure of the taxonomy.

To start the pre-processing phase, structured representations of both training documents and their topics are needed: each document is reduced to a set of words contained in it along with their frequencies, while each category is represented with a similar set with words taken from all documents belonging to it. Words in each set have associated weights according to their occurrence both in the referred document or category and in the rest of the collection. This representation is based upon the classic bag-of-words approach, with the difference that each set of words is an independent entity: no global set of words (*dictionary*) is considered and no feature selection is performed.

Then, a set $E \subseteq \mathcal{D} \times \mathcal{C}$ of document-category training couples must be prepared, which will determine the training set. In order to build an accurate knowledge model, these couples should be representative of all the possible relationships of interest between documents and categories: in the case of the related/unrelated distinction cited above, a reasonable amount of couples of both types would be needed.

At this point, for each couple $(d, c) \in E$, a so-called *Semantic Matching Algorithm* (hence abbreviated in SMA) finds out the semantic relationships holding between relevant words of d and c. The SMA accepts as input the two sets of words with associated weights representing d and c: for each couple of words obtained from their cross product, by using a source of semantic knowledge like WordNet, the algorithm obtains a set of semantic relationships held between them, such as synonymy, hyponymy, antonymy and so on. Using data obtained from all couples of words, the SMA finally outputs a fixed-size vector of numeric values, indicating how many times each type of semantic relationship has been found and the summed up weights of the involved terms. Figure 3.9 sketches the operation of the SMA through a simplified example using only absolute counts.

Each vector obtained by the SMA is labeled with the effective relationship holding between d and c, such as *related* or *unrelated*: we refer to these labels as *relationship classes*. This set of labeled vectors is finally used as an input to a supervised learning algorithm to learn the knowledge model, which can be used to infer relationship classes for subsequent document-category couples, after preprocessing their representations with the SMA

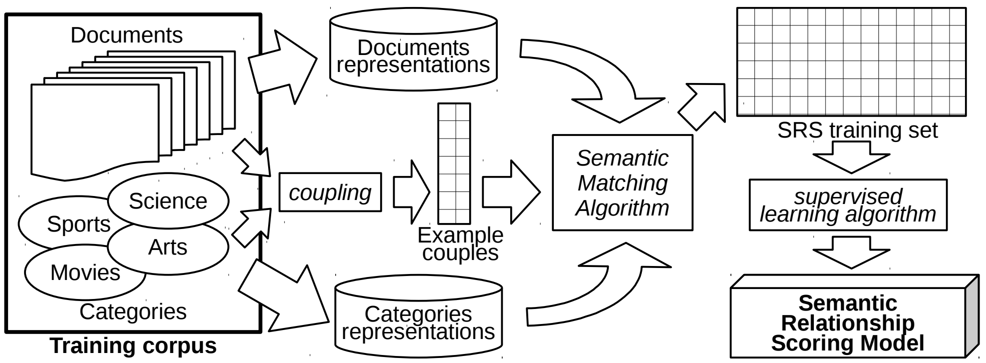

Documents
Science
Sports
Arts
Movies
Categories
Training corpus
Documents representations
coupling
Example couples
Categories representations
Semantic Matching Algorithm
SRS training set
supervised learning algorithm
Semantic Relationship Scoring Model

Figure 3.10 – *Diagram of the Semantic Relationship Scoring Model training process*

as done for training couples.

In the most general case, given a document and a category, a model can predict whether they are *related*, *unrelated* or any other possible relationship class. However, the method uses a *probabilistic* classification model, which gives as output a distribution of probability P between all possible classes, such as "64% related, 36% unrelated". We'll consider the probability $P(\varphi)$ of each possible class φ as its *score*, which denotes the estimated degree at which the corresponding relationship holds between the elements of the analyzed couple. This allows, for example, to consider a category as "more related" than another one to some document.

Having just *related* and *unrelated* as possible labels, the probability for the former given by the model for a document-category couple could be seen as a measure of similarity between the two elements: this would somehow correspond to measure the relatedness between a document and a category prototype in the Rocchio method, for which cosine similarity is usually employed. A key advantage of the proposed model, derived from considering distinct semantic relationships between terms, is that the relatedness between documents and categories is represented through multiple values rather than a single one: this enables it to more precisely characterize the relationships between these objects, rather than simply evaluating relatedness in a linear scale. This is useful to distinguish documents treating a general topic from those addressing a specific branch thereof, as discussed in the next section.

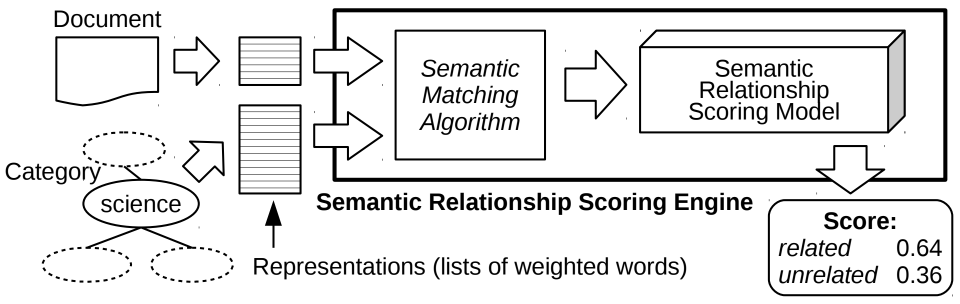

Document
Category
science
Semantic Matching Algorithm
Semantic Relationship Scoring Model
Semantic Relationship Scoring Engine
Representations (lists of weighted words)
Score:
related 0.64
unrelated 0.36

Figure 3.11 – *Diagram summing up how scores for possible relationship classes between a document and a category are estimated by means of the Semantic Relationship Scoring engine.*

Summing up, a probabilistic supervised learning algorithm extracts the model used to output a probability distribution among possible relationship classes for document-category couples: we denote this as the *Semantic Relationship Scoring Model* (hence SRS model). The process described above to produce this model is sketched in Figure 3.10.

Classification

The SMA compares two lists of words to extract a vector of values based on mutual semantic relationships, while the SRS model infers a proabability distribution of relationship classes from these values. Their combinations makes a component able to weight the possible relationships between generic documents and categories given their representations: we denote this component as the *Semantic Relationship Scoring Engine* (hence SRS engine), as illustrated in Figure 3.11.

Chonxi uses the SRS engine as the core component of a higher-level algorithm to classify documents in a hierarchy of categories. As explained later, each input document is compared with multiple categories, to progressively find out the most likely one for it through a top-down search in the hierarchy.

As stated above, the SRS model uses a fixed set of features based on known types of semantic relationships, rather than on specific terms or concepts of the training corpus. For this reason, the engine built upon it may handle representations of any document and any category, even with

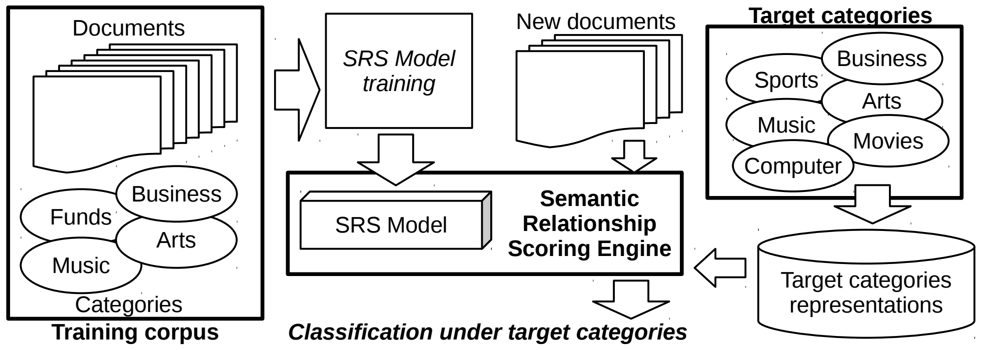


Figure 3.12 – *Diagram showing how a SRS model may be trained from one corpus with a set of topics and used in a SRS engine which analyzes and classifies documents on a different set of* target *categories, given their representations.*

different words or dealing with different topics with respect to those used for training. In our view, this topic-independence aspect enables the user to introduce new categories without retraining the SRS model.

Formally, while the SRS model is trained using documents labeled with topics from a set $\mathcal{C}$, we may use the resulting engine to compare new documents with categories from another generic set $\mathcal{C}^*$, which we denote as the set of *target* categories: this concept is sketched in Figure 3.12. This allows us to perform training only considering a part of the taxonomy of topics of interest ($\mathcal{C} \subset \mathcal{C}^*$) to reduce training times and even to reuse the generated model on a different taxonomy ($\mathcal{C} \cap \mathcal{C}^* = \emptyset$). As the training phase is independent from the target categories, these may be initially unknown and introduced later. In any case, once the model is trained with a sufficient amount of training data, it just needs the representations of all categories in $\mathcal{C}^*$ to classify documents in them.

We'll recall on these possibilities in the next section, after presenting the concrete process for hierarchical classification.

3.5.2 Hierarchical Classification with Chonxi

This section presents the concrete process followed to train and use the core SRS model and engine introduced above to perform automatic classification of documents in a hierarchy of categories. In this context, categories are

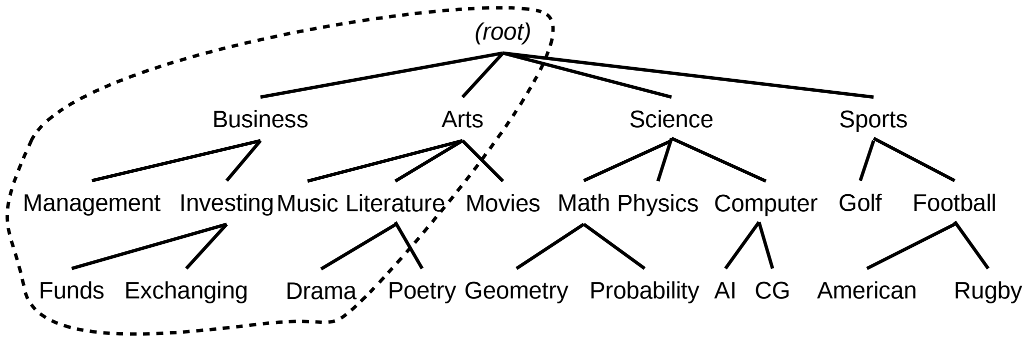


Figure 3.13 – *Excerpt of the DMoz taxonomy of categories. In the proposed method, a knowledge model may be built upon the circled part of topics and then used on the whole taxonomy.*

nodes of a tree-like hierarchical taxonomy: each of them (apart from a single *root* node) treats a specific branch of the topic related to its single *parent* category. Figure 3.13 depicts an example of such a taxonomy.

As above, we consider a set $\mathcal{D}$ of training documents, a set $\mathcal{C}$ of their possible categories and a labeling $L \subseteq \mathcal{D} \times \mathcal{C}$. For compatibility with common target datasets, we refer specifically to *single-label* classification: each document d is labeled with one and only one category, denoted with $l(d)$ ($\forall d \in \mathcal{D} : (d, c) \in L \Leftrightarrow l(d) = c$). We also denote with $L(c) = \{d \in \mathcal{D} : (d, c) \in L\}$ the set of documents belonging to a category c.

We make use of the concept of taxonomy $\mathcal{T} = \langle \mathcal{C}, \prec \rangle$ introduced in Section 3.3.1. We denote with $Children(c)$ the set of categories having c as their parent. A category c with $Children(c) = \emptyset$ is a *leaf* category. We also denote with $SubTree(c) = \{c\} \cup \{x \in \mathcal{C} : x \prec c\}$ the set of all categories having c as ancestor, also including c itself. With $SubTree_n(c) \subseteq SubTree(c)$ we denote instead the set of descendant nodes limited to n lower levels: for example $SubTree_0(c) = \{c\}$ and $SubTree_1(c) = \{c\} \cup Children(c)$.

Documents Representation

First, we illustrate how sets of weighted words representing documents are built. These representations, along with those for categories, are used as an input for the SMA.

For each document d, a typical tokenization and filtering process is first performed to extract single words from it, leaving out other elements like punctuation and HTML tags. Stopwords like articles and prepositions are removed according to a predefined list. Contrarily to other works, no stemming algorithm is applied, as it may alter the exact semantic of some words.

We denote with T_d the set of all distinct words extracted from the text of d and with $n_d(t)$ the number of occurrencies in it of each term $t \in T_d$. We use the common *tf.idf* (Eq. 3.4) to assign a weighy to all terms. We also introduce a *hierarchical* variant of *tfidf*, denoted with $tfidf^H$, following the *flat* idea proposed in Section 3.2.5; in which the *idf* factor is computed considering only documents outside of the category of the current one and of all its descendant categories. In the fraction of this formula, 1 is added to both numerator and denominator to prevent division by zero.

$$tfidf^H_d(t) = tf_d(t) \cdot \log \frac{|\mathcal{D}^d| + 1}{|\{x \in \mathcal{D}^d : t \in T_x\}| + 1}$$

$$\text{where } \mathcal{D}^d = \{x \in \mathcal{D} : l(x) \notin SubTree(l(d))\}$$

The goal of this variant is to not decrease the weight of words contained in documents belonging to the same or more specific categories, so that relevance of terms frequent in these categories but uncommon elsewhere is kept high.

We use $tfidf$ and $tfidf^H$ as two alternative methods to score the weight (or relevance) of terms. In the following, we will refer to the *weight* $w_d(t)$ of a term t in a document d as either one of $tfidf_d(t)$ or $tfidf^H_d(t)$, according to the setup of the process; as a generalization, any other suitable term weighting scheme may be used.

As a last step, the representation of each document d is limited to the k_D terms with the highest weight, or less if the total number is lower than k_D: we denote with $W_d \subseteq T_d$ the final set of selected terms. k_D is a parameter of the process: we will discuss later its influence on computational complexity.

In the end, the representation $R(d) = \langle W_d, tf_d, w_d \rangle$ of a document d is given by a set W_d of up to k_D most relevant terms and associated frequency and weight measures.

Categories Representation

We now discuss how categories, like documents, are represented as sets of weighted words. Generally, we consider for every category $c \in \mathcal{C}$ a set $\mathcal{D}_c \subseteq \mathcal{D}$ of documents related to it: we consider two possible representations.

In the *simple* representation, a category is just represented by the documents labeled with it: $\mathcal{D}_c = L(c)$.

In the *bottom-up* representation, given an integer parameter $z > 0$, each category c is represented by all documents in c or in any of its descendant categories found in the z levels below it.

$$\mathcal{D}_c = \{d \in \mathcal{D} : l(d) \in SubTree_z(c)\}$$

For example, with $z = 1$ for each category are considered only documents in it or in its direct children; while $z = 0$ yields the simple representation. Using words of some lower-level categories aids the top-down classification process in choosing the right path in the tree when classifying documents.

Once the set $\mathcal{D}_c$ has been determined, the initial set of words T_c representing c is simply the union of sets of relevant words from the representations of all these documents.

$$T_c = \bigcup_{d \in \mathcal{D}_c} W_d$$

For each word t, both term frequency *tf* and weight are computed as the average of their respective values across all representations for documents in $\mathcal{D}_c$, considering a value of 0 where t is not present ($\forall t, d : t \notin T_d \Leftrightarrow tf_d(t) = w_d(t) = 0$).

$$tf_c(t) = \frac{1}{|\mathcal{D}_c|} \sum_{d \in \mathcal{D}_c} tf_d(t) \text{ (same for } w_c(t))$$

In the end, as for documents, the number of terms to represent a category c is limited to the k_C with the highest weights: we denote with W_c the set of these terms. k_C is another parameter of the method.

These elements constitute the representation $R_z(c) = \langle W_c, tf_c, w_c \rangle$ of a category c parameterized by the value of z: with $z = 0$ it is a simple representation, with positive values it is a bottom-up representation. We'll

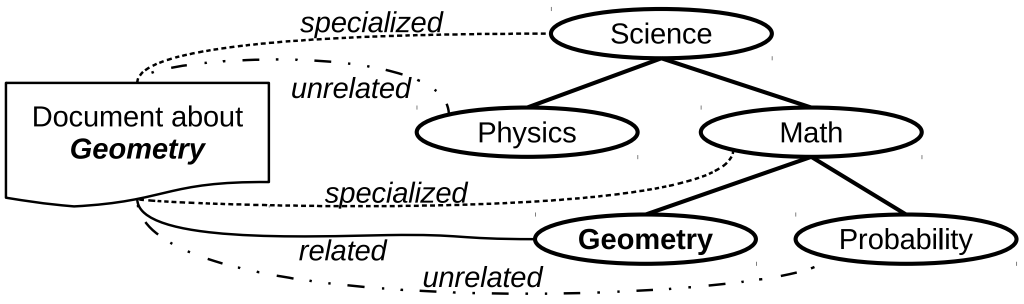


Figure 3.14 – *Coupling of a document with various categories in a hierarchy. The document is* related *to its own category,* specialized *w.r.t. ancestors of its category and* unrelated *to any other category.*

denote with z_C a parameter of the process indicating the standard number of sublevels considered in the bottom-up representation of categories.

Selection and Labeling of Training Couples

To train the SRS model, the process must build a set $E \subseteq \mathcal{D} \times \mathcal{C}$ of example couples and label them with the respective relationship classes. In our process aimed to hierarchical classification, as also illustrated in Figure 3.14, we distinguish three possible labels for each (d, c) couple:

related if d belongs to c $(l(d) = c)$;

specialized if the category of d is a descendant of c $(l(d) \prec c)$;

unrelated if there is no relationship between c and the category of d.

The selection of example couples performed in Chonxi is based on these classes. For each training document $d \in \mathcal{D}$, the method selects:

- one *related* couple $(d, l(d))$ with d and its category;
- n_{spc} random *specialized* couples with d and ancestors of its category;
- n_{unr} random *unrelated* couples with d and categories in different branches of the tree.

A total of $|\mathcal{D}| \cdot (1 + n_{spc} + n_{unr})$ example couples is obtained. n_{spc} and n_{unr} are parameters of the process, which may be tuned to adjust the accuracy of the SRS model and the resources needed to train it.

The Semantic Matching Algorithm

For each couple $(d, c) \in E$, the corresponding representations $R(d)$ and $R_z(c)$ are given in input to the SMA described below.

In the following, we define the *term ranking* $r_x(t)$ of a word t in the representation of either a document or a category x as the number of terms in W_x weighting lower than or equal to t. In other words, the ranking is a decreasing numbering, starting from k_D (in a document) or k_C (in a category), of terms in a set, ordered by descending weight.

$$r_x(t) = |\{y \in W_x : w_x(y) \leq w_x(t)\}|$$

For each single couple of words $(t_d, t_c) \in W_d \times W_c$, a set of possible semantic relationships between them is extracted from the source of semantic knowledge. As a knowledge base for this operation, we choose to use *WordNet* (Miller, 1995), a lexical database of the English language, comprising more than 150,000 terms among nouns, verbs, adjectives and adverbs. WordNet defines more than 100,000 *synsets* ("synonym sets"): each is a set of terms representing a common concept with equal or very close meaning. As more words may represent the same concept (*synonymy*), a single word may represent one of different concepts according to its context (*polysemy*).

In addition to words and synsets, WordNet defines relationships among them. WordNet distinguishes between 28 different types of binary relationships, defined either between whole synsets (*semantic*) or between single words of different synsets (*lexical*). Our method considers all kinds of primitive, direct relationships defined by the WordNet database (without considering derived ones, like *sister terms* with a common hypernym): other than the most commonly used *hypernymy/hyponymy* relationships between general terms and specific concepts (like "fruit" vs. "berry"), less common relationships are also considered, such as *similarity* and *derivation*. We address the interested reader to (Fialho et al., 2011) for details about the different kinds of semantic relationships (also known as *pointers*) defined by WordNet.

To determine how two words t_d and t_c are related, as any word may appear in more than one synset, all relationships between any occurrence of t_d and t_c in WordNet and those between corresponding synsets are considered. As the two terms may appear in a same synset or even be equal, an explicit *synonymy relationship* is introduced between such terms, increasing the number of different relationship types to 29. We denote with $\mathcal{R}$ the set of these types.

Once the SMA has found relations for every couple of terms $(t_d, t_c) \in W_d \times W_c$, it considers, for each type $\rho \in \mathcal{R}$ of semantic relationship, the set $C(\rho) \subseteq W_d \times W_c$ of word couples among these which are related by ρ. Using this information, along with the weights of terms in both d and c, for each relation type $\rho \in \mathcal{R}$ it computes the following four *relationship metrics*:

- the ratio between the number of couples related by ρ and the total number of couples;

$$relRatio_\rho = \frac{|C(\rho)|}{|W_d \times W_c|}$$

- the sum of the terms frequencies of words from related couples;

$$tfSum_\rho = \sum_{(t_d,t_c)\in C(\rho)} tf_d(t_d) + tf_c(t_c)$$

- the sum of rankings for words from related couples;

$$rankSum_\rho = \sum_{(t_d,t_c)\in C(\rho)} r_d(t_d) + r_c(t_c)$$

- the sum of the weights of words from related couples.

$$wSum_\rho = \sum_{(t_d,t_c)\in C(\rho)} w_d(t_d) + w_c(t_c)$$

The final vector given as output by the SMA is composed of the values of these metrics for each relation type, with added as a further feature the sum of ratios for all types.

$$totalRelRatio = \sum_{\rho\in\mathcal{R}} relRatio_\rho$$

Note that, beyond the simplified example in Figure 3.8, four features are computed for each relation type rather than one. Still, as discussed above, the number of features does not depend on the training set, but only on the constant number of known relationship types: specifically, considering 29 relationship types, $4 \cdot 29 + 1 = 117$ features are extracted from each couple. Other than being fixed, this number has the advantage of being significantly smaller than the features count in methods learning directly from terms, where thousands or more of them are generally needed to get an adequately accurate model.

Hierarchical Classification Process

The execution of the SMA on all document-category couples in E generates a set of vectors labeled as either *related*, *specialized* or *unrelated*, which constitute the training set to build the SRS model. At this point a suitable learning algorithm is used to train a probabilistic model, which will be used in the SRS engine to give, for any couple of a document and a category, the probabilities $P(R)$, $P(S)$ and $P(U)$ for them to be respectively related, specialized and unrelated ($P(R) + P(S) + P(U) = 1$).

We describe here the algorithm which uses the SRS engine to classify documents in a given hierarchical taxonomy $\mathcal{T}^* = \langle \mathcal{C}^*, \prec^* \rangle$, possibly different from the training taxonomy $\mathcal{T}$. For each generic test document d, the algorithm either predicts a single category $c \in \mathcal{T}^*$ or outputs *unclassified* if no suitable enough category can be found. Reflecting the characteristics of the test collections used in the experiments, we assume that a document may be labeled with any category, either a leaf or an internal node of the hierarchy.

Essentially, the algorithm is a top-down search which starts from the single root node c_R of the hierarchy and walks the tree down to the predicted category. For each explored node, the algoritm decides whether to stop and return the current category as the predicted topic or to continue the descent by picking the best child node to explore.

As a first step, the process builds the representation of the test document d, consisting of the set W_d of words in it and of their respective weights (term frequency and $tfidf$). After that, the recursive procedure described below is started from the root node c_R. Figure 3.15 shows the pseudo-code for both the initial step and the recursive procedure.

Being c_{cur} the currently explored category node, the SRS engine is first used to determine the probability for d of being related to c_{cur} considering its simple representation: this constitutes the *score* for c_{cur}. The SRS engine is then used to compare d to each direct child category of c_{cur}, represented

```
function HClassify($d$ (test document), $\mathcal{T}^*$)
    $rDoc \leftarrow$ DocumentRepr($d$, $k_D$)
    $c_R \leftarrow$ Root($\mathcal{T}^*$)        ▷ start iterating from the root of the hierarchy
    return Iterate($rDoc$, $c_R$)
end function
function Iterate($rDoc$, $c_{cur}$)
    if $Children(c_{cur}) = \emptyset$ then
        return $c_{cur}$                          ▷ stop if leaf node reached
    end if
    $rCur \leftarrow$ CategorySimpleRepr($c_{cur}$, $k_C$)
    $P_{cur} \leftarrow$ SRSEngine($rDoc$, $rCur$)
    $bestChild \leftarrow \perp$
    $bestScore \leftarrow 0$
    for all $c \in Children(c_{cur})$ do
        $rChild \leftarrow$ CategoryBottomUpRepr($c$, $k_C$, $z_C$)
        $P_c \leftarrow$ SRSEngine($rDoc$, $rChild$)
        if $P_c(R) + P_c(S) > bestScore$ then
            $bestChild \leftarrow c$
            $bestScore \leftarrow P_c(R) + P_c(S)$
        end if
        if $P_{cur}(R) > bestScore$ then
            return $c_{cur}$
        else if $bestScore \geq \tau$ then
            return Iterate($rDoc$, $bestChild$)
        else
            return *unclassified*
        end if
    end for
end function
```

Figure 3.15 – *Algorithm for hierarchical classification of documents.*

with the bottom-up representation on a fixed number z_C of sublevels: each $c \in Children(c_{cur})$ is scored with the combined probability for the couple (d, c) of being related or specialized.

After these scores are computed, if c_{cur} has the highest one, then the recursive process terminates labeling d with c_{cur} as the predicted category;

otherwise, if the highest score reaches a predefined threshold τ, the procedure continues by running the algorithm on the corresponding child. If no child obtains a score of at least τ, the process terminates with *unclassified* as output, indicating that a wrong path has been presumably followed, as no one of the examined categories seems somehow related to the current document.

To reduce the odds of ending in a wrong path, we use a generalization of this algorithm: we consider a set of current nodes bounded to a fixed size $n_{cc} \geq 1$, rather than a single one (with $n_{cc} = 1$ we obtain the algorithm described above). The algorithm starts from the set containing only the root node c_R. In every step, scores are computed as above for all current categories (probability of being related) and for all their respective children (probability of being related or specialized). If one of the current categories outscores both other ones and all children, then it is returned as the prediction, otherwise the modified algorithm recurses on the set of up to n_{cc} of all the analyzed children having highest scores, ignoring those below the τ threshold. Should be noted that, while exploring more than one path, this improved algorithm keeps going down of one level in the hierarchy at each step with no backtracking: its complexity is therefore linear in the maximum number n_{cc} of nodes visited at each step.

Dealing with Previously Unknown Categories

We seen that the SRS engine used in the classification algorithm works on general representations of documents and categories. Given its potential topic-independence, we may even consider topics which are outside of the hierarchy used while training the SRS model. As discussed above, this allows us to classify documents in a taxonomy $\mathcal{T}^*$ of target categories $\mathcal{C}^*$ different from the training categories $\mathcal{C}$ and even to alter this taxonomy at run-time.

Firstly, in the pre-processing phase, as depicted in Figure 3.13, we may select a subset $\mathcal{C} \subseteq \mathcal{C}^*$ of categories of the taxonomy and use only those, along with documents therein, to train the SRS model. For this, we still need representations for all categories of the taxonomy where to classify new documents, but the training process requires less computational resources.

More generally, the set of categories $\mathcal{C}$ used to train the SRS model does not need to be a subset of the target categories $\mathcal{C}^*$ and may even be disjoint from it. As a practical example, the SRS model can be built using any large enough collection of training documents and then used to classify documents in multiple completely different taxonomies, after trivially compiling representations of respective categories.

As discussed, in a real setting, we may even be interested in altering the set of target categories after the SRS engine is built. In practice, to add a new topic at run-time, we need to compile its representation and to add it to the taxonomy explored by the top-down algorithm. Similarly, we may remove categories and update representations of already existing ones.

A representation $B_z(c)$ for a previously unseen category c (or an updated representation for a known category), composed of a set W_c of relevant words and of their weights, may be built from a set of documents related to it, using the process described in Section 3.5.2. The accuracy of this representation generally improves as the number, the quality and the relatedness of documents used to create it grows: anyway, labeled documents are not always available in large quantity and CHONXI should thus be able to work with a fair accuracy even when a limited number of representative documents is available for each topic.

In the experiments section, other than results obtained with traditional setups, we'll see how the accuracy of classification changes as differently sized parts of taxonomies are used to train a model for the SRS engine applied to whole ones. To assess the generality of the SRS engine, we'll also present the results of some *cross-dataset* experiments, where the SRS model is trained using a dataset and tested on a different one. We will also show results of tests where representations for all categories are built upon not more than 20 representative documents for each: this demonstrates that, each time a new topic is introduced, is usually sufficient to gather such a number of representative documents to reach reported accuracies.

Time Computational Complexity

We analyze the computational complexity in terms of time of CHONXI, in pre-processing, learning and classification phases, showing the impact of the various parameters of the process. In the following, we denote with δ the depth of the categories hierarchy and with γ the average number of children per non-leaf node.

For a comparison, in a classical approach to hierarchical classification, we should consider the base complexity of the used learning algorithm and deduce from it the cost of training a classifier for each internal node of the hierarchy. Any learning algorithm has a complexity which is function of the number $|\mathcal{D}|$ of data vectors (documents), η of features (words) and $|\mathcal{C}|$ of classes (categories). For instance, fast classifiers such as Naïve Bayes have a training complexity $O(|\mathcal{D}| \cdot \eta \cdot |\mathcal{C}|)$, so the total time complexity for training in a hierarchy is $O(\delta \cdot |\mathcal{D}| \cdot \gamma \cdot \eta)$. We address the reader to (Ceci and Malerba, 2007) for a more extended analysis.

For the initial training in Chonxi, we must take into account the preprocessing phase where the training set is built and the learning of the SRS model from it: these are one-time operations, even if new categories are added.

For each document-category couple, a number of terms comparisons up to $W_d \cdot W_c$ is performed. Each training document in $\mathcal{D}$ creates a number of instances equal to the couplings, which are 1 related, n_{unr} unrelated and up to n_{spc} specialized. The number of instances $|E| = |\mathcal{D}| \cdot (1 + n_{spc} + n_{unr})$ in the training model for each document is linear in the number of training documents, so the complexity of the preprocessing phase is $O(|\mathcal{D}| \cdot (1 + n_{spc} + n_{unr}) \cdot W_d \cdot W_c) = O(|E| \cdot W_d \cdot W_c)$.

As this training set has a fixed and relatively small number of both features (semantic relationships between terms) and classes (possible document-category relationships), the complexity of training the SRS model just depends on the number $|E|$ of example couples: many learning algorithms such as SVMs achieve linear complexity in $|E|$.

To classify a document, it is necessary to couple its W_d representative terms with the W_c terms of each category to be compared through the SRS engine. This comparison must be done at each iteration of the top-down algorithm with all children of the current node, or of n_{cc} nodes in the extended algorithm. Considering the worst case in which all the documents are classified under the leaves of the hierarchy, the complexity is $O(\delta \cdot n_{cc} \cdot \gamma \cdot W_d \cdot W_c)$.

When a new category are added, all is needed is to create its representation by averaging those for n_R given documents, which requires $O(n_R \cdot W_d)$ time.

3.5.3 Experimental Results

We propose Chonxi as a new classification model with the ambition of performing a good hierarchical classification and also of being flexible enough to provide good performance with previously unknown topics. We designed and performed a set of experiments to answer the following questions.

- Is CHONXI able to do good hierarchical document classification, w.r.t. the state of the art?
- What is the influence of the various parameters involved in the process?
- To which extent CHONXI can classify documents with categories/subtrees of the hierarchy which were unknown in the training phase?
- Can we use a SRS model trained on a dataset to classify documents of another dataset?
- How much does the number of terms and relations found on Wordnet affect the effectiveness of the classification?
- Can CHONXI recognize topics represented by a low number of documents?
- Which of the semantic relationships recognized by WordNet are the most useful in determining document-category relationships?

Datasets

In order to evaluate CHONXI, we picked four different existing data sources and also created a fifth one, larger than the others. The five sources differ considerably in the number of documents and categories and for the breadth and depth of the hierarchy tree, as summarized in Table 3.11. A brief description of the collections is reported hereafter.

Yahoo!_C.M. , provided[7] and used by (Ceci and Malerba, 2007), was extracted from *Yahoo! Search Directory*: it contains the 907 web documents indexed in the first three levels of the "Science" category. Blank and script-only documents have been removed. The final dataset consists of 901 documents grouped in 69 categories, organized in a 4-level taxonomy.

[7]Both *Yahoo!_C.M.* and *DMoz_C.M.* are available at `http://www.di.uniba.it/~malerba/software/webclass/WebClassIII.htm`.

Table 3.11 – *Distribution of categories and documents in the used datasets.*

	Level	root	2	3	4	5	6	7	8	9
Yahoo!_C.M.	$\|\mathcal{C}\|$	1	6	27	35					
(4 levels)	$\|\mathcal{D}\|$	0	98	349	454					
DMoz_C.M.	$\|\mathcal{C}\|$	1	21	81	85	32	2			
(6 levels)	$\|\mathcal{D}\|$	0	350	1,563	2,703	1,163	57			
Ohsumed	$\|\mathcal{C}\|$	1	22	49	17	6				
(5 levels)	$\|\mathcal{D}\|$	760	4,115	9,822	1,291	83				
20Newsgroups	$\|\mathcal{C}\|$	1	6	9	13					
(4 levels)	$\|\mathcal{D}\|$	0	1,771	8,871	8,186					
DMoz_new	$\|\mathcal{C}\|$	1	5	112	605	811	601	176	47	8
(9 levels)	$\|\mathcal{D}\|$	202	372	3,776	22,475	22,814	14,311	3,730	1,027	198

DMoz_C.M. , again provided and used by (Ceci and Malerba, 2007), covers a portion of the DMoz web directory. The dataset contains documents found in the first five levels of the subtree rooted in the "Conditions and Diseases" node of the "Health" branch of DMoz: it consists of 222 categories organized in a 6-level taxonomy containing 5,836 documents.

Ohsumed is a subset of the MEDLINE database created by Hersh and colleagues at the Oregon Health Sciences University and consists of 348,566 documents obtained from 270 medical journals over the period 1987 to 1991. The documents were manually indexed using the taxonomy MeSH (Medical Subject Headings), which consists of 18,000 categories. The choice used in the literature is the consideration of only the articles in the "HealthDiseases" category and its subtree in the MeSH taxonomy. This sub-hierarchy has 119 categories, of which only 102 contain usable data. The initial construction of this collection provides a multi-label classification for each document; in order to use this dataset, we eliminated all the documents classified with more than one category, getting a total of 16,071 documents.

20Newsgroups is a collection of 18,828 newsgroup documents partitioned across 20 groups. This dataset was originally created as a flat set of categories, and has often been used for flat text classification. Since some of the newsgroups are very closely related to each other (like "comp.sys.ibm.pc.hardware" / "comp.sys.mac.hardware"), while others are highly unrelated (like "misc.forsale" / "soc.religion.christian"), these 20 topics can be arranged according to the subject matter in a 4-levels hierarchy of 29 nodes.

DMoz_new is a new dataset[8] extracted from DMoz, more extensive than *DMoz_C.M.* in terms of documents and topics, created in order to better test the specific features of Chonxi with portions of datasets of non-negligible size. This dataset consists of 68,905 documents organized in 2,366 categories. The taxonomy has a maximum depth of 9 levels and 5 top-level categories: "Arts", "Business", "Recreation", "Shopping", "Sports".

Can be noted that these datasets have marked differences in the topics they cover. The first three dataset are more domain-specific: *Yahoo!_C.M.* is about science, *DMoz_C.M.* and *Ohsumed* about health. *20Newsgroups* has a limited number of categories, but spans over multiple, heterogeneous interest areas (computers, politics, science, etc.). *DMoz_new* has a far greater number of categories within five branches with some different degrees of mutual relatedness: for example, "Sports" can be considered as loosely related to "Recreation", but highly unrelated to "Arts".

Evaluation Measures

We evaluate the effectiveness of Chonxi with more than one measure. The first one is the standard accuracy (Acc) defined in machine learning, that is the ratio of documents correctly classified over all testing documents, whose set is here denoted with $\mathcal{D}_E$. The second is the universally used F_β-measure (Rijsbergen, 1979), the harmonic mean of recall (r) and precision (p):

$$F_\beta = \frac{(\beta^2 + 1) \cdot p \cdot r}{\beta^2 \cdot p + r}$$

The β parameter is the relative weight of precision and recall. For ease of comparison with the papers from which the test datasets are picked, we use the same weight, thus $\beta = 1$, and in the following we will refer to the measure as F_1. Since we have to evaluate the classification into a hierarchy

[8]The dataset is available for download at `http://tinyurl.com/chonxi-dmoz-new`.

of topics, we use a hierarchical variant of precision and recall (Silla and Freitas, 2011). The definition of the hierarchical F_1-measure is as follows:

$$hF_1 = \frac{2 \cdot hP \cdot hR}{hP + hR}$$

$$\text{where } hP = \frac{\sum_{d \in \mathcal{D}_E} |\hat{P}_d \bigcap \hat{T}_d|}{\sum_{d \in \mathcal{D}_E} |\hat{P}_d|}, \; hR = \frac{\sum_{d \in \mathcal{D}_E} |\hat{P}_d \bigcap \hat{T}_d|}{\sum_{d \in \mathcal{D}_E} |\hat{T}_d|}$$

where hP and hR are the hierarchical precision and the hierarchical recall. Given a test document $d \in \mathcal{D}_E$, $\hat{P}_d$ is the set consisting of the specific category predicted for d and all its ancestor classes and $\hat{T}_d$ is the set consisting of the true most specific category of d and all its ancestor categories. This variant of precision and recall is used in order to adjust the weight of errors according to the distance between the predicted and the correct category.

Setup of Experiments

Considering the large amount of experiments and the time required, we use the basic *hold-out* method: for each category, 2/3 of documents are used as training set and 1/3 as test set. Normally, training documents are used both to represent respective categories and to form example couples feeding the training set of the SRS model; test documents, as usual, are instead used for posterior evaluation. To make results comparable against each other, training and test sets of each dataset are always the same across all experiments.

To train the SRS model, we use the *random forest* learner (Breiman, 2001) as implemented in the Weka machine learning software (Hall et al., 2009) with all parameters set to their defaults.

Here are the default values of all parameters of CHONXI, used unless otherwise noted. We set $tfidf^H$ as term weighting scheme, considering $z_C = 2$ sublevels for the bottom-up categories representations. Each document is represented by $k_D = 10$ terms and each category by a number dependent on the ratio between the number of categories and documents in each dataset: $k_C = 100$ in *Yahoo!_C.M.*, $k_C = 500$ in *DMoz_C.M.* and *DMoz_new*, $k_C = 1000$ in *Ohsumed* and *20Newsgroups*. Finally, the number n_{cc} of explored nodes during classification is set to 5. The effects of varying some of these parameters are investigated.

Table 3.12 – *Hierarchical text classification best results obtained by* CHONXI *and by related works.*

		Chonxi	**Ceci (Ceci and Malerba, 2007)**	**Li (Li et al., 2012c)**	**Ruiz**
Yahoo!_C.M.	**Acc**	**0.839**	≈ 0.62	-	
	hF$_1$	0.879	-	-	
DMoz_C.M.	**Acc**	**0.635**	≈ 0.5	-	
	hF$_1$	0.686	-	-	
Ohsumed	**Acc**	0.405	-	-	
	hF$_1$	0.511	-	$\approx$ **0.63**	
20Newsgroups	**Acc**	0.568	-	-	
	hF$_1$	**0.658**	-	≈ 0.58	
DMoz_new	**Acc**	0.454	-	-	
	hF$_1$	0.566	-	-	

Classification Experiments with Known Topics

We first investigate the effectiveness of CHONXI in a classic text classification setting, where labeled documents are available in advance for all categories: here we compare our results with those reported from other works and test the effects of altering some parameters.

We first compare results obtained with default parameters with the best results reported by other works: for this purpose, we specifically picked those performing hierarchical single-label classification on the same datasets. The results on *Yahoo!_C.M.* and *DMoz_C.M.* are compared with those obtained by (Ceci and Malerba, 2007): CHONXI obtains better accuracy on both datasets. Regarding *Ohsumed*, there are multiple works of comparison (Cai and Hofmann, 2003; Li et al., 2012c; Ruiz and Srinivasan, 2002). Our results are better than some of them, however, we have a negative result compared to (Li et al., 2012c). We suspect that this happens mostly because documents from *Ohsumed* mostly contain very specific medical terms, while WordNet is a general purpose lexical database, so the relationships between terms within Wordnet for this dataset are very few, as will be shown in Section 3.5.3. This penalizes our model, which is built specifically on the relations between words recognized by the underlying knowledge base. In fact, to strengthen this argument, we can compare our method with the same work also on *20Newsgroups*: in this dataset, CHONXI

Table 3.13 – *Results varying the feature selection utilized for the term selection.*

	$tfidf$ **(standard)**		$tfidf^H$	
dataset	**Acc**	$\mathbf{hF_1}$	**Acc**	$\mathbf{hF_1}$
Yahoo!_C.M.	0.862	0.888	0.839	0.879
DMoz_C.M.	0.636	0.683	0.635	0.686
Ohsumed	0.296	0.348	0.405	0.511
20Newsgroups	0.564	0.652	0.568	0.658
DMoz_new	0.462	0.579	0.454	0.566

obtained an higher hF_1 measure.

We now show the effectiveness of classification by varying some parameters of the framework. Table 3.13 compares the results reported above with the case where the standard $tfidf$ is used in place of the $tfidf^H$ we proposed for term weigthing in Section 3.5.2. While in most cases there is no significant difference between the results, we observe that $tfidf^H$ guarantees an important accuracy improvement on the *Ohsumed* dataset.

Figure 3.16 shows the study on the behavior of the classifier varying the number z_C of sublevels used for the *bottom-up* representation of each category, discussed in Section 3.5.2: if 0 a category is represented only by words in strictly related documents, if 1 also documents in direct children categories are considered and so on. We observe that the trend saturates after a few sublevels. This means that our representation does not require looking deeply down the hierarchy: considering only 2 or 3 sublevels generally gives good results even for deeper hierarchies. We point out that, even considering many sublevels, the SMA only considers the k_C most important terms of each category representation for comparison, so the choice of z_C has limited influence on the time required to build the SRS model.

Topic Independence-Based Experiments

Up to now, we considered a standard setting where all topics are known at training time. In the following tests instead, a SRS model is trained on a set of categories and then applied to either a larger set or a disjoint one: these experiment are performed to evaulaute the topic-independence aspect

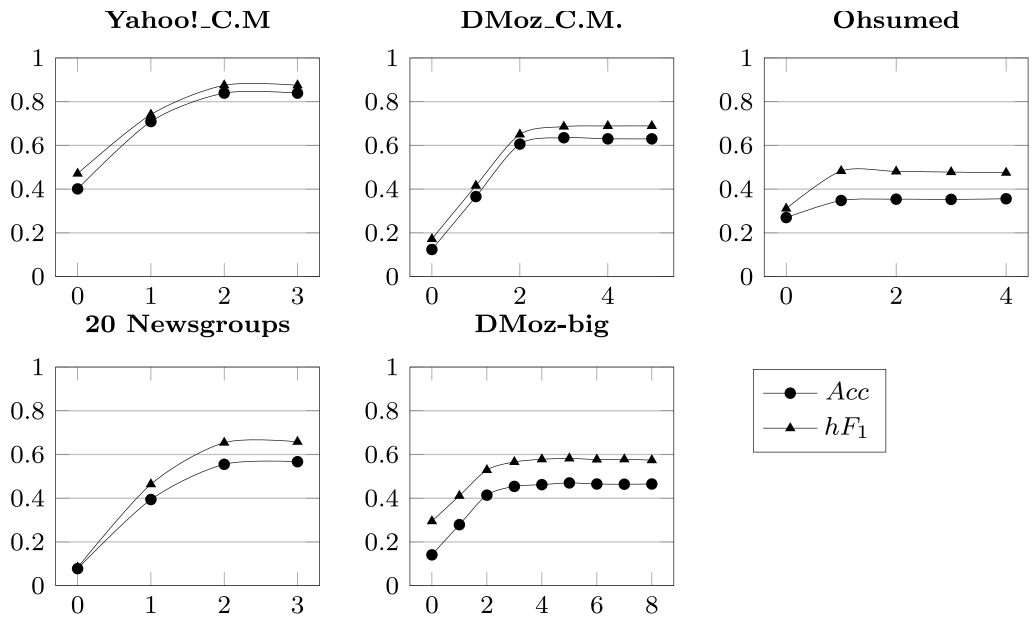


Figure 3.16 – *Results varying the number of sublevels used for the* bottom-up *categories representation.*

of the method, the expected capability to seamlessly add new topics in the classification phase without building a new model.

We first investigate the effectiveness of Chonxi when the training set only contains a part of the categories where documents must be classified. We envisage that, to reduce the time required to initially build the SRS model, it can be trained on a reduced portion of the dataset. This is also a first simulation of the case where some target categories are unknown during the training phase and introduced later. For each dataset, we run tests where a given percentage of all categories is randomly selected and the training phase is run considering only these categories and documents therein; then, representations for remaining categories are built and, as in previous experiments, evaluation is performed on test documents of all categories. The choice of the training categories in the different simulations is not random, but weighted in order to have a training sub-hierarchy that contains the right amount of specialized and unrelated categories among them.

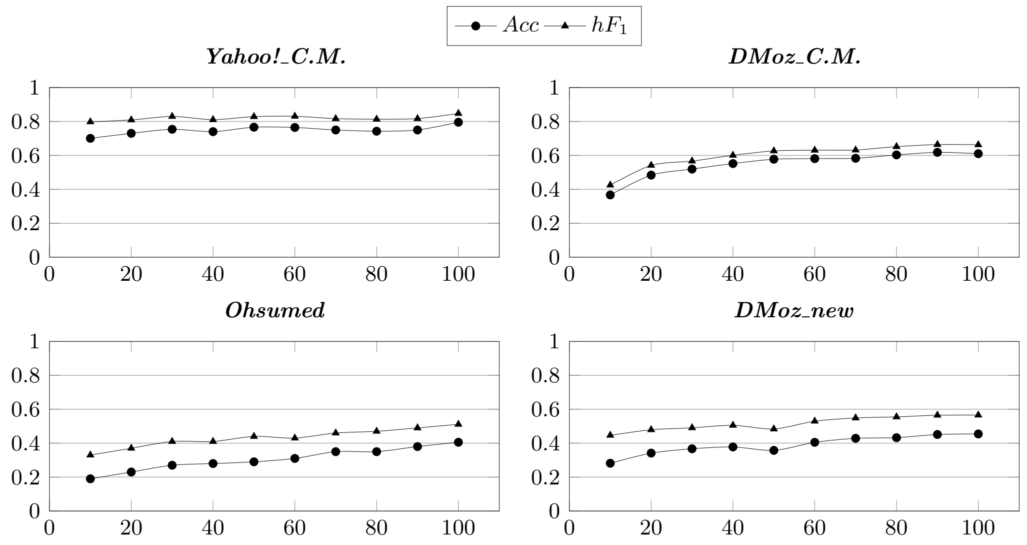


Figure 3.17 – *Results varying the percentage of categories used in the training set (on X axis).*

Figure 3.17 shows the accuracy and the hF_1 obtained creating the model with various percentages of training categories, with increments of 10%. The tests for each percentage were repeated with 5 different configurations of training categories. This test has not been done on the *20Newsgroups* dataset, since its taxonomy is too small to consider only a part.

The tests with 100% of training categories exactly correspond to the standard setting considered previously, where all topics are known at training time, which obviously is the most favorable one. Nonetheless, the results show how the use of a small portion of the topics hierarchy is enough to guarantee an effective classification, and also that the removal of a few categories (results on high percentages in the figure) does not affect much the effectiveness of the classification. This may indicate that the classification using semantic relationships between couples of terms can be used to handle huge amounts of topics, but at the same time only a portion of the categories is necessary to build the SRS model. In addition, the same model can also be used in contexts where a dynamic change of the topics into which documents are classified is expected.

To test at a larger extent the topic-independence aspect of CHONXI which allows it to classify documents under previously unseen topics, we performed *cross-dataset* experiments: the SRS model trained on the topics (and related documents) of a dataset is used to classify documents of a different dataset, with topics completely unknown in the learning phase. Specifically, we used the SRS model trained on *DMoz_new* to classify documents of the other datasets; this can be done despite, as reported in Section 3.5.3, *20Newsgroups* has a different coverage of topics and the other datasets treat in more detail specific domains (science and health) which are not even treated in *DMoz_new*. Comparing these results with the best ones obtained in the standard classification, where training and test sets are parts of the same dataset (Table 3.12), the loss of effectiveness is very low (on *Yahoo!_C.M.* the result is even better), but we have the advantage of having built and trained only a single SRS model and applied it to multiple datasets just extracting relevant categories representations, which guarantees shorter setup times and is generally not possible with standard document classification algorithms.

Table 3.14 – *Results of cross-dataset classification, with the SRS model trained on* DMoz_new *and tested on different datasets.*

dataset	**Acc**	**hF$_1$**
Yahoo!_C.M.	0.956	0.978
DMoz_C.M.	0.589	0.633
Ohsumed	0.397	0.529
20Newsgroups	0.489	0.6

In the following, we specifically simulate a realistic scenario where the SRS model is initially trained on some known categories and then used in a context where new topics are progressively introduced, likely for example to classification of news stories, where new facts and events frequently happen over time. In each of the two performed simulations, each related to one dataset, we define an initial part of the taxonomy, upon which the SRS model is trained, then across multiple steps the tree of known categories is progressively enriched: after each step, a classification evaluation is performed on test documents of all categories. In practice, at each step we partition the taxonomy in three groups of categories: *training* categories used to initially build the SRS model (fixed throughout the simulation), *added* categories (growing at each step) and *unknown* categories (shrinking at each step). We expect that, at each evaluation, test documents of known topics (both training and added) are correctly classified within them, while documents of unknown topics should be labeled as *unclassified*, reflecting the fact that no suitable category is present in the taxonomy at that time. We stop the simulation before including all branches of the taxonomy in order to have some unclassifiable documents at every step, as likely in a real situation in which there never is a complete picture of all possible topics.

Considering that the introduction of new categories requires to compile their representations from pertinent documents, as the user should be relieved as most as possible from manual classification, we set a limit on the number of documents supposed to be available for each topic: each category representation is then built on the basis of at most 20 random documents, for both training and added categories. To make these results comparable with those given above, the training-test split is not changed: this implies that some training document are discarded, while test documents remain exactly the same.

To simulate a progressively growing taxonomy of categories, we split the whole one into multiple parts and define an order in which these parts are added, with the first one being used to train the SRS model. We decided to split the taxonomy of each dataset according to the respective top-level categories, those which are direct children of the root node: the initial tree is made of one or more of the available top-level categories along with their whole subtrees, while in each step another one more branch is added. Considering this, we choosed to perform this type of test on *Yahoo!_C.M.* and *DMoz_new*, because these are the datasets whose top-level categories are least similar to each other, so that new categories deal with topics which can be very different from those used for training. Table 3.15 shows in detail on which training top-level categories the SRS models for the two datasets are built (step 0) and in which order further top categories are added in subsequent steps (all more specific topics within the subtree of each top category are always included implicitely): *Yahoo!_C.M.* deals with some different branches of the "Science" general category, while *DMoz_new* covers a wider spectrum of more diversified topics.

In order to separately assess the effectiveness of CHONXI in correctly labeling documents of training and added categories and leaving documents of unknown categories out of the taxonomy, we measure three distinct accuracy metrics. Two hF_1 measures limited to documents of respectively

Table 3.15 – *Order in which top-level categories and respective subcategories (branches) are added to the taxonomy in the simulations*

		Categories count	
Step	**Added branches**	**known**	**unknown**
Yahoo!_C.M. (37 training categories, 70 total)			
0 (training)	Biology, Chemistry and Earth Sciences	37	33
1	+ Agricolture	50	20
2	+ Alternative	64	6
left out	(Mathematics)		
DMoz_new (543 training categories, 2,366 total)			
0 (training)	Recreation	543	1,823
1	+ Arts	974	1,392
2	+ Sports	1,178	1,188
3	+ Shopping	1,498	868
left out	(Business)		

training and added categories are computed. Additionally, we report the *unclassification accuracy* as the ratio of documents of unknown categories which are correctly labeled as *unclassified* w.r.t. their total number.

In Figure 3.18 the progression across simulation steps of these metrics is reported for the two used datasets. As the taxonomy grows throughout subsequent steps, a decrease of the accuracy is expected at some extent: due to the increasing number of known categories, the probabilities of error increase for documents of both training and added categories and also the erroneous classification of a document of an unknown class in one of the known topics becomes more likely. Anyway, the plots show that this natural decay of accuracy is generally limited to few percentage points for all the three measures. The drop on the hF_1 for training categories is very restrained, being about 1.5% and 6% across the whole simulations on *Yahoo!_C.M.* and *DMoz_new* respectively. For what concerns added categories, the begin-to-end variation is slightly negative on *Yahoo!_C.M.* (less than 1% is lost) and positive (about 2%) on *DMoz_new*. The unclassification accuracy generally has more relevant variations, but never drops below 60%. Overall, we note that the reported hF_1 measures are very close to

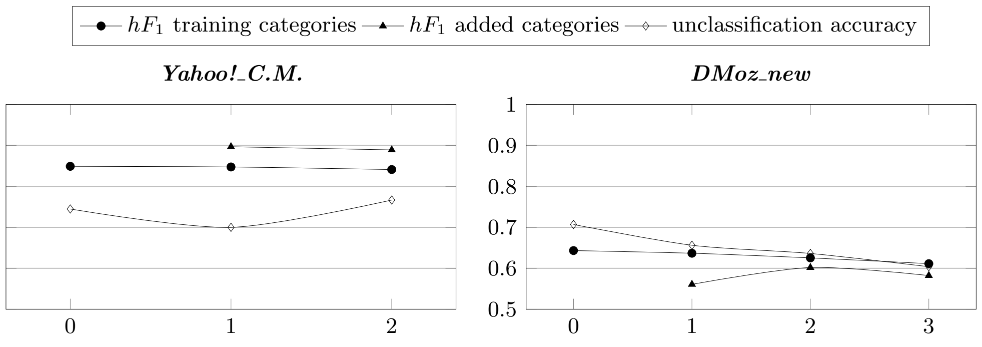


Figure 3.18 – *Accuracy measures on the simulations of* Chonxi *where new topics are progressively added (hF_1 for added categories at step 0 is missing because only training categories are known).*

those reported in previous experiments for the same datasets, despite the imposed limit of 20 documents to represent each category.

Relationships and Features Analysis

The key aspect of the proposed method is the supposed correlation between the mutual relationship between a document and a category and the semantic relationships between terms contained in them, which then become the predictive features used to label the type and degree of relatedness within couples. Table 3.16 shows, for each semantic type of couple document-category, the average percentages of the couples of terms with at least one semantic relationship between them. This statistic shows that in *Ohsumed* the couples of related terms are much less than in the other datasets, to the point that there is no siginficant difference in the amount of relationships found in related, specialized and unrelated couples. This, in our opinion, explains why Chonxi produces the worst results with this dataset.

As cited above, using WordNet, the method is able to distinguish between 29 distinct types of semantic relationships. In the following, we discuss which of these relationship types are most important in determining actual relationships between elements.

The learning algorithm we used, *random forest*, builds models in form of sets of decision trees, which indicate the relationship class of any input vector by walking across branches to a leaf node indicating the prediction,

Table 3.16 – *Percentages of couples of terms with at least one semantic relationship in WordNet.*

dataset	**related**	**specialized**	**unrelated**
Yahoo!_C.M.	11.33%	1.98%	0.76%
DMoz_C.M.	3.05%	1.16%	0.79%
Ohsumed	0.58%	0.41%	0.40%
20Newsgroups	1.18%	#np	0.50%
DMoz_new	3.15%	0.78%	0.48%

according to specific values of the vector. An example of such a decision tree obtained in one of the models generated in the experiments is reported in Figure 3.19, pruned to the higher-level nodes for readability (a complete tree contains hundreds or thousands of nodes): to make a prediction for a document-category couple, we start from the root and follow branches of the tree according to the computed relationship metrics. This example shows the topic-independence aspect of the model: neither words nor topics are referenced, contrarily to knowledge models directly learned from bags of words.

While this is only a part of one model, we analyzed the trees obtained from all experiments to understand which features prove to be more informative. In the following, we will use the name of each type of semantic relationship (*synonymy*, *hyponymy*, etc.) to refer to the group of four features derived from it.

For each of the five benchmark datasets, once having built the training set for the SRS model, we measured their correlation to the relationship class of couples (related, specialized, unrelated) according to the chi-squared statistic, always relying on the Weka implementation: in this way, for each dataset, we obtain a ranking of the features according to how much useful they are to predict the document-class relationships.

While the exact ranking of features is different for each dataset, we obtained important indications which are valid for all of them. Namely, most importantly, there are three (groups of) features which always are by far the most discriminative ones, although with differing orders: *rates-Sum*, *synonymy* and *derivationally related form*. The former two are quite expected to be important to distinguish related elements, while form deriva-

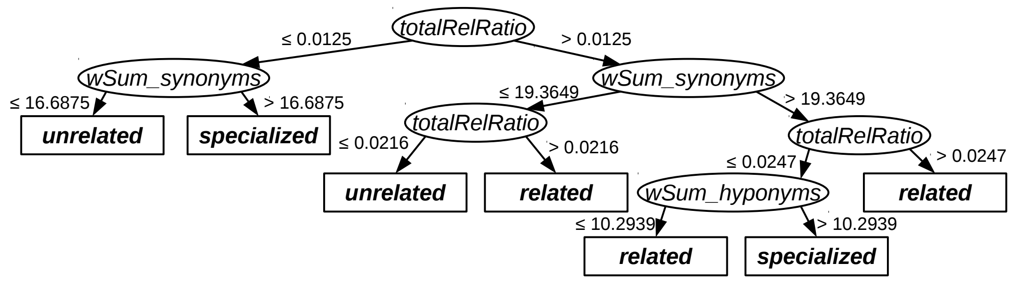


Figure 3.19 – *Example of decision tree generated from the experiments, pruned to higher-level nodes. Starting from the root, branches are chosen according to the value of the specified feature. Leafs (rectangles) indicate the relationship class for the majority of training instances.*

tion, which links nouns and verbs with common stems, is justified as is by far the most common lexical (between single words) relationship in WordNet. Other relationships with an average high ranking after these three are *hypernym* and *hyponym*, which are the most common relationships between whole synsets; also present are two features nearly indicating synonymy: *similar to* (adjectives with similar meaning but not exact synonyms) and *verb group* (verbs with a common more general meaning). Summing up, the most informative semantic knowledge is either exact or near synonymy (including exact matching) and hypernymy/hyponymy relationships, other than the overall presence of semantic relationships.

While this information tells which types of relationships (and therefore features of the SRS model) are most discriminative to determine the document-category relationships, may be interesting to understand more thoroughly how the two are correlated; in other words, which features explicitely support and which negate that documents and categories are potentially related and/or have different detail levels.

Likely to the ranking of features, the trees generated across multiple experiments are different, but we observed recurring patterns across all of them. A rather predictable recurrence is that unrelated couples are generally found in branches where feature values, especially synonymy and other features with high ranking in the chi-squared tests cited above, are below certain thresholds, indicating that a little number of semantic relationships denotes with high confidence a lack of relatedness between compared objects. On the other hand, distinguishing between related and specialized couples is more tricky (model accuracy measures on these two classes are generally lower than those for unrelatedness), but representative features can be spotted for them as well. Features like synonymy and form derivation are often slightly higher for similar couples, but also specialized ones bear a notable amount of occurrencies of these relationships. Instead, interestingly, specialized couples are notably more tied to the hypernymy and hyponymy relationhsips: these couples, infact, can often be distinguished from others (unrelated and even related) from the high incidence of these kinds of semantic links. This recurrence somehow reflects the effective type of relationship between documents and categories in couples labeled as specialized: as one treats a specific branch of the broader topic covered by the other, we visualize that corresponding *is-a* relationships hold between respective representative terms. The example tree in Figure 3.19 follows some of these observations.

3.5.4 Discussions

This section proposes Chonxi, a new learning method for hierarchical text classification based on infering the relatedness between documents and topic categories from the semantic relationships between respective representative words. Chonxi extracts and uses knowledge supposed to be independent from the specific domain of the training set, thus allowing the later introduction of further topic categories at negligible computational cost, just providing a limited number of example documents.

Chonxi works by training a Semantic Relationship Scoring model, a probabilistic knowledge model used to compare documents and categories that is independent on the document words because it employs, as features, quantitative information about WordNet semantic relationships among relevant terms rather than the words or concepts themselves, as commonly adopted used in the literature.This model has a fixed number of features, generally much smaller with respect to traditional learning methods, and, provided that an adequately large training set is given, we expect it to be *topic-independent*, and therefore valid even for topics missing in the training phase.

With extensive experiments on well known hierarchical text datasets, we demonstrated that our classification method using the SRS model in a top-down algorithm is highly effective, as it outperforms the state-of-the-art hierarchical classification methods on three out of four datasets. Our worst result, which is lower with respect to just one work, is found testing a dataset of the medical field, containing a large number of terms not covered by WordNet, the lexical database we used.

We have also shown how Chonxi can obtain interesting results over the entire test set even if a small portion of topics has been used to train the SRS model. This allows to use our method in situations in which performing training on all the taxonomy is too expensive, such as with big data settings and large number of topics, or where the expected taxonomy evolves dynamically and retraining the model at every change would be too expensive. We also shown that it is possible to use a limited number of documents to represent each topic and that an SRS model trained on one dataset can be readily reused as is on other ones where representations of categories are available.

3.6 Sentiment Analysis

Sentiment analysis (or *opinion mining*) refers to the extraction from a collection of text of the general attitude expressed regarding some particular object (Pang and Lee, 2008; Liu, 2012). This branch of text analysis includes some different cases. In a conceptually simple case, sentiment analysis means classifying a set of text documents (e.g. reviews of a movie) as expressing a *positive* or *negative* (or maybe *neutral*) opinion. To perform a finer distinction, documents may be labeled with a rating, implying different possible levels of positive or negative polarity, as expressed for example with ratings “from one to five stars”. To extract even more useful information, the analysis may identify different aspects of the same object and evaluate the average polarity for each of them (e.g. reviews of a movie may declare that it has an *interesting* plot, but that it is *poorly* directed). Another possibility is to understand whether a polarity is present at all, distinguishing subjective from objective texts.

Sentiment analysis methods can be in some cases considered as specific applications of other known task: classifying reviews as either positive or negative can be seen for example as a text categorization problem. Anyway, specific techniques are often employed when dealing with opinions, for example the extraction of composite phrases rather than single words and the use of external knowledge indicating the polarity of each of them.

3.7 Markov Chain Based Method for In-Domain and Cross-Domain Sentiment Classification

As previously seen, cross-domain text classification techniques are proposed in literature. An interesting area of application of this task, and on which current research is focused, is the sentiment analysis. Through cross-domain methods, it is potentially possible to perform this classification on reviews of objects of one type by training a classifier on reviews of different objects: it is possible for example to classify comments about books by training on reviews of movies which are already labeled for their polarity (e.g. with a 1-to-5-stars rating), or similarly to distinguish positive and negative reviews of hotels by learning from those addressed to restaurants. As for the other cases, cross-domain learning works better if the two domains are enough similar and there are many common words between the two.

Language heterogeneity between source and target domain is the trickiest issue in cross-domain setting so that a preliminary transfer learning phase is generally required. The best performing techniques addressing this point are generally complex and require onerous parameter tuning each time a new source-target couple is involved. This section introduces a simpler method, presented in (Domeniconi et al., 2015b), based on the Markov chain theory to accomplish both transfer learning and sentiment classification tasks. In fact, this straightforward technique requires a lower parameter calibration effort. Experiments on popular text sets show that our approach achieves performance comparable with other works.

3.7.1 Method

Markov chain is a mathematical model that is subject to transitions from one state to another in a states space $\mathcal{S}$. In particular, it is a stochastic process characterised by the so called *Markov property*, namely, future state only depends on current state, whereas it is independent from past states.

Before talking about our method in detail, notice that the entire algorithm can be split into three main stages, namely, the text pre-processing phase, the learning phase and the classification phase. We argue that the learning phase and the classification phase are the most innovative parts of the whole algorithm, because they accomplish both transfer learning and sentiment classification by means of only one abstraction, that is, the Markov chain.

Text Pre-Processing Phase

The first stage of the algorithm is text pre-processing. Starting from a corpus of documents written in natural language, the goal is to transform them in a more manageable, structured format.

Initially, standard techniques are applied to the plain text, such as word tokenisation, punctuation removal, number removal, case folding, stopwords removal and the Porter stemming algorithm (Porter, 1980). Notice that stemming helps the sentiment classification process, because words having the same morphological root are likely to be semantically similar.

The representation used for documents is the common bag-of-words, using as initial weight, for each term t^d, the *relative frequency* $rf(t, d)$, computed as follows:

$$\mathbf{w}_t^d = rf(t, d) = \frac{f(t, d)}{\sum\limits_{\tau \in \mathcal{T}} f(\tau, d)} \tag{3.39}$$

After having built the bags of words, a feature selection process is performed to limit the curse of dimensionality. Feature selection is an essential task, which could considerably affect the effectiveness of the whole classification method. Thus, it is critical to pick out a feature set as good as possible to support the classification process. Here, we try to tackle this issue by following some different feature selection approaches. Below, we briefly describe these methods, while the performed experiments where they are compared are presented in the next section.

Feature Selection Variants

The first feature selection method we use for testing is based on *document frequency*, defined above in 3.2. Each term having a *df* higher than an established threshold is selected. The second alternative consists in selecting only the terms included in a list (i.e. opinion wordlist) of commonly used words in expressing sentiment, regardless of their frequency, their document frequency or other aspects. These words simply are general terms that are known to have a certain polarity. The opinion wordlist used is that proposed in (Liu, 2012), containing 2003 positive words and 4780 negative words.

The previously presented feature selection methods are both unsupervised. On the contrary, we present straight away another viable option to perform the same task exploiting the knowledge of class labels. First of all, we use a supervised scoring function to find the most relevant features with respect to their ability to characterize a certain category. This function is either *information gain IG* or *chi-square* χ^2, defined as in Section 3.2.4. The ranking obtained as output is used on the one hand to select the best n features and on the other hand to change term weighting inside documents, using the combination of this global factor with the local relative frequency of a term in a document.

Learning Phase

The learning phase is the second stage of our algorithm. As in any categorisation problem, the primary goal is to learn a model from a training set, so that a test set can be accordingly classified. Though, the mechanism should also allow transfer learning in case of cross-domain setting.

The basic idea consists in modelling term co-occurrences: the more words co-occur in documents the more their connection should be stronger. We could represent this scenario as a graph whose nodes represent words and whose edges represent the strength of the connections between them. Considering a document corpus $\mathcal{D} = \{d_1, d_2, ..., d_N\}$ and a dictionary $\mathcal{T} = \{t_1, t_2, ..., t_k\}$, $A = \{a_{ij}\}$ is the set of connection weights between the term t_i and the term t_j and each a_{ij} can be computed as follows:

$$a_{ij} = a_{ji} = \sum_{d=1}^{N} \mathbf{w}_{t_i}^d \cdot \mathbf{w}_{t_j}^d \qquad (3.40)$$

The same strategy could be followed to find the polarity of a certain word, unless having an external knowledge base which states that a word is intrinsically positive, negative or neutral. Co-occurrences between words and classes are modelled for each document whose polarity is given. Again, a graph whose nodes are either terms or classes and whose edges represent the strength of the connections between them is suitable to represent this relationship. In particular, given that $\mathcal{C} = \{c_1, c_2, ..., c_M\}$ is the set of categories and $B = \{b_{ij}\}$ is the set of edges between a term t_i and a class c_j, the strength of the relationship between a term t_i and a class c_j is augmented if t_i occurs in documents belonging to the set $D^j = \{d \in \mathcal{D} : c_d = c_j\}$.

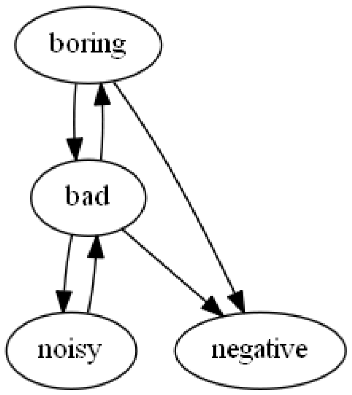


Figure 3.20 – *Transfer learning from a book specific term like boring to an electrical appliance specific term like noisy through a common term like bad.*

$$b_{ij} = \sum_{d \in D^j} \mathbf{w}_{t_i}^d \tag{3.41}$$

Careful readers may have noticed that the graph representing both term co-occurrences and term-class co-occurrences can be easily interpreted as a Markov chain. In fact, graph vertices are simply mapped to Markov chain nodes and graph edges are split into two directed edges (i.e. the edge linking states t_i and t_j is split into one directed edge from t_i to t_j and another directed edge from t_j to t_i). Moreover, for each state a normalisation step of all outgoing arcs is enough to satisfy the probability unitarity property. Finally, Markov property surely holds because each state only depends on directly linked states, since we evaluate co-occurrences considering just two terms (or a term and a class) at a time.

Now that we have introduced the basic idea behind our method, we explain how the learning phase is performed. We rely on the assumption that there exist a subset of common terms between source and target domain that can act as a bridge between domain specific terms, allowing and supporting transfer learning. So, these common terms are the key to let information about classes flow from source specific terms to target specific terms, exploiting term co-occurrences, as shown in Figure 3.20.

We would like to point out that the just described transfer learning process is not an additional step to be added in cross-domain problems; on

Table 3.17 – *This table shows the structure of MCTM. It is composed of four submatrices, representing transition probability that, starting from a current state (i.e. row), a future state (i.e. column) is reached. Both current states and future states can be either terms or classes.*

	$\mathbf{t_1}, ..., \mathbf{t_k}$	$\mathbf{c_1}, ..., \mathbf{c_M}$
$\mathbf{t_1}, ..., \mathbf{t_k}$	A'	B'
$\mathbf{c_1}, ..., \mathbf{c_M}$	E	F

the contrary, it is implicit in the Markov chain mechanism and, as such, it is performed in in-domain problems as well. Obviously, if both training set and test set are extracted from the same domain, it is likely that most of the terms in test set documents already have a polarity.

Apart from transfer learning, the Markov chain we propose also fulfils the primary goal of the learning phase, that is, to build a model that can be subsequently used in the classification phase. Markov chain can be represented as a transition matrix (MCTM), composed of four logically distinct submatrices, as shown in Table 3.17. It is a $(k+M)\times(k+M)$ matrix, having current states as rows and future states as columns. Each entry represents a transition probability, which is computed differently depending on the type of current and future states (term or class), as described below.

Let $\mathcal{D}_{Tr}$ and $\mathcal{D}_{Te}$ be the subsets of document corpus $\mathcal{D}$ chosen as training set and test set respectively. The set A, whose each entry is defined by equation 3.40, is rewritten as

$$a_{ij} = a_{ji} = \begin{cases} 0, & i = j \\ \sum\limits_{d \in \mathcal{D}_{Tr} \cup \mathcal{D}_{Te}} \mathbf{w}_{t_i}^d \cdot \mathbf{w}_{t_j}^d, & i \neq j \end{cases} \tag{3.42}$$

and the set B, whose each entry is defined by equation 3.41, is rewritten as

$$b_{ij} = \sum_{d \in \mathcal{D}_{Tr}^j} \mathbf{w}_{t_i}^d \tag{3.43}$$

where $\mathcal{D}_{Tr}^j = \{d \in \mathcal{D}_{Tr} : c_d = c_j\}$. The submatrices A' and B' are the normalised forms of equations 3.42 and 3.43, computed so that each row of the Markov chain satisfies the probability unitarity property. Instead, each entry of the submatrices E and F looks like as follows:

$$e_{ij} = 0 \tag{3.44}$$

$$f_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{3.45}$$

Notice that E and F deal with the assumption that classes are absorbing states, which can never be left once reached.

Classification Phase

The last step of the algorithm is the classification phase. The aim is classifying test set documents by using the model learnt in the previous step. According to the bag-of-words representation, a document $d_t \in \mathcal{D}_{Te}$ to be classified can be expressed as follows:

$$d_t = (\mathbf{w}_{t_1}^{d_t}, ..., \mathbf{w}_{t_k}^{d_t}, c_1, ..., c_M) \tag{3.46}$$

$\mathbf{w}_{t_1}^{d_t}, ..., \mathbf{w}_{t_k}^{d_t}$ is the probability distribution representing the initial state of the Markov chain transition matrix, whereas $c_1, ..., c_M$ are trivially set to 0. We initially hypothesize to be in many different states (i.e. every state t_i so that $\mathbf{w}_{t_i}^{d_t} > 0$) at the same time. Then, simulating a single step inside the Markov chain transition matrix, we obtain a posterior probability distribution not only over terms, but also over classes. In such a way, estimating the posterior probability that d_t belongs to a certain class c_i, we could assign to d_t the most likely label $c_i \in \mathcal{C}$. The posterior probability distribution after one step in the transition matrix, starting from document d_t, is:

$$d_t^* = (\mathbf{w}_{t_1}^{d_t^*}, ..., \mathbf{w}_{t_k}^{d_t^*}, c_1^*, ..., c_M^*) = d_t \times MCTM \tag{3.47}$$

where d_t is a column vector having size $(k+M)$ and $MCTM$ is the Markov chain transition matrix, whose size is $(k+M) \times (k+M)$. At this point, the category that will be assigned to d_t is computed as follows:

$$c_{d_t} = \underset{i \in C^*}{\operatorname{argmax}}\, c_i^* \tag{3.48}$$

where $C^* = \{c_1^*, ..., c_M^*\}$ is the posterior probability distribution over classes.

Computational Complexity

The computational complexity of our method is the time required to perform both the learning phase and the classification phase. Regarding the learning phase, the computational complexity overlaps with the time needed to build the Markov chain transition matrix, say $time(MCTM)$, which is

$$time(MCTM) = time(A) + time(B) + time(A^{'} + B^{'}) + time(E) + time(F)$$

Remember that A and B are the submatrices representing the state transitions having a term as current state. Similarly, E and F are the submatrices representing the state transitions having a class as current state. $time(A^{'} + B^{'})$ is the temporal length of the normalisation step, necessary in order to observe the probability unitarity property. On the other hand, E and F are simply a null and an identity matrix, requiring no computation. Thus, since time complexity depends on these factors, all should be estimated.

The only assumption we can do is that in general $|\mathcal{T}| >> |\mathcal{C}|$. The time needed to compute A is $O(\frac{|\mathcal{T}|^2}{2} \cdot (|\mathcal{D}_{Tr}| + |\mathcal{D}_{Te}|))$, which in turn is equal to $O(|\mathcal{T}|^2 \cdot (|\mathcal{D}_{Tr}| + |\mathcal{D}_{Te}|))$. Regarding transitions from terms to classes, building the submatrix B requires $O(|\mathcal{T}| \cdot |\mathcal{C}| \cdot |\mathcal{D}_{Tr}|)$ time. In sentiment classification problems we could also assume that $|\mathcal{D}| >> |\mathcal{C}|$ and, as a consequence, the previous time becomes $O(|\mathcal{T}| \cdot |\mathcal{D}_{Tr}|)$. The normalisation step, which has to be computed one time only for both A and B, is $O(|\mathcal{T}| \cdot (|\mathcal{T}| + |\mathcal{C}|) + |\mathcal{T}| + |\mathcal{C}|) = O((|\mathcal{T}| + 1) \cdot (|\mathcal{T}| + |\mathcal{C}|))$, which can be written as $O(|\mathcal{T}|^2)$ given that $|\mathcal{T}| >> |\mathcal{C}|$. Further, building the submatrix E requires $O(|\mathcal{T}|^2)$ time, whereas for submatrix F $O(|\mathcal{T}| \cdot |\mathcal{C}|)$ time is needed, which again can be written as $O(|\mathcal{T}|)$ given that $|\mathcal{T}| >> |\mathcal{C}|$. Therefore, the overall complexity of the learning phase is

$$time(MCTM) \simeq time(A) = O(|\mathcal{T}|^2 \cdot (|\mathcal{D}_{Tr}| + |\mathcal{D}_{Te}|))$$

In the classification phase, two operations are performed for each document to be categorised: the matrix product in equation 3.47, which requires $time(MatProd)$, and the maximum computation in equation 3.48, which requires $time(Max)$. Hence, as we can see below

$$time(CLASS) = time(MatProd) + time(Max)$$

the classification phase requires a time that depends on the previous mentioned factors. The matrix product can be computed in $O((|\mathcal{T}|+|\mathcal{C}|)^2 \cdot |\mathcal{D}_{Te}|)$ time, which can be written as $O(|\mathcal{T}|^2 \cdot |\mathcal{D}_{Te}|)$ given that $|\mathcal{T}| >> |\mathcal{C}|$. On the other hand, the maximum requires $O(|\mathcal{C}| \cdot |\mathcal{D}_{Te}|)$ time. Since the assumption that $|\mathcal{T}| >> |\mathcal{C}|$ still holds, the complexity of the classification phase can be approximated by the calculus of the matrix product.

Lastly, the overall complexity of our algorithm, say $time(Algorithm)$, is as follows:

$$\begin{aligned} time(Algorithm) &= time(MCTM) + time(CLASS) \simeq \\ &\simeq time(MCTM) = O(|\mathcal{T}|^2 \cdot (|\mathcal{D}_{Tr}| + |\mathcal{D}_{Te}|)) \end{aligned}$$

This complexity is comparable to those we have estimated for the other methods, which are compared in the upcoming experiments section.

3.7.2 Experimental Results

The Markov chain based method has been implemented in a framework entirely written in Java. Algorithm performance has been evaluated through the comparison with *Spectral feature alignment* (*SFA*) by Pan et al. (Pan et al., 2010) and *Joint sentiment-topic model with polarity-bearing topics* (*PBT*) by He et al. (He et al., 2011), which, to the best of our knowledge, currently are the two best performing approaches.

We used a common benchmark dataset to be able to compare results, namely, a collection of Amazon[9] reviews about four domains: Book (B), DVD (D), Electronics (E) and Kitchen appliances (K). Each domain contains 1000 positive and 1000 negative reviews written in English. The text pre-processing phase described in 3.7.1 is applied to convert plain text into the bag-of-words representation. Then, before the learning and classification phases, we perform feature selection in accordance with one of the

[9]www.amazon.com

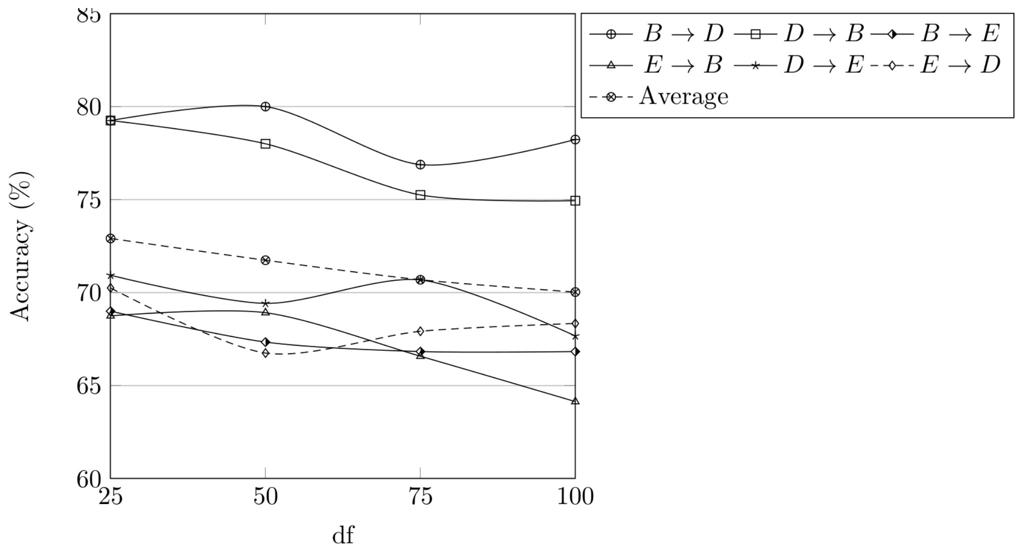


Figure 3.21 – *Cross-domain classification by varying the minimum df to be used for feature selection.*

alternatives presented previously. The goodness of results is measured by accuracy, averaged over 10 different source-target splits.

Performances with every feature selection method are shown below and compared with the state of the art. Differently, the Kitchen domain is excluded from both graphics and tables due to space reasons. Anyway, accuracy values are aligned with the others, so that considerations to be done do not change. The approaches we compare only differ in the applied feature selection method: document frequency, opinion wordlist and supervised feature selection technique with term ranking in the Markov chain.

In the first experiment we perform, as shown in Figure 3.21, the only parameter to be set is the minimum document frequency *df* that a term must have in order to be selected, varied from 25 to 100 with step length 25. In other words, setting the minimum *df* equal to n means that terms occurring in less than n documents are ruled out from the analysis. Each line in Figure 3.21 represents the accuracy trend for a particular source-target couple, namely, $B \to D$, $D \to B$, $B \to E$, $E \to B$, $D \to E$, $E \to D$; further, an extra line is visible, portraying the average trend.

We can see that accuracy decreases on average when minimum document frequency increases. This probably is due to the fact that document frequency is an unsupervised technique; therefore, considering smaller feature sets (i.e. higher *df* values) comes out not to be effective in classifying target documents. Unsupervised methods do not guarantee that the selected features help in discriminating among categories and, as such, a bigger dictionary could support the learning phase. Further, accuracy is lower anytime E is considered, because E is a completely different domain with respect to B and D, which instead have more common words. Therefore, it is increasingly important to select the best possible feature set in order to help transfer learning.

An alternative to unsupervised methods consists in choosing the Bing Liu wordlist as dictionary. In this case, no parameter needs to be tuned but Porter stemmer has to be applied to the wordlist so that terms inside documents match those in the wordlist. Comparing the performance of our algorithm when using opinion words as features with that achieved using terms having minimum $df = 25$ (Figure 3.22), we may notice that the former outperforms the latter on average. This outcome suggests that, when features have a stronger polarity, inferred by the source domain, transfer learning is eased and our algorithm achieves better performance.

The Bing Liu wordlist only contains opinion words, which are general terms having well-known polarity. Nevertheless, there may be other words, possibly domain dependent, fostering the classification process. For this purpose, supervised feature selection techniques can be exploited to build the dictionary to be used. We would like to remark that, including domain dependent terms, the transfer learning capability of our method is increasingly important to let information about polarity flow from source domain to target domain terms.

Below, two tests are presented with reference to supervised feature selection techniques: in the former (Figure 3.23), features are selected by means of chi-squared (χ^2), varying the number of selected features from 250 to 1000 with step length 250; in the latter (Figure 3.22), the two supervised feature selection techniques mentioned in 3.7.1 are compared. Figure 3.23 reveals that there are no relevant variations on average in performance by increasing the number of features to be selected. On the one hand, this means that 250 words are enough to effectively represent the source-target couple, reminding that these are the best features according to the supervised method used. On the other hand, this proves that our algorithm produces stable results by varying the number of features to be selected.

Figure 3.22 shows that on average the usage of a supervised feature

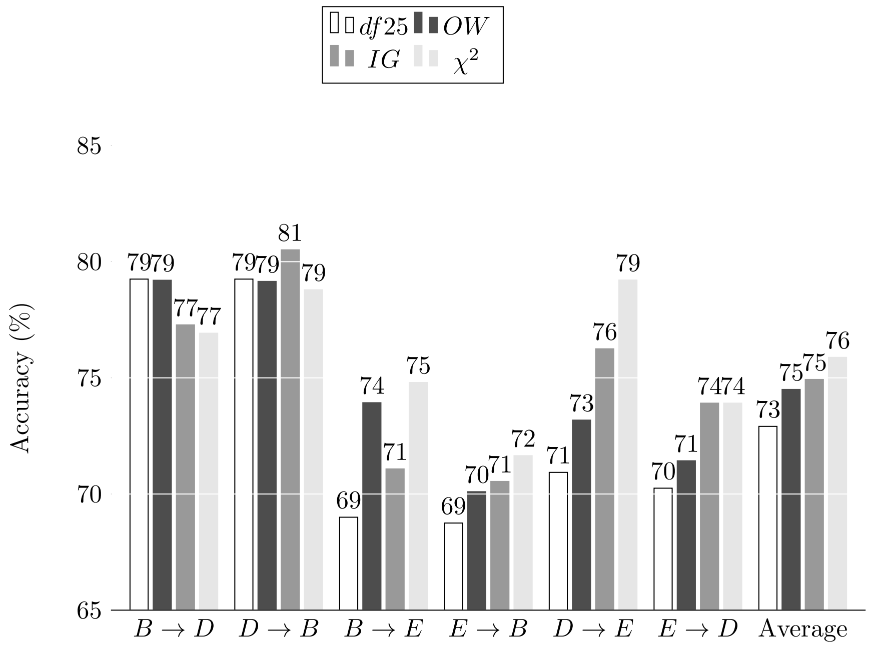


Figure 3.22 – *Cross-domain classification by comparing some feature selection methods, such as the unsupervised minimum document frequency df, the opinion wordlist and the supervised information gain IG and chi-square χ^2 scoring functions. Minimum $df = 25$ is set regarding the unsupervised method. Instead, the best 250 features are selected in accordance with the supervised scoring functions.*

selection technique is better than selecting just well-known opinion words. This confirms the ability of our method in performing transfer learning. Further, the same configuration, where 250 features are selected by using χ^2 scoring function, also achieves better accuracy than using IG to perform the feature selection process (Figure 3.22). Summing up, our Markov chain based method achieves its best performance in Amazon datasets by selecting the best 250 features by means of χ^2 scoring function in the text pre-processing phase.

Table 3.18 displays a comparison between our new method, referred as MC, and other works, namely SFA and PBT. Accuracy values are comparable, despite both SFA and PBT perform better on average. On the other hand, we would like to put in evidence that our algorithm only

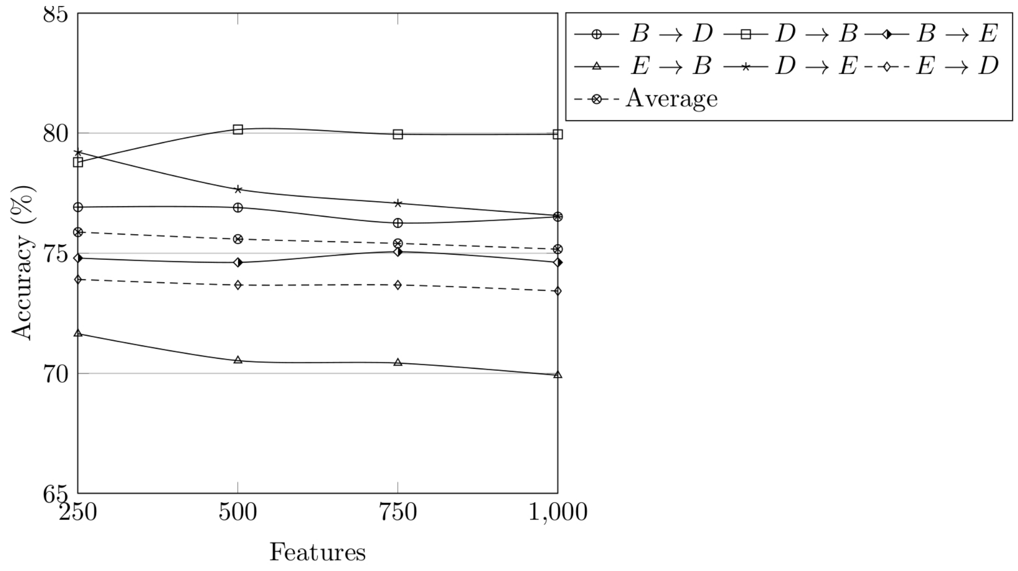


Figure 3.23 – *Cross-domain classification by varying the number of features to be selected in a supervised way. χ^2 scoring function is used in order to select features.*

requires 250 features, whereas PBT needs 2000 features and SFA more than 470000. Therefore, growing the computational complexity of the three approaches quadratically with the number of features, the convergence of our algorithm is supposed to be faster even if this hypothesis cannot be proved without implementing the other methods.

Finally, in Table 3.19 we can see that similar considerations can be done in an in-domain setting. Notice that nothing needs to be changed in our method to perform in-domain sentiment classification, whereas other works use standard classifiers completely bypassing the transfer learning phase.

3.7.3 Discussions

In this section is introduced a novel method for in-domain and cross-domain sentiment classification relying on Markov chain theory. We proved that this new technique not only fulfils the categorisation task, but also allows transfer learning from source domain to target domain in cross-domain setting. The algorithm aims to build a Markov chain transition matrix, where

Table 3.18 – *Results in cross-domain sentiment classification, compared with other works. For each dataset, the best accuracy is in bold.*

	MC	SFA	PBT
$B \rightarrow D$	76.92%	**81.50**%	81.00%
$D \rightarrow B$	78.79%	78.00%	**79.00**%
$B \rightarrow E$	74.80%	72.50%	**78.00**%
$E \rightarrow B$	71.65%	**75.00**%	73.50%
$D \rightarrow E$	**79.21**%	77.00%	79.00%
$E \rightarrow D$	73.91%	**77.50**%	76.00%
Average	75.88%	76.92%	77.75%

Table 3.19 – *Results in in-domain sentiment classification, compared with other works. For each dataset, the best accuracy is in bold.*

	MC	SFA	PBT
$B \rightarrow B$	76.77%	**81.40**%	79.96%
$D \rightarrow D$	**83.50**%	82.55%	81.32%
$E \rightarrow E$	80.90%	**84.60**%	83.61%
Average	80.39%	82.85%	81.63%

states represent either terms or classes, whose co-occurrences in documents, in turn, are employed to compute state transitions. Then, a single step in the matrix is performed for the sake of classifying a test document.

The proposed approach was compared with other two works and we showed that it achieves comparable performance in term of effectiveness. On the other hand, lower parameter tuning is required than previous works, since only the pre-processing parameters need to be calibrated. Finally, in spite of having a comparable computational complexity, growing quadratically with the number of features, much fewer terms are demanded to obtain good accuracy.

3.8 Text Clustering

Text clustering refers to the identification of groups (*clusters*) of similar documents in a potentially large set of them (Liu et al., 2006; Aggarwal and Zhai, 2012). Likely to text classification described above, the goal is to organize documents in groups; the difference is that the groups are not priorly given.

As in general data clustering, groups should be created so that each document is largely similar to others of the same group and different from those of other groups. Clustering results may be used to automatically organize a vast collection in groups of similar documents in order to ease their browsing. In this case, to be more useful in practice, especially if their number is large, clusters should be automatically labeled with some meaningful names or descriptions of the documents in each. This task is a form of *text summarization*. The organization of documents in clusters may even be used as an indexing method to support other tasks, such as retrieval of documents relevant to an arbitrary query.

There exist different clustering approaches, the most known are:

- Connectivity models: the most common example is the h*ierarchical clustering*, it builds models based on distance connectivity.
- Centroid models: the most common is the *k-means* algorithm, it represents each cluster by a single mean vector.
- Distribution models: clusters are modeled using statistical distributions, such as multivariate normal distributions used by the *Expectation-maximization algorithm.*
- Density models: common examples are *DBSCAN* and *OPTICS*, they define clusters as connected dense regions in the data space.

In this dissertation, two are the clustering approaches used, namely the Density and Connectivity models. In the following they are further analysed.

Density-Based Clustering

As density-based clustering algorithm, the most used is the DBSCAN (Ester et al., 1996) which, given a set of points in some space, it groups together points that are closely packed together (with many nearby neighbors). Two are the algorithm parameters: the minimum number *min* of points per cluster, and a threshold θ that defines the neighborhood distance between points in a cluster.

The algorithm starts by an arbitrary starting point, which has not been visited, and by retrieving its θ-neighborhood it creates a cluster if the number of points in that neighborhood is equals or greater than min. In case that a point is a dense part of an existing cluster, its θ-neighbor points are retrieved and are added to the cluster, as is their own θ-neighborhood when they are also dense. This process stops when the density-connected cluster is completely found. Then, the algorithm processes new unvisited points in order to discover any further clusters.

Connectivity-Based Clustering

The most common example of this approach is the Agglomerative hierarchical clustering method (Bouguettaya et al., 2015), where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Running the agglomerative method requires the choice of an appropriate linkage criteria which is used to determine the distance between sets of observations as a function of pairwise distances between clusters that should be merged or not. Three are the most commonly used linkage criteria, namely the *single*, *complete* and *average linkage* (Manning et al., 2008). Given two clusters Ω_y and Ω_z, the three linkage criteria are defined as follows:

$$singleLinkCl(\Omega_y, \Omega_z) = min\{SIM(\omega_i, \omega_j) : \omega_i \in \Omega_y, \omega_j \in \Omega_z\}$$

$$completeLinkCl(\Omega_y, \Omega_z) = max\{SIM(\omega_i, \omega_j) : \omega_i \in \Omega_y, \omega_j \in \Omega_z\}$$

$$avgLinkCl(\Omega_y, \Omega_z) = \frac{1}{|\Omega_y||\Omega_z|} \sum_{\substack{\omega_i \in \Omega_y \\ \omega_j \in \Omega_z}} SIM(\omega_i, \omega_j)$$

The agglomerative clustering method consists into an iterative process that merges the two clusters with highest average linkage score. After each merge of clusters, the algorithm starts by recomputing the new average linkage scores between all clusters. This process runs until a cluster pair exists with a similarity greater than a given threshold.

3.9 Conversation Threads Detection

In recent years, online texting has become a part of most people's everyday lives. The use of email, web chats, online conversations and social groups has become widespread. It is a fast, economical and efficient way of sharing information and it also provides users the ability to discuss different topics with different people. Analysis of their context finds a large spectrum of applications, ranging from social network marketing, expert finding, improvement of user experience in email management, and understanding of human relations and interactions (Jurczyk and Agichtein, 2007; Coussement and den Poel, 2008; Glass and Colbaugh, 2010; F. M. Khan and Pottenger, 2002).

The increased use of online fora leads to people being overwhelmed by information. For example, this can happen when a user has hundreds of new unread messages in a chat or one needs to track and organise posts in forums or social groups. To instantly have a clear view of different discussions, which requires expensive and tedious efforts for a person, we need to automatically organise this data stream into threads.

There has been considerable effort in extracting topics from document sets, mainly through a variety of techniques derived from Probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). However, the problem of detecting threads from conversational messages differs from document topic extraction for several aspects: (i) conversational message are generally much shorter than usual documents making the task of topic detection much more difficult (ii) thread detection strongly depends on social interactions between the users involved in a message exchange, (iii) as well the time of the discussion. In particular, the authors of (F. M. Khan and Pottenger, 2002; Adams and Martell, 2008) focus on detecting threads from chat dialogs. Specifically, the former uses patterns provided by experts, and the latter an augmented $tf.idf$. Thread detection in text streams is studied in (Shen et al., 2006; Huang et al., 2012). The authors of (Shen et al., 2006) presents a single-pass clustering algorithm based on features exploiting the temporal information of text streams. A single-pass clustering algorithm is also used in (Huang et al., 2012) which employs a contextual correlation between short text streams. In (Yeh, 2006), the authors use string similarity metrics and a heuristic algorithm to reassemble threads in the absence of header information in email conversations. Finally, recent work in (Dehghani et al., 2013) studies the conversation tree reconstruction, by first detecting the threads from a set of emails. Specifically, they map the thread detection problem to a graph clustering task. They create a semantic network of a set of emails where the nodes denote emails and the weighted edges represent co-thread relationships between emails. Then, they use a clustering method to extract the conversation threads. However, their approach is focus only on email datasets and their results are strongly bound with the used features, since when they do not take into account all features they have a high reduction in their accuracy.

Other works deal with thread tree reconstruction (Wang et al., 2008b; X. Wang and Chen, 2008; Aumayr et al., 2011) where given a known thread they seek to construct the conversation tree. In particular, in (Wang et al., 2008b), they reconstruct the thread structure in discussion forums where explicit meta-data is missing. Thread reconstruction in forums is also studied in (Aumayr et al., 2011), where the authors propose an algorithm which uses reply behaviours in threads in order to output the threads' structure. The authors of (X. Wang and Chen, 2008) use headers extracted from emails to create the parent/child relationships of a thread.

3.10 Combining Clustering and Classification for Thread Detection in Conversational Messages

This section addresses the efficient detection of conversation thread from pools of online messages - for example from social groups, pages chats, email messages etc. - namely the detection of sets of messages related with respect to contents, time and involved users. The thread detection problem is different from thread tree reconstruction, since the latter requires that the conversation threads are known. In contrast the former, seeks to find the messages which form a conversation thread. In other words, thread detection is the essential initial step of thread tree reconstruction.

We consider a three dimensional representation (Zhao and Mitra, 2007) which consists of text content, temporal information, and social relations. In Figure 3.24, we depict the three dimensional representation which illustrates 3 threads with different colours and shapes, that yields to total of 14 messages. The green circles and red squares threads have the same so-

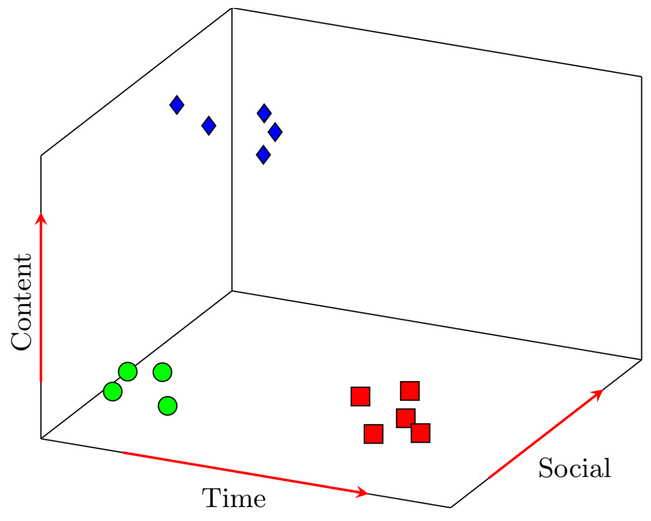


Figure 3.24 – *Three dimensional representation of threads messages*

cial and content dimensions but not time. While the blue diamonds thread treats different topics and users, it occurs in the same time frame of the green circles one. The use of the three dimensional representation leads to emphasis of thread separation.

We propose several measures to exploit the messages features, based on this three dimensional representation. Then, the generated features are embedded into a metric distance in density and hierarchical clustering algorithms (Ester et al., 1996; Bouguettaya et al., 2015) which cluster messages in threads. In order to enhance our approach to efficiently detect threads in any type of dataset, we build a classification model from a set of messages previously organised in threads. The classifier exploits the same features used in the clustering phase and it returns the probability of a pair of messages of belonging to the same thread. In other words, a binary supervised model is trained with instances, each referring to a pair of messages. Each instance uses the same features described previously, and a label describing whether the two messages belong to the same thread or not. This model provides a probability of being in the same thread for a pair of messages, we propose to use this probability as a similarity distance in clustering methods to detect the threads. We observe that the classifiers output can help the clustering process to achieve higher accuracy by identifying the threads correctly. We have extensively evaluated our approach with real world datasets including emails and social group chats. Our experimental results show that our method can identify the large majority of the threads in several type of dataset, such as web conversation including emails, chats and posts.

In summary, this section presents the following contributions:

- a three dimensional message representation based on textual semantic content, social interactions and time to generate features for each message;
- clustering algorithms to detect threads, on top of the features generated from the three dimensional representation;
- combination of the generated features to build a classifier that identifies the membership probability of pair of messages to the same thread and this probability is used as a distance function for the clustering methods to detect threads;
- the combined classification technique with clustering algorithms provides higher accuracy than solely using the clustering.

3.10.1 Method

In this section, we outline a generic algorithm for finding out the threads from a set of messages $\mathcal{M}$, such as emails, social group posts and chats. As an intermediate step, the algorithm addresses the problem of computing the similarity measure between pairs of messages. We propose a suite of features and two methods to combine them (one unsupervised and one supervised) to compute the similarity measure between two messages. We also present clustering algorithms which detect threads based on this similarity measure.

Data Model

We consider a set of messages $\mathcal{M} = \{m_1, m_2, ...\}$ that refers to online texts such as emails, social group chats or forums. Each message is characterized by the following properties: (1) pieces of text data (content and subject in case of emails), (2) creation time, and (3) the involved users (author or sender/recipients in case of emails). We represent each message as a three-dimensional model (Zhao and Mitra, 2007; Zhao et al., 2007) to capture all these components. Thus, a message $m \in \mathcal{M}$ can be denoted as a triplet $m = <c_m, \mathcal{U}_m, t_m>$, where c_m refers to text content, $\mathcal{U}_m = \{u_1, u_2, ...\}$ refers to the set of users that are involved in m, and t_m refers to the creation time. Some dimensions can be missing, for instance chat, groups and forum messages provide only the author information, without any recipients.

A conversation thread is defined as a set of exchanged messages on the same topic among the same group of users during a time interval, more formally, the set of messages $\mathcal{M}$ is partitioned in a set of conversatons $\mathcal{C}$. Each message $m \in \mathcal{M}$ belongs to one and only one conversation $c \in \mathcal{C}$. The goal of the thread reconstruction task is to automatically detect the conversations within a pool of messages. To this aim, we propose a clustering-based method that relies on a similarity measure between a pair of messages, called $SIM(m_i, m_j)$. In the following, we define different proposed approaches to calculate the similarity measure. We will use the notation $\Omega = \{\omega_1, \omega_2, ...\}$ to refer the predicted extracted conversations.

Messages Features

Social text messages, like emails or posts, can be summarized by three main components: text content, temporal information, and social relations (Zhao and Mitra, 2007). Each of the three main components can be analyzed under different points of view to compute the distance between a pair of messages, which involves the creation of several features. The function $SIM(m_i, m_j)$ relies on these features and returns a similarity value for each pair of messages (m_i, m_j), which is used by the clustering algorithm that returns the finding threads. We now present the extracted features to measure the similarity between two messages.

The content component relies on the semantics of the messages. Two are the main sources: the messages text and the subject, if present (e.g., social network posts do not have this information). The first considered feature is the similarity of the messages text content. We make use of the common *bag of words* representation, that describes a textual message m by means of a vector $\mathbf{w}_m = \{w_1^m, w_2^m, ...\}$, where each entry indicates the presence or absence of a word w_i^m. Single words occurring in the messages text are extracted, discarding punctuation. A stopwords list is used to filter-out all the words that are not informative enough. The standard Porter stemming algorithm (Porter, 1980) is used to group words with common stems. The $tf.idf$ (Salton and Buckley, 1988) weighing scheme is used in order to balance the relative importance of each word. Using *BoW* representation, the similarity between two vectors m_i, m_j can be measured by means of the commonly used *cosine similarity* (Eq. 3.1). In data with available subject, the same process is carried out, computing the similarity cosine $f_{C_S}(m_i, m_j)$ of words contained in the messages subject.

The cosine similarity allows a lexical comparison between two messages but does not consider the semantic similarity between two messages. There are two main shortcomings of this measure: the lack of focus on keywords, or semantic concepts expressed by messages, and the lack of recognition of lexicographically different words but with similar meaning (i.e. synonyms), although this is partially computed through the stemming. In order to also handle this aspect, we extend the text similarity by measuring the correlation between entities, keywords and concepts extracted using AlchemyAPI (AlchemyAPI). AlchemyAPI is a web service that analyzes the unstructured content, exposing the semantic richness in the data. Among the various information retrieved by AlchemyAPI, we take into consideration the extracted topic keywords, involved entities (e.g. people, companies, organizations, cities and other types of entities) and concepts which are the abstractions of the text (for example, "My favorite brands are BMW and Porsche = "Automotive industry"). Each of these extracted information is retrieved by AlchemyAPI with a quantification ranging from 0 to 1 of the relatedness to the message text. We create three vectors, one for each different information, using the retrieved keywords/entities/concepts for each message and using the related relevance extracted by AlchemyAPI as weight. Again we compute the cosine similarity of these vectors, creating three novel features:

- $f_{C_K}(m_i, m_j)$: computes the cosine similarity of the keywords of m_i and m_j. This permit to quantify the similarity of the contents focusing only on the key aspects, namely the keywords, instead of the whole text content.

- $f_{C_E}(m_i, m_j)$: computes the cosine similarity of the entities in m_i and m_j, allowing the recognition of shared entities between the two messages.

- $f_{C_C}(m_i, m_j)$: computes the cosine similarity of the concepts in m_i and m_j, allowing a comparison of the two messages on an higher abstraction level: from words to the expressed concepts.

The second component is related to the social similarity. For each message m, we create a vector of involved users $\mathcal{U}(m) = \{u_1, u_2, ...\}$ defined as union of sender and recipients of m (note that the recipients information is generally not provided in social network posts). We exploit the social relatedness of two messages through two different features:

- The similarity of the users involved into the two messages $f_{S_U}(m_i, m_j)$, defined as the Jaccard similarity between $\mathcal{U}(m_i)$ and $\mathcal{U}(m_j)$:

$$f_{S_U}(m_i, m_j) = \frac{|\mathcal{U}(m_i) \cap \mathcal{U}(m_j)|}{|\mathcal{U}(m_i) \cup \mathcal{U}(m_j)|}$$

- The neighborhood Jaccard similarity $f_{S_N}(m_i, m_j)$ of the involved users. The neighborhood set $\mathcal{N}(u)$ of an user u is defined as the set of users that have received at least one message from u. We also include each user u in its neighborhood $\mathcal{N}(u)$ set. The neighborhood similarity of two messages m_i and m_j is defined as follows:

$$f_{S_N}(m_i, m_j) = \frac{1}{|\mathcal{U}(m_i)||\mathcal{U}(m_j)|} \sum_{\substack{u_i \in \mathcal{U}(m_i) \\ u_j \in \mathcal{U}(m_j)}} \frac{|\mathcal{N}(u_i) \cap \mathcal{N}(u_j)|}{|\mathcal{N}(u_i) \cup \mathcal{N}(u_j)|}$$

Finally, the last component relies on the time of two messages. We define the time similarity as the logarithm of the inverse of the distance between the two messages, expressed in days, as follows:

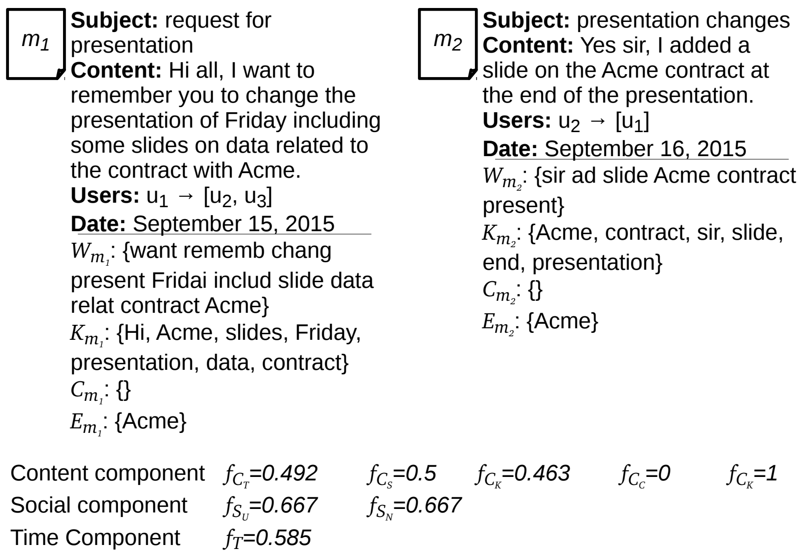
$$f_T(m_i, m_j) = \log_2(1 + \frac{1}{1 + |t_{m_i} - t_{m_j}|})$$

We use the inverse normalization of the distance in order to give a value ranging from 0 to 1, where 0 correspond to a high temporal distance and values close to 1 (thus the highest score) refers to messages with low distance.

As a practical example, Figure 3.25 shows two messages, with the related properties, and the values of the features generated from them.

Clustering

In this section, we present the clustering methods used to detect the threads. Based on the set of aforementioned features $\mathcal{F} = \{f_{C_T}, f_{C_S}, f_{C_K}, f_{C_E}, f_{C_C},$



Subject: request for presentation
Content: Hi all, I want to remember you to change the presentation of Friday including some slides on data related to the contract with Acme.
Users: $u_1 \rightarrow [u_2, u_3]$
Date: September 15, 2015
W_{m_1}: {want rememb chang present Fridai includ slide data relat contract Acme}
K_{m_1}: {Hi, Acme, slides, Friday, presentation, data, contract}
C_{m_1}: {}
E_{m_1}: {Acme}

Subject: presentation changes
Content: Yes sir, I added a slide on the Acme contract at the end of the presentation.
Users: $u_2 \rightarrow [u_1]$
Date: September 16, 2015
W_{m_2}: {sir ad slide Acme contract present}
K_{m_2}: {Acme, contract, sir, slide, end, presentation}
C_{m_2}: {}
E_{m_2}: {Acme}

Content component	f_{C_T}=0.492	f_{C_S}=0.5	f_{C_K}=0.463	f_{C_C}=0	f_{C_E}=1
Social component	f_{S_U}=0.667	f_{S_N}=0.667			
Time Component	f_T=0.585				

Figure 3.25 – *Example of features calculation for a pair of messages. For each message we depict the various components: Subject, Content, Users (sender $\rightarrow$ recipients) and creation date. $\mathbf{w}_{m_i}$ refers to the bag of words of a message obtained after the tokenization, stopwords removal and stemming. The vectors of keywords ($\mathcal{K}(m_i)$), concepts ($\mathcal{C}(m_i)$) and entities ($\mathcal{E}(m_i)$) extracted from AlchemyAPI are shown. In the bottom the values for each proposed feature are also shown. For simplicity, we assume binary weight for each word, keyword, concept or entity.*

f_{S_U}, f_{S_N}, $f_T\}$, we define a distance measure that quantifies the similarity between two messages:

$$SIM(m_i, m_j) = \Pi_{f \in \mathcal{F}}(1 + f(m_i, m_j)) \tag{3.49}$$

We compute a $N \times N$ matrix with the similarities between each pair of emails (m_i, m_j) and we use density based and hierarchical clustering algorithms (Section 3.8), being the two most common distance-based approaches. For the Agglomerative clustering algorithm we examined, in preliminary experiments, the three linkage criteria and we observed that average linkage clustering leads to the best results.

Classification

The clustering algorithms described above rely on the similarity measure SIM, that combines with a simple multiplication several features, to obtain a single final score. This similarity measure in Eq. 3.49 gives the same weight, namely importance, to each feature. This allows avoiding the tuning of parameters related to feature weights, but could provide an excessively rough evaluation. A different possible approach, is to combine the sub components of similarity measure SIM as features into a binary supervised model, in which each instance refers to a pair of messages, the features are the same described above and the label is one if the messages belong to the same thread and zero otherwise. At runtime, this classifier is used to predict the probability to be in the same thread of two messages, using this probability as distance between the pairs of messages into the same clustering algorithms. The benefit of such approach is that it automatically finds the appropriate features to use for each dataset and it leads to a more complete view of the importance of each feature. Although it is shown in (Aumayr et al., 2011) that decision trees are faster and more accurate in classifying text data, we experimented with a variety of classifiers.

The classification requires a labeled dataset to train a supervised model. The proposed classifier relies on data in which each instance represents a pair of messages. Given a set of training messages $\mathcal{M}_{Tr}$ with known conversation subdivision, we create the training set coupling each training message $m \in \mathcal{M}_{Tr}$ with n_s messages of $\mathcal{M}_{Tr}$ that belong to the same thread of m and n_d messages belonging to different threads. We label each training instance with one if the corresponding pair of messages belong to same thread and zero otherwise. Each of these coupled messages are picked randomly. Theoretically we could create $(|\mathcal{M}_{Tr}| \cdot |\mathcal{M}_{Tr} - 1|)/2$ instances, coupling each message with the whole training set. In preliminary tests using *Random Forest* as the classification model, we notice that coupling each training message with a few dozen same and different messages can attain higher performances. All the experiments are conducted using $n_s = n_d = 20$, i.e. each message is coupled with at maximum 20 messages of the same conversation and 20 of different ones. We will refer to the proposed clustering algorithm based on a supervised model, as *SVC*.

Table 3.20 – *Characteristics of datasets*

Dataset	# messages	# threads	# users
BC3	261	40	159
Apache	2945	334	113
Redhat	12981	802	931
WhoWorld	2464	132	1853
HealthyChoice	1115	132	601
Healthcare Advice	3436	468	801
Ireland S. Android	4831	408	354

3.10.2 Experimental Results

In this section, we compare the accuracy of the clustering methods described in the previous section in terms of detecting the actual threads.

Datasets

For evaluating our approach we consider the following seven real datasets:

- The *BC3* dataset (Ulrich et al., 2008), which is a special preparation of a portion W3C corpus (Soboroff et al., 2006) that consists of 40 conversation threads. Each thread has been annotated by three different annotators, such as extractive summaries, abstractive summaries with linked sentences, and sentences labeled with speech acts, meta sentences and subjectivity.

- The *Apache* dataset which is a subset of Apache Tomcat public mailing list[10] and it contains the discussions from August 2011 to March 2012. We have removed small threads, the ones that contain fewer than four emails.

- The *Redhat* dataset which is a subset of Fedora Redhat Project public mailing list[11] and it contains the discussions that took place in the first six months of 2009. We eliminated all threads with fewer than four emails.

[10]http://tomcat.apache.org/mail/dev
[11]http://www.redhat.com/archives/fedora-devel-list

- Two Facebook pages datasets, namely *Healthy Choice*[12] and *World Health Organizations*[13]. They consist of real posts and relative replies between June and August 2015. We considered only the text content of the posts (discarding links, pictures, videos, etc.) and only those written in English (AlchemyAPI is used to detect the language), we have removed small conversation with less than four posts.

- Two Facebook public groups datasets, namely *Healthcare Advice*[14] and *Ireland Support Android*[15]. They consist of conversations between June and August 2015. Also for this dataset we considered only the text content of the posts written in english and only the non-small threads.

We use the first three datasets that consist of emails in order to compare our approach with existing related work (Dehghani et al., 2013; Erera and Carmel, 2008; Wu and Oard, 2005) on conversation thread reconstruction in email messages. To our knowledge, there are no publicly available datasets of social network posts with a gold standard of conversation subdivision. We use the four Facebook datasets to evaluate our method in a real social network domain.

The considered datasets have different peculiarities, in order to evaluate our proposed method under several perspectives. *BC3* is a quite small dataset (only 40 threads) of emails, but with the peculiarity of being manually curated. In this dataset is possible to have emails with different subjects in the same conversation. However, in *Apache* and *Redhat* the messages in the same thread, have also the same subject. With regards to Facebook datasets, we decided to use both pages and groups. Facebook pages are completely open for all users to read and comment in a conversation. In contrast, only the members of a group are able to view and comment a group post and this leads to a peculiarity of different social interaction nets. Furthermore, each message - post - in these datasets has available only the text content, the sender and the time, without information related to subject and recipients. Thus, we do not take into account the similarities that use the recipients or subject. Table 3.20 provides a summary of the characteristics of each dataset.

[12]https://www.facebook.com/healthychoice

[13]https://www.facebook.com/WHO

[14]https://www.facebook.com/groups/533592236741787

[15]https://www.facebook.com/groups/848992498510493

In the experiments requiring a labeled set to train a supervised model, the datasets are evaluated with 5-fold cross-validation, subdividing each of those in 5 thread folds.

Evaluation Metrics

The *precision*, *recall* and F_1-*measure* (Manning et al., 2008) are used to evaluate the effectiveness of the conversation threads detection. Here, we explain these metrics in the context of the conversation detection problem. We evaluate each pair of messages in the test set. A true positive (TP) decision correctly assigns two similar messages to the same conversation. Similarly, a true negative (TN) assigns two dissimilar messages to different threads. A false positive (FP) case would be when the two messages do not belong to the same thread but are labelled as co-threads in the extracted conversations. Finally, false negative (FN) case is when the two messages belong to the same thread but are not co-threads in the extracted conversations. From this four possible cases, we compute the common precision (p), recall (r) and F_1-*measure*.

We also use the *purity* metric to evaluate the clustering. The dominant conversation is selected from each extracted thread cluster, namely the one with the highets number of messages in the cluster itself. Then, purity is measured by counting the number of correctly assigned messages considering the dominant conversation as cluster label and finally dividing by the number of total messages. We formally define purity as

$$purity(\Omega, \mathcal{C}) = \frac{1}{|\mathcal{M}|} \sum_k \max_j |\omega_k \in c_j|$$

where $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ is the set of extracted conversations and $\mathcal{C} = \{c_1, c_2, ..., c_j\}$ is the set of real conversations.

To better understand the purity metric, we refer to the example of thread detection depicted in Figure 3.26. For each cluster, the dominant conversation and the number of related messages are: $\omega_1 : c_1, 4$, $\omega_2 : c_2, 4$, $\omega_3 : c_3, 3$. The total number of messages is $|\mathcal{M}| = 17$. Thus, the purity value is calculated as $purity = (4 + 4 + 3)/17 = 0.647$.

A final measure of the effectiveness of the clustering method, is the simple comparison between the the number of detected threads ($|\Omega|$) against the number of real conversations ($|\mathcal{C}|$).

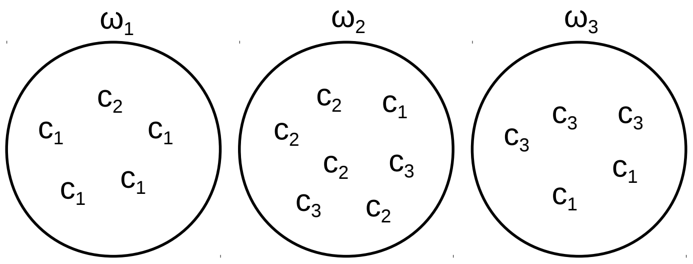


Figure 3.26 – *Conversation extraction example. Each ω_k refers to an extracted thread and each c_j corresponds to the real conversation of the message.*

Table 3.21 – *Conversation detection results on email datasets. The top part of the table shows the results obtained by methods using subject information, while the lower part shows those achieved without such feature. For clustering and our approaches we report results with best threshold tuning.*

	BC3			Apache			Redhat		
Methods	p	r	F_1	p	r	F_1	p	r	F_1
Wu and Oard	0.601	0.625	0.613	0.406	0.459	0.430	0.498	0.526	0.512
Erera and Carmel	0.891	0.903	0.897	0.771	0.705	0.736	0.808	0.832	0.82
Dehghani et al.	0.992	0.972	0.982	0.854	0.824	0.839	0.880	0.890	0.885
DBSCAN (+ subject)	0.871	0.737	0.798	0.359	0.555	0.436	0.666	0.302	0.416
Agglom. (+ subject)	1.0	0.954	0.976	0.358	0.918	0.515	0.792	0.873	0.83
SVC (+ subject)	1.0	0.986	**0.993**	0.998	1.0	**0.999**	0.995	0.984	**0.989**
DBSCAN	0.696	0.615	0.653	0.569	0.312	0.403	0.072	0.098	0.083
Agglomerative	1.0	0.954	0.976	0.548	0.355	0.431	0.374	0.427	0.399
SVC	1.0	0.952	**0.975**	0.916	0.972	**0.943**	0.966	0.914	**0.939**
SVC (no alchemy)	0.967	0.979	0.973	0.892	0.994	0.94	0.815	0.699	0.753

Results

In Table 3.21 and 3.22, we report the results obtained in the seven datasets. All the results in these tables are obtained using the Weka (Hall et al., 2009) implementation of *Random Forest* algorithm, using a 2×2 cost matrix with a weight of 100 for the instances labeled with one. The reported results are related to the best tuning of the threshold parameter of the clustering approaches, both for DBSCAN and Agglomerative. Further analysis on the parameters of our method will be discussed in the next section.

Table 3.21 shows the results on the email datasets, on which we can

Table 3.22 – *Conversation detection results on Facebook post datasets. For clustering and our approach we report results with best threshold tuning.*

	Healty Choice			World Health Org.		
Methods	p	r	F_1	p	r	F_1
DBSCAN	0.027	0.058	0.037	0.159	0.043	0.067
Agglomerative	0.228	0.351	0.276	0.154	0.399	0.223
SVC	0.67	0.712	**0.69**	0.552	0.714	0.623
SVC (no alchemy)	0.656	0.713	0.683	0.543	0.742	**0.627**
	Healthcare Advice			Ireland S. Android		
Methods	p	r	F_1	p	r	F_1
DBSCAN	0.206	0.051	0.082	0.201	0.002	0.004
Agglomerative	0.429	0.498	0.461	0.143	0.141	0.142
SVC	0.809	0.721	0.763	0.685	0.655	0.67
SVC (no alchemy)	0.802	0.733	**0.766**	0.708	0.714	**0.711**

compare our results (*SVC*) with other existing approaches, such as the studies of Wu and Oard (Wu and Oard, 2005), Erera and Carmel (Erera and Carmel, 2008) and the lastest one of Dehghani et al. (Dehghani et al., 2013). All of these existing approaches use the information related to the subject of the emails, we show in the top part of the table a comparison using also the subject as feature in our proposed approach. We want point out that in *Apache* and *Redhat* dataset, the use of the subject could make the clusterization effortless, since all messages of a thread have same subject. It is notable how our supervised approach obtains really high results, reaching almost perfect predictions and always outperforming the existing approaches, particularly in *Redhat* and *Apache* dataset.

In our view, the bottom of Table 3.21 is of particular interest, where we do not considered the subject information. The results, especially in *Redhat* and *Apache*, have a little drop, remaining anyhow at high levels, higher than all existing approaches that take into consideration the subject. Including the subject or not, the use of a supervised model to evaluate the similarity between two messages, brings a great improvement to the clustering performances, compared to the use of a simple combination of each feature as described in Section 3.10.1. In the lower part of Table 3.21 is also shown the effectiveness of our *SVC* predictor without the three features related to AlchemyAPI information; these features lead to an improvement

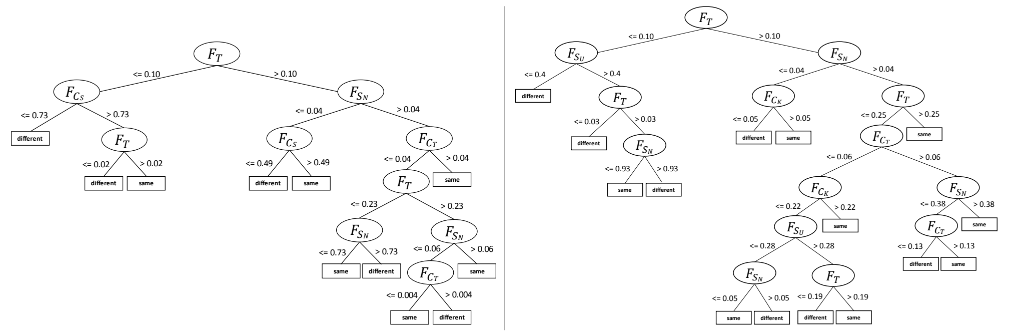


Figure 3.27 – *Decision trees created for BC3 dataset, (left) including and (right) excluding subject as feature.*

of results especially in *Redhat*, which is the largest and more challenging dataset.

The aforementioned considerations, are valid also for the experiments on social network posts. To the best of our knowledge, there is not any related work on such type of datasets. In Table 3.22, we report the results of our approach on the four Facebook datasets. These data do not provide the subject and recipients information of messages, thus the reported results are obtained without the features related to the subject and neighborhood similarities, namely $f_{C_S}(m_i, m_j)$ and $f_{S_N}(m_i, m_j)$. We notice that the pure unsupervised clustering methods, particularly DBSCAN, achieve low *precision* and *recall*. This is due to the real difficulties of these post's data: single posts are generally short with few semantic information. For example suppose we have two simultaneous conversations *t1: "How is the battery of your new phone?"* - *"good!"* and *t2: "how was the movie yesterday?"* - *"awesome!"*. By using only the semantic information of the content, it is quite impossible to associate the replies to the right question, thus the time and the social components become crucial. Although there is a huge literature to handle grammatical errors or misspelling, in our study we have not taken into account these issues. Despite these difficulties, our method guided by a supervised model achieve quite good results in such data, with an improvement almost always greater than 100% with respect the pure unsupervised clustering. Results in Table 3.22 show the difficulties also for AlchemyAPI to extract valuable information from short text posts. In fact, results using the AlchemyAPI related features does not lead to better results.

The learning algorithm we used, *Random Forest*, builds models in form of sets of decision trees, which predicts the relationship between an input features vector by walking across branches to a leaf node. An example of such a decision tree obtained in one of the models generated in the experiments is reported in Figure 3.27: to predict the relation between a pair of messages, the classifier start from the root and follow branches of the tree according to the computed features. The numbers in the edges are the threshold of feature values which classifier checks in order to choose which branch of the tree should pick. For example, in Figure 3.27(left) if threshold values are greater than 0.10 from F_T feature classifier selects F_{S_N}. We notice that the time similarity is the most significant feature in both cases. When we include the subject, we notice that the second significant features are the text cosine similarity and the Jaccard neighborhood. However, when we exclude the subject, involved users Jaccard similarity takes the place of text cosine similarity.

Parameter Tuning

Parameter tuning in machine learning techniques is often a bottleneck and a crucial task in order to obtain good results. In addition, for practical applications, it is essential that methods are not overly sensitive to parameter values. Our proposed method requires the setting of few parameters. In this section, we show the effect of changing different parameter settings. A first investigation of our *SVC* regards the supervised algorithm used to define the similarity score between a pair of messages. We conducted a series of experiments on the benchmark datasets varying the model. Namely, we used decision trees (*Random Forest* and *J48*), SVM (*LibSVM*) and *Logistic Regression*. For all the algorithms we used the Weka implementation using the default parameter values. For space reasons we omit the tuning experiments, we point out that the Random Forest algorithm obtained the best results, thus we decided to use it in all the conducted tests.

The main parameter of our proposed method regards the threshold value used in the clustering algorithms. We experimented with the use of a supervised model in the DBSCAN clustering algorithm, but we noticed the results were not good. This is not surprising if we consider how DBSCAN works: it groups messages in a cluster iteratively adding the neighbors of the messages belonging to the cluster itself. This leads to the erroneous merge of two different conversations, if just one pair of messages is misclassified as

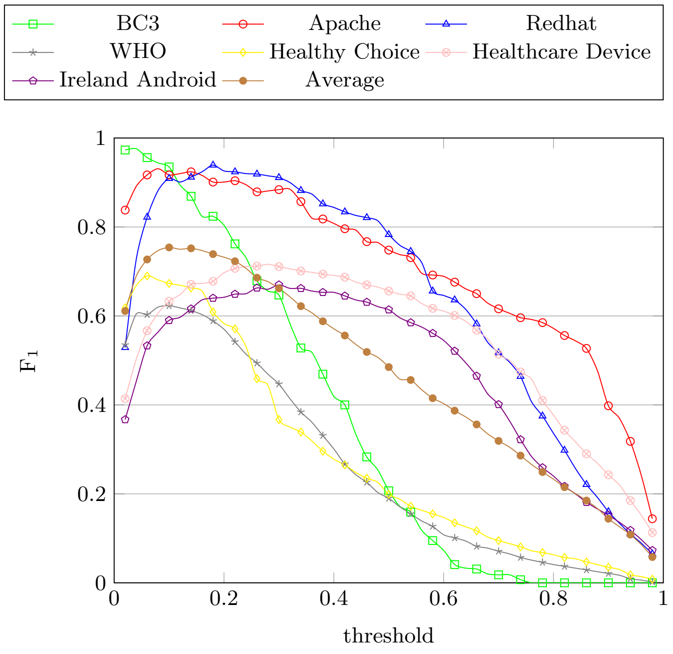


Figure 3.28 – *F_1 measure for varying number of threshold*

similar, bringing a sharp decline to the clustering precision. The previous issue, however, does not affect the agglomerative clustering, because of the use of average link of two messages inside two clusters, to decide whether to merge them or not. In this approach the choice of the threshold parameter is crucial, namely the stop merge criterion. Figure 3.28 shows the F_1 trend varying the agglomerative threshold, using the weighted Random Forest as the supervised model. Is notable that all the trends have only one peak that corresponds to a global maximum, thus with a simple gradient descent is possible to find the best threshold value. Furthermore, our method is generally highly effective for threshold values ranging from 0.1 to 0.3, as shown in Figure 3.28. This is also confirmed by the average trend, that has a peak with a threshold equal to 0.1.

Towards Subject-Based Supervised Models

In this section, we discuss the possibility of creating an - incomplete - training set from which to create the supervised model of *SVC*, with the peculiarity that is not a labeled set known a priori. This proposed method is particularly suitable for email datasets. The main assumption is that a conversation of emails can be formed by emails with the same subject and also by emails with different ones. It is quite common, but not always true, that emails with the same subject refer to the same conversation (Wu and Oard, 2005). This intuition can provide preliminary conversation detection of a message pool in an unsupervised way. Our proposal is to create the training set using this simple clusterization as labels for distinguishing messages belonging to the same or different threads (i.e. if the two messages have same subject). We can train the classifier with this labeled set using this model inside the clustering method as described in section 3.10.1. Since the subject is a known feature of an email dataset, this approach guarantees the use of a supervised classifier even if there is no labeled dataset as input.

From the benchmark datasets we considered, only BC3 provides emails with different subject in the same threads. To test this approach, there is no need of a crossfold validation: the whole dataset is initially labeled based only on the subject of emails, a classifier is trained with this data and then used to clusterize the starting dataset. The results are extremely promising: we obtain a *purity* and a *precision* equal to 1, a *recall* equal to 0.986 and the resulting F_1 *measure* is equal to 0.993, higher than the state of art.

3.10.3 Discussions

In this section, we focus on the problem of detecting threads from a pool of messages that correspond to social network chats, mailing list, email boxes, chats, forums etc. We address the problem by using a three-dimensional representation for each message, which involves textual semantic content, social interactions and creation time. Then, we propose a suite of features based on the three dimensional representation to compute the similarity measure between messages, which is used in a clustering algorithms to detect the threads. We also propose the use of a supervised model which combines these features using the probability to be in the same thread estimated by the model as a distance measure between two messages. We show that the use of a classifier leads to higher accuracy in thread detection, outperforming all earlier approaches.

Chapter 4

Recommendation

Recommender, or recommendation, systems (RS) are a subclass of IR that seek to predict the *rating* or *preference* that a user would give to an object. In their simplest form, personalized recommendations are offered as ranked lists of items. In performing this ranking, RSs try to predict what the most suitable products or services are, based on the users preferences and constraints. In order to complete such a computational task RSs collect from users their preferences, which are either explicitly expressed, e.g., as ratings for products, or are inferred by interpreting user actions (Ricci et al., 2011). RSs have become extremely common in recent years, and are applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general (Eck et al., 2008; Phelan et al., 2009; Adomavicius and Tuzhilin, 2005).

RSs are typically subdivided into two major approaches: collaborative or content-based filtering. *Collaborative filtering* approaches building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in (Sarwar et al., 2001). *Content-based filtering* approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties (Mooney and Roy, 2000). These two approaches are often combined into Hybrid Recommender Systems.

Each type of system has its own strengths and weaknesses. For instance, collaborative filtering techniques requires a large amount of information on users in order to make accurate recommendations. This leads to the common *cold start problem*, in which is hard, or even impossible, to make predictions for new users or users with few preferences (Schein et al., 2002). While content-based filtering needs few information to get started, it is far more limited in scope; for instance, it can only make recommendations that are similar to the original topics.

4.1 Job Recommendation

Job hunting (or job seeking or job searching) refers to the process people looking for a job perform in order to find it. Differently, finding out the right employee is a key aspect for enterprises, which continuously have to recruit according to their current needs. Both tasks are sides of the same general problem, namely allowing communication between companies and potential applicants for the sake of establishing an employment relationship. Since each manual search is onerous and tedious, methods exist that help automating these processes, such as *job recommendation systems* (Paparrizos et al., 2011; Malinowski et al., 2006) on the one hand and *recruiting systems* (Lee, 2007; McGovern et al., 2002; Baumgarten and Kelly, 2001; Eckhardt et al., 2008) on the other hand. The former cope with the problem of automatically finding a job which is as inherent to people skills as possible. Vice versa, the latter are used by Human Resources departments to select candidates fitting the skills enterprises are looking for. The concept of skills is crucial in both previous mentioned tasks, because it could help pointing out people capabilities even better than in the state of the art approaches, which only focus on either academic degree (Dinesh and Radhika, 2014; Chirumamilla et al., 2014) or preceding job positions (Paparrizos et al., 2011).

The recruiting (or recruitment) process, due to its fundamental importance, has been extensively studied in human resources field (Medsker et al., 1994; Allen and Van der Velden, 2001), giving increasing attention to the E-recruitment, namely a recruitment process based on web information (Kinder, 2000; Thompson et al., 2008) especially gathered from social networks such as LinkedIn, Facebook, Twitter, Xing (Broughton et al., 2013; Zide et al., 2014; Flecke, 2015; Peterson and Dover, 2014). For further reading, (Breaugh and Starke, 2000) and (Breaugh, 2008) offer a comprehensive analysis on the employee recruitment and some research issues. The majority of these works focus on the *human resource* aspect of the recruitment process (Buettner, 2014; Paparrizos et al., 2011). Instead the job hunting problem, i.e. finding out the best job positions available in relation to users' skills and qualities, has seldom been analyzed.

Several past studies proved the helpfulness of data mining and machine learning approaches for job placement. For instance, in (Min and Emam, 2003), rules created by a decision tree are used to manage the recruitment of truck drivers. For this study, the authors use empirical data derived from a survey submitted to several transport companies. (Buckley et al., 2004) present an automated recruitment and screening system, showing conservative savings of 6 to 1 ratio, due to reduced employee turnover, reduced staffing costs and increased hiring process efficiency. In (Chi, 1999), the authors apply principal component analysis to establish jobs that can be adequately performed by various types of disabled workers. A dataset of 1259 occupations summarized in 112 different jobs is used; 41 available skills are analyzed with principal component analysis, finding five principal factors (occupational hazard, verbal communication education and training, visual acuity, body agility, and manual ability). Finally, the 112 job titles are classified into 15 homogeneous clusters, creating useful data to expand both the counselor and counselee perspectives about job possibilities and job requirements through the five principal factors.

Some existing works (Dinesh and Radhika, 2014; Ramesh et al., 2011; Chirumamilla et al., 2014; Anuradha and Velmurugan, 2015) focus on the academic degree of students, aiming to predict both their academic performance at early stage of their curricula and their placement chance, using supervised classifiers like decision trees, SVMs or neural networks. (Elayidom et al., 2011) propose a decision tree based approach which helps students choosing a good branch that may fetch them placement, although the analyzed sectors are only rural and urban.

There exist several works related to job recommendation starting from the candidate profiles (Siting et al., 2012; Paparrizos et al., 2011). (Rafter et al., 2000) propose CASPER: an online Job Finder engine that uses a collaborative filtering algorithm with some user preferences. In (Malinowski et al., 2006) a bilateral people-job recommender system is proposed. A supervised probabilistic model estimates the probability that an applicant likes a job. The goal of the authors is to match applicants to job opening profiles, assuming that both are available. Differently, (Paparrizos et al.,

2011) recommend job positions to applicants based only on the job history of other employees. Starting from an individual who is currently employed in an institution, the aim of this work is to predict the next institution where the individual will be employed. (Buettner, 2014) proposes a recommender system based on social network information. This approach relies on three fit measures related to candidates, namely organization fit (a macro-perspective compatibility between the employees personalities and the organizations culture), group fit (a user social interactions perspective) and job fit (a micro-perspective matching between a candidates characteristics, such as abilities or preferences, and job demands). Not too different is the work of (Gupta and Garg, 2014), in which candidate profile matching as well as preserving their job behaviours or preferences are used to perform job recommendation.

User profiling is one of the major issues of these approaches, because retrieving, selecting and handling such data is hard. (Rubin et al., 2002) show the importance of extracurricular activities as users' skills indicator. Further, they point out that corporate recruiters and talent pool managers would benefit from a tool that allows the systematic categorization and valuing of extracurricular and contextual activities. (Paparrizos et al., 2011) define user profiles with three components: personal information, current and past professional positions, current and past educational information. Similarly, (Buettner, 2014; Gupta and Garg, 2014) use information related to companies, users, user preferences and social interactions; on the other hand, (Chi, 1999) uses a set of 41 skills.

According to (Zide et al., 2014), social media are seldom exploited for recruiting purposes. In their work, the authors study variables that recruiters assess when looking at applicants' LinkedIn profiles, finding some key elements (such as, among others, completeness of information and spelling mistakes) used when making hiring decisions.

4.2 Job Recommendation from Semantic Similarity of LinkedIn Users' Skills

Nowadays, except from custom private solutions possibly built in-house, social networks like LinkedIn[1], Facebook[2] and Twitter[3] are the most used recruiting systems by enterprises, because information available in such context are proven to be useful in the recruitment process (Flecke, 2015). (Davison et al., 2011) pointed out that LinkedIn provides more accurate information compared to Facebook because everybody in a person's network can easily contradict her assertions. This can be a reason why (Zide et al., 2014) define LinkedIn as the world's largest professional network. In addition to its reliability, LinkedIn also offers recruiter accounts aiming to support the recruiting process, so that about 94% of recruiters use it (Kasper, 2015). Instead, the same trend does not hold within social media job seekers, where only 40% makes use of this network, although members are sometimes notified of possibly interesting job offers. LinkedIn professional secrecy does not allow us a complete understanding of the techniques used to recommend job positions. Anyway, analyzing some public profiles and the relative recommended job positions, we notice that there are often wrongly retrieved (i.e. not interesting) offers because of homonymy. This issue could make the job seeking process less effective, more manually conducted and time consuming.

Diversely, everybody should have the opportunity of quickly finding a rewarding job that both satisfies the job seeker's inclination and ensures adequate quality of work and expectation for the company. These should be primary goals of a job recommendation system: it should help meeting requests and offers of jobs by favouring the best possible fit among candidates and companies according to people skills and companies' needs.

In this section, we introduce a job recommendation system based on *Latent Semantic Analysis* (LSA) (Dumais et al., 1988) for the support in the job seeking process, evaluating its performance through a hierarchical clustering of job positions. Hierarchical clustering aids to build a folksonomy (i.e. a job position taxonomy) useful to correlate jobs, whereas normally only each person's job positions are available as plain text. So, the basic

[1]`www.linkedin.com`
[2]`www.facebook.com`
[3]`twitter.com`

idea we rely on is first of all to discover similarity between different job positions and then to find out their latent associations with people skills.

Job positions are represented as vectors of skills and mapped into a transformed space by applying *LSA* to the skill-position co-occurrence matrix. Then, a complete-linkage hierarchical clustering technique is applied to correlate the transformed job positions, using cosine similarity as inter-cluster distance measure. Instead, the job recommendation algorithm we propose aims to suggest a list of recommended jobs that fit people skills. In fact, people are represented as vectors of skills just like job positions. The algorithm starts from a skill-position matrix built with training data and expanded by applying *LSA*. Afterwards, cosine similarity between people and positions in the skill-position matrix is computed for each test instance. Thus, since the algorithm basically outputs how much jobs fit people skills, an ordered list of recommended job positions can be built according to their similarity with each person's skills.

To evaluate our method, we take LinkedIn as reference scenario because of its widespread use, crawling real public profiles from which we can easily infer information about people skills and current job positions. We assume that current job position is the one fitting best the skill set of each person (i.e. the label of each test vector), although we are aware that this criteria is only partly correct, because somebody's job might not fit her skills. Then, we perform classification assigning the most likely k positions to each test vector; finally, we test performance by computing the maximum recall within the k suggested positions, exploiting the previously built hierarchy.

To the best of our knowledge, in literature there are no works about job recommendation exploiting either LinkedIn or other social networks. The most similar work with respect to our proposal and used data is (Bastian et al., 2014), presenting a large-scale topic extraction pipeline, which includes constructing a folksonomy of skills and expertise and implementing an inference and recommender system for skills. Similarly to our proposed approach, (Bastian et al., 2014) analyze LinkedIn users' skills, using a large-scale dataset of about 150,000 skill phrases (i.e keywords) extracted directly from LinkedIn; these skills are then semantically deduplicated through Wikipedia-based crowdsourcing into approximately 50,000 skill topics. The aim is to help users into profile skill filling; the authors propose several recommendation approaches like collaborative filtering, logistic regression but finally they opt for a Naïve Bayes classifier. The authors conclude their work pointing out the feasibility as well as the practical utility of job recommendation based on LinkedIn users' skills, without further discussion and detailed analysis. Finding out relationships between jobs and users' skills through machine learning techniques is one of the main proposal of our work, addressing both the recruitment and the job hunting processes.

Our approach is therefore not directly comparable with the state of the art, because we focus on large scale data in terms of both job positions and skills. Moreover, differently from other approaches (Chi, 1999; Malinowski et al., 2006; Paparrizos et al., 2011), we argue that our method does not require manually collecting data, because they are already available on social networks. Finally, we would like to point out that neither the hierarchical clustering of positions nor the job recommendation algorithm require parameters to be tuned. This evidence makes our approach easy to use and profitable in real scenarios, because little effort is required for a job seeker to find a position fitting her skills.

4.2.1 LinkedIn Profiles Dataset

The benchmark dataset we used has been extracted from publicly accessible LinkedIn profiles of users from Italy. Among other details, these profiles report the set of skills declared by their owners and the currently occupied job position: we used this information to compose our dataset.

Both skills and positions are specified by each user as free text: many of them are present in multiple instances across profiles, but the majority of skills are only present in a few or single profiles, due e.g. to typos or uncommon indications. Another issue emerging from free text is the use of different languages across different users. In our case, while many users filled in their profile in Italian due to being their native language, many others used English instead to target a wider audience with their public profiles. As a consequence of these aspects, the same actual skill or position can be present multiple times with different names.

We performed a couple of preprocessing operations to obtain two disjoint groups of profiles suitable as training and test sets, respectively. The former is used to compute similarities between skills and positions and to build the hierarchy. The latter is used instead to evaluate the accuracy of the recommendation method.

We started from a preliminary dataset split between 51,000 profiles for training and 37,000 for testing. The training set initially included 2,410 distinct positions: as some of them consisted only of punctuation (e.g. "?")

or were not actual jobs (e.g. “mr.”), we removed them from the set $\mathcal{P}$ of possible positions. This also required to remove all training and test profiles having any of them as their position.

In a second preprocessing step, we removed all training profiles declaring less than 3 skills, in order to improve the quality of our knowledge base. This led to the removal of both some positions and some skills, which were left without appearances through the training set. For consistency, we also removed them from the test set, consequently deleting profiles having any of the removed positions.

After these steps, we have a final dataset of 42,056 profiles for training and 30,639 for test, including 6,985 unique skills and 2,241 distinct positions. Distribution of both skills and positions is highly skewed: for example, the most recurring position is “studente” (Italian for *student*) with 1,693 training profiles and 1,383 test ones, while there are some positions with only one representative profile.

4.2.2 Method

We describe here the process used to perform clustering of job positions and to recommend such positions to any person given its set of skills.

Data Model

We consider a set $\mathcal{U} = \{u_1, u_2, \ldots\}$ of *user profiles* (hence just *profiles*), each of them being the description of a specific person.

To each profile u is associated a set $S(u)$ of *skills*, representing the competencies which the corresponding person declares to have. The same skill may be associated to more than one profile. The union of all distinct skills from all profiles is denoted with $\mathcal{S} = \{s_1, s_2, \ldots\}$.

To each profile u is also associated a current *job position* $p(u)$. The same job position may be the current one for more than one profile. We denote with $\mathcal{P} = \{p_1, p_2, \ldots\}$ the set of all distinct job positions.

Clustering of Job Positions

We are interested in obtaining a folksonomy of possible job positions (hence just *positions*) from the available data. In order to do so, we perform a hierarchical clustering of positions in $\mathcal{P}$.

To this extent, we need a structured representation of possible positions, in order to measure their mutual similarity. We extract a vector-based representation, where skills are used as features. Specifically, from the set $\mathcal{U}$ of known profiles, we build a $|\mathcal{S}| \times |\mathcal{P}|$ matrix $\mathbf{C}$ counting the co-occurrences between skills and positions across them. Each cell of $\mathbf{C}$ is computed as follows:

$$c_{i,j} = |u \in \mathcal{U} : s_i \in S(u) \land p_j = p(u)| \tag{4.1}$$

In practice, $c_{i,j}$ is the number of profiles having both s_i among skills and p_j as position. This matrix somehow represents each position as a mix of different skills, according to those possessed by persons having that job.

Within the set $\mathcal{S}$ of distinct skills, some of them are similar or strictly related to each other, such as "ms office" and "ms word"; moreover, the same actual skill may be represented more than once by different *synonym* elements of the set. This representation of the positions would be improved by augmenting for each position the relevance of skills related to those explicitly included in the mix.

In the text analysis domain, a well-known solution to this issue is *Latent Semantic Analysis* (LSA), where Singular Value Decomposition (SVD) is applied to a term-document matrix in order to obtain a lower-rank approximation of it, where the noise present within the data is mostly removed and the similarities between elements emerge (Dumais et al., 1988). Similarly, we apply LSA to $\mathbf{C}$ in order to obtain a transformed matrix $\mathbf{C}'$. First we apply SVD on the original matrix, thus obtaining three factors.

$$\mathbf{C} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T \tag{4.2}$$

Specifically, $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices with eigenvectors of $\mathbf{C}$, while Σ is a diagonal matrix with singular values. Each of these values represents the importance of a dimension in the latent space. Lower values are supposed to represent dimensions with noise: by setting them to 0 and multiplying back the three components, we obtain the transformed matrix $\mathbf{C}'$. In order to apply this transformation we need to decide which singular values are to be retained: we iterate through them in decreasing order and keep retaining them until their sum is at least 50% of all values sum. The rank of the resulting matrix $\mathbf{C}'$ is equal to the count of retained singular values.

The transformed matrix $\mathbf{C}'$, as the original one, contains for each position p_k a column vector $\mathbf{p}_k$ representing it. We evaluate the relatedness between two positions by means of *cosine similarity*. From this, we can trivially obtain a *distance* measure between positions.

$$d(p_a, p_b) = 1 - \cos(\mathbf{p}_a, \mathbf{p}_b) = 1 - \frac{\mathbf{p}_a \cdot \mathbf{p}_b}{||\mathbf{p}_a|| \cdot ||\mathbf{p}_b||} \tag{4.3}$$

The mutual distances between positions are finally given in input to a complete-linkage agglomerative clustering algorithm, which extracts a dendrogram of all positions.

The dendrogram has the form of a binary tree with positions as leaves. In Section 4.2.3 some excerpts of the dendrogram obtained in our tests are shown as examples.

Job Recommendation

Other than extracting a consistent hierarchy of positions, the knowledge of a set of profiles can be used to infer the most advisable job positions for some other profile u_e, whose set of skills $S(u_e)$ is given.

This constitutes in practice a job recommendation system, where the best positions are suggested to any person according to his or her skills. While the positions of known profiles are assumed to be correct, it should be noted that there are usually multiple advisable positions corresponding to a set of skills. A recommendation system should return a set of most likely positions and all of them can be equally valid.

The recommendation method we use is simply based on representing both positions and profiles as comparable vectors and seeking for each profile the positions with the most similar vectors. Skills of the set $\mathcal{S}$ are used as features. To each profile will correspond a ranking of the known positions, of which only the first k items are usually considered.

Each profile u_e is simply represented by a binary vector $\mathbf{u}_e$, whose values are 1 in correspondence of skills known by the person and 0 elsewhere.

For what regards the vectors representing positions, we consider two alternatives, both reusing intermediate results from the positions clustering method. The first option is to use columns of the skill-position co-occurrences matrix $\mathbf{C}$, the second one is to replace it with the low-rank approximated matrix $\mathbf{C}'$ computed through LSA as described above.

In both cases, we compare the profile skills vector $\mathbf{u}_e$ to each column $\mathbf{p}_j$ of the used matrix by means of cosine similarity. For a given number k of positions to be recommended, we return the k positions whose vectors are most similar to $\mathbf{u}_e$, whose set is denoted with $R_k(u_e)$: these constitute the positions recommended for u_e.

4.2.3 Experimental Results

The methodology described above to extract a hierarchy of job positions and to recommend them has been tested on a set of data extracted from LinkedIn. Operations have been carried out by software based both on the Java platform and on the open source R environment for statistical analysis.

Positions Hierarchy

We applied the first step of the methodology to infer a hierarchy of job positions from the training profiles. The goal of this part is to obtain a valid folksonomy of possible job positions, where similar occupations are grouped together and well separated from unrelated ones.

Due to the absence of a compatible gold standard, it is not possible to quantitatively evaluate the correctness of the inferred hierarchy. Instead, we browsed through the obtained tree to check whether the obtained clusters are meaningful. As a sample, we report in Figures 4.1, 4.2 and 4.3 three clusters of the hierarchy we obtained, also showing the bottom-most binary splits between elements. As discussed above, tracked positions have both English and Italian names; we provide in the figures a translation of the latter ones for readers' convenience.

From the first two samples, it can be observed that the clustering algorithm mostly succeeded in outlining groups of related job positions. As we used the co-occurrences with skills shared across profiles to infer the relatedness between positions rather than words used to express them, similar occupations are effectively grouped into the same cluster, even if expressed with different terms. For the same reason, also equal positions with distinct English and Italian names are mostly successfully grouped together.

On the other hand, as the sample of Figure 4.3 suggests, even some unrelated positions have been eventually grouped together in the clustering. This can be attributed to some singularities in the co-occurrences between skills and positions in our training set. While many positions are associated to a sufficient number of profiles and skills to have a consistent and valid representation, other ones occur too rarely throughout the profiles and end

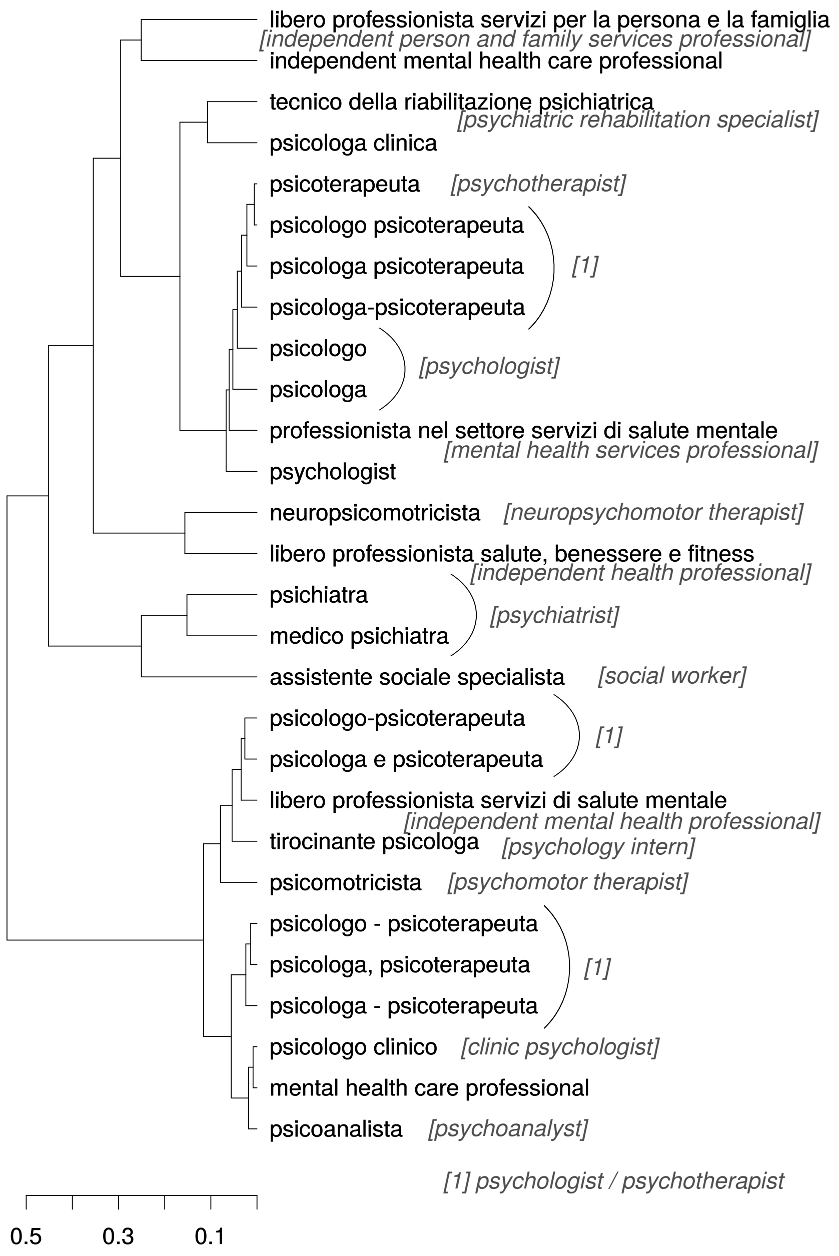

libero professionista servizi per la persona e la famiglia
[independent person and family services professional]
independent mental health care professional
tecnico della riabilitazione psichiatrica
[psychiatric rehabilitation specialist]
psicologa clinica
psicoterapeuta
[psychotherapist]
psicologo psicoterapeuta
psicologa psicoterapeuta
[1]
psicologa-psicoterapeuta
psicologo
psicologa
[psychologist]
professionista nel settore servizi di salute mentale
[mental health services professional]
psychologist
neuropsicomotricista
[neuropsychomotor therapist]
libero professionista salute, benessere e fitness
[independent health professional]
psichiatra
medico psichiatra
[psychiatrist]
assistente sociale specialista
[social worker]
psicologo-psicoterapeuta
psicologa e psicoterapeuta
[1]
libero professionista servizi di salute mentale
[independent mental health professional]
tirocinante psicologa
[psychology intern]
psicomotricista
[psychomotor therapist]
psicologo - psicoterapeuta
psicologa, psicoterapeuta
psicologa - psicoterapeuta
[1]
psicologo clinico
[clinic psychologist]
mental health care professional
psicoanalista
[psychoanalyst]
[1] psychologist / psychotherapist
0.5
0.3
0.1

Figure 4.1 – *Cluster of* mental health care *job positions.*

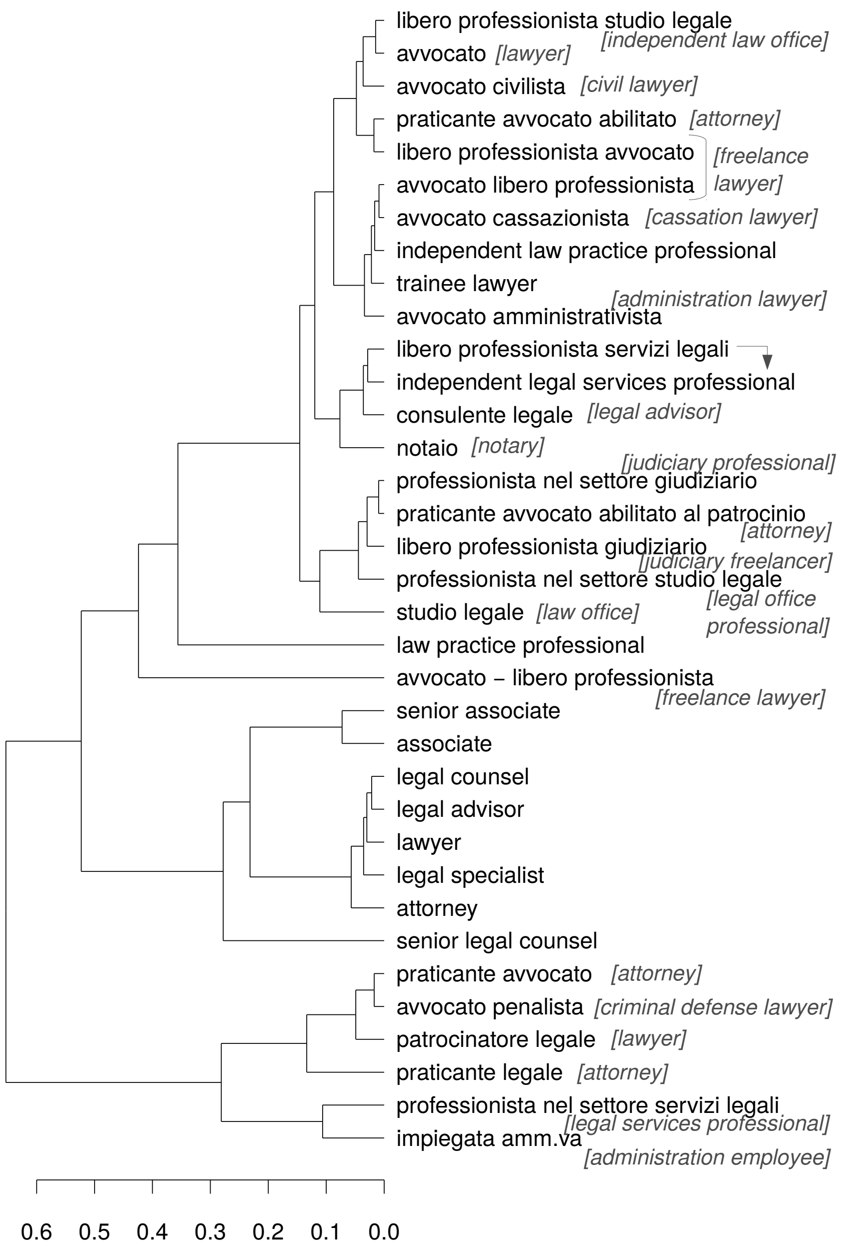

libero professionista studio legale
[independent law office]
avvocato [lawyer]
avvocato civilista [civil lawyer]
praticante avvocato abilitato [attorney]
libero professionista avvocato
avvocato libero professionista
[freelance lawyer]
avvocato cassazionista [cassation lawyer]
independent law practice professional
trainee lawyer
[administration lawyer]
avvocato amministrativista
libero professionista servizi legali
independent legal services professional
consulente legale [legal advisor]
notaio [notary]
[judiciary professional]
professionista nel settore giudiziario
praticante avvocato abilitato al patrocinio
[attorney]
libero professionista giudiziario
[judiciary freelancer]
professionista nel settore studio legale
[legal office professional]
studio legale [law office]
law practice professional
avvocato – libero professionista
[freelance lawyer]
senior associate
associate
legal counsel
legal advisor
lawyer
legal specialist
attorney
senior legal counsel
praticante avvocato [attorney]
avvocato penalista [criminal defense lawyer]
patrocinatore legale [lawyer]
praticante legale [attorney]
professionista nel settore servizi legali
[legal services professional]
impiegata amm.va
[administration employee]
0.6 0.5 0.4 0.3 0.2 0.1 0.0

Figure 4.2 – *Cluster of* legal *job positions.*

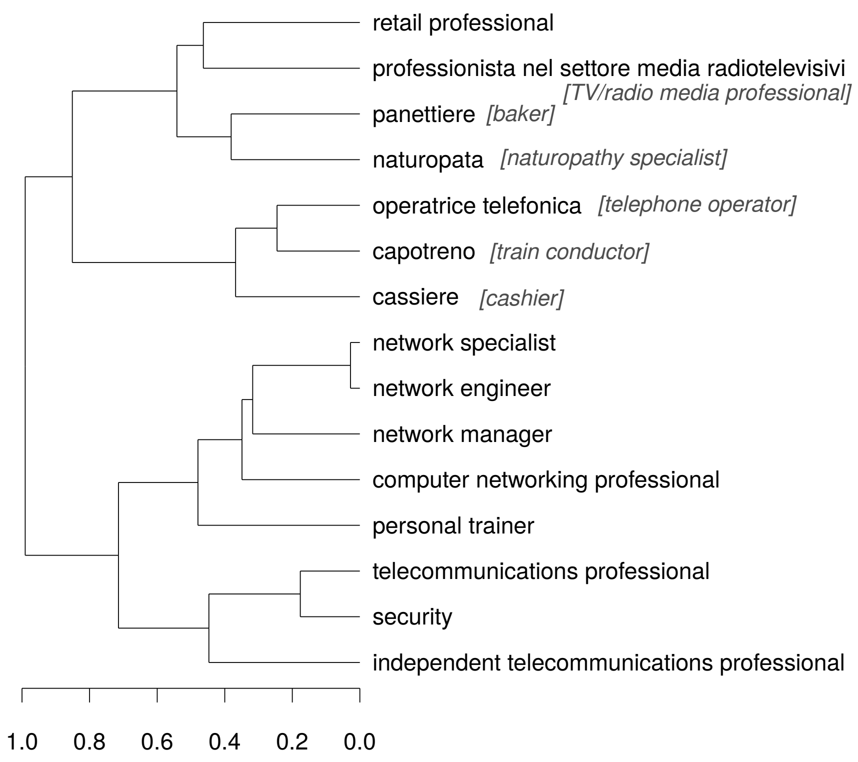


Figure 4.3 – *Cluster of mixed job positions.*

up being mostly associated to unrelated skills. This can be due for example to profiles with multiple qualifications not related to each other or to positions with few representative skills.

For example, by looking at the sample cluster, the "personal trainer" position has been considered similar to occupations dealing with computer networks. While this appears illogical, the cause can be found in the profiles used to infer the taxonomy. Of the 9 training profiles having "personal trainer" as the current position, two of them declare IT and telecommunications-related skills such as "linux" and "tcp/ip". In a profile set where there are no other occupations significantly similar to "personal trainer" with sufficient occurrences, this position ends up to be grouped with unrelated ones due to some profiles declaring their peculiar skills together, thus erroneously "linking" them. Another example is the "panettiere" (Italian for *baker*) position: only two of our training profiles declare this as current employment. While one of them explicitly includes "bakery" within abilities, all the other skills of both are unrelated, mostly consisting of very generic ones, such as "teamwork" and "problem solving", which can be equally linked to other uncommon positions.

To sum up, the obtained hierarchy successfully delineates a large number of groups of similar job positions, although with few clusters of unrelated occupations which are not sufficiently characterized in the training set. We used this hierarchy in the rest of our experiments to evaluate recommendations of job positions.

Results of Job Recommendation

In the second part of our experiments, we computed job recommendations for profiles of the test set, hereby denoted with $\mathcal{U}_{\text{test}}$, comparing the answers from our method with the known ones.

In our experimental evaluation, ignoring further information, we consider the current occupation of each person as the correct answer that should be given by the recommender. However, it should be noted that this position can't actually be with certainty among the best possible recommendations. There are many possible reasons for this: as discussed above, due to use of free text, a position may have many synonyms and misspelled variants which indicate the same concept but are considered distinct elements of $\mathcal{P}$. A recommended job may also be strongly related to the actual one, such that it requires a very similar set of skills. Ultimately, for practical reasons, any person may even be practicing a job which is notably unrelated to his or her skills. All these aspects introduce some outliers and potential errors in both training and test data; which can be detrimental for quantitative evaluations of accuracy.

The algorithm can output any number k of most recommended positions: according to the value of k, the known position of any test profile could either be among them or not. We want to evaluate for different values of k how much frequently the recommender hits the actual positions of test profiles. For all values of k ranging from 1 to 50, we evaluated the *recall@k*, i.e. the ratio of test profiles w.r.t. their total for which the known position is among the top k recommendations.

$$\text{R@}k = \frac{|u \in \mathcal{U}_{\text{test}} : p(u) \in R_k(u)|}{|\mathcal{U}_{\text{test}}|} \tag{4.4}$$

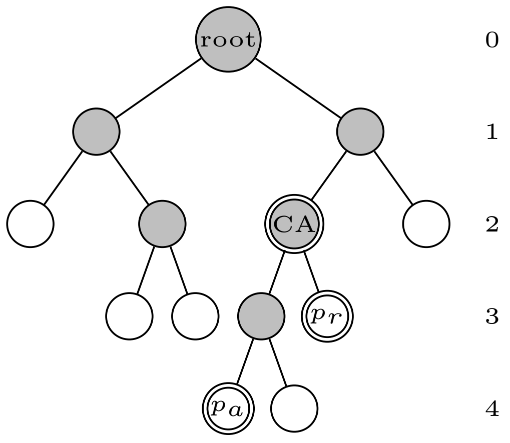


Figure 4.4 – *Example tree for illustration of hierarchcal recall: gray nodes are clusters, white nodes are single positions (leafs), numbers on the right indicate the depth of aligned nodes.*

As discussed above, any position given by the method for a profile may actually be a good recommendation even if different from the known one for that profile. Specifically, positions which are similar to the target one are usually equally good recommendations. Considering this, we can leverage the hierarchical clustering of positions computed in the previous step to evaluate how much a recommended position is close to the actual one. To this extent, we use the *hierarchical recall* (hR) measure proposed in (Silla and Freitas, 2011), based on the steps to be made from the root of the tree to get to the actual and predicted leaves. Specifically, given an actual position p_a and a single recommendation p_r, the hR is the ratio between the depth (i.e. the distance from the root, denoted here with Δ) of their deepest common ancestor (CA) and that of p_a. For example, the hR in the case depicted in Figure 4.4 is $2/4 = 0.5$, as the actual position is found at depth 4 and the deepest common ancestor between it and the recommended position is at depth 2. For k recommendations for the same profile, the maximum hR between them is considered; for the whole test set of profiles, the mean of these results is computed.

$$\mathrm{hR@}k = \frac{1}{|\mathcal{U}_{\mathrm{test}}|} \sum_{u \in \mathcal{U}_{\mathrm{test}}} \max_{r \in R_k(u)} \frac{\Delta(\mathrm{CA}(p(u), r))}{\Delta(p(u))} \tag{4.5}$$

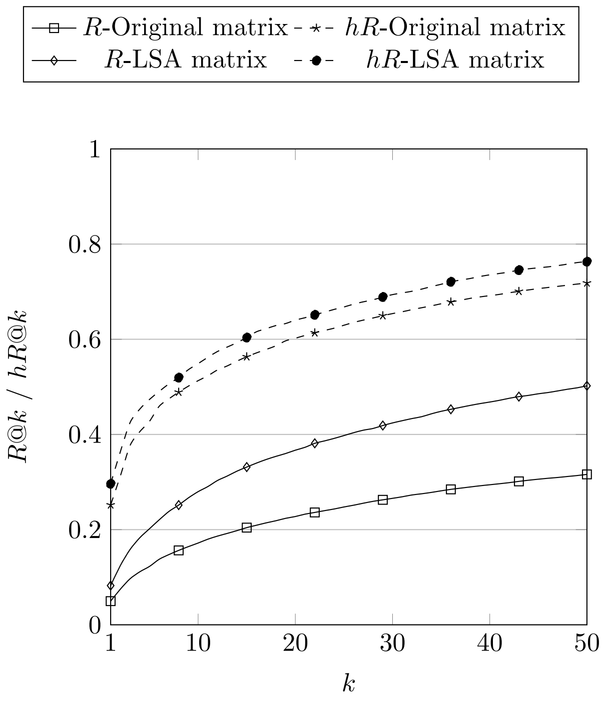


Figure 4.5 – *Trends of standard (solid lines) and hierarchical (dashed lines) recall for all values of k from 1 to 50.*

Recommendations of job positions for all test profiles have been computed using both the described approaches, i.e. representing positions with either the original co-occurrences matrix $\mathbf{C}$ or its low-rank approximation $\mathbf{C}'$ obtained from LSA. In both cases, we compared recommendations with known positions to compute both standard and hierarchical recall for all values of k from 1 to 50. Table 4.1 reports recall values for some specific values of k, while the plot in Figure 4.5 shows all the measurements.

A comparison between results obtained with the two matrices shows that the use of LSA always appears to be beneficial for the accuracy of the recommendations, as it improves the representation of positions according to their statistically estimated relatedness. Considering this, we focus the rest of the analysis on the results for the LSA matrix.

Obviously, the accuracy grows as the number k of recommendations to be returned is raised, because it is more likely to hit the exact position or a very similar one. However, a smaller set of good recommendations can often

Table 4.1 – *Standard and hierarchical recall for selected values of k.*

	Original matrix		LSA matrix	
k	R@k	hR@k	R@k	hR@k
1	0.050	0.252	0.082	0.296
5	0.123	0.428	0.202	0.474
10	0.172	0.513	0.280	0.549
20	0.228	0.602	0.367	0.640
50	0.316	0.718	0.502	0.763

be more valuable in practice than a larger one, which could more likely include improper elements. Looking at the standard recall, we see in practice that a single recommendation for each profile exactly matches the known occupation in 8.2% of the test cases. As the number of recommendations grows, the known position is more likely to be hit: this happens in about one case every five with $k = 5$, one every four with $k = 8$ and one every two with $k = 50$.

It is interesting to compare the standard recall to the hierarchical one for equal values of k: the latter is superior to the former with a consistent gap, ranging between 21% and 27%. This suggests that in many cases where the exact known position is not within the recommendations, at least one of them is anyway a very similar occupation.

Of course, this can be also observed by manually comparing recommendations to known positions. Table 4.2 shows, for some test profiles, both the actual known position and the recommended ones. It can be noted that, while the method fails at getting the exact occupation within the very top recommendations, these are nonetheless positions intuitively quite similar to it or even synonyms, which are in general equally valid and plausible for the given skills.

4.2.4 Discussions

This section presented a job recommendation system based on exploiting known co-occurrences between skills and potential job positions, which are elaborated by means of LSA to discover latent relationship between them. Furthermore, it also shows how the same data can be used to automatically build a folksonomy of job position by means of hierarchical clustering, in order to discover groups of related occupations.

Table 4.2 – *Example test profiles with skills, known positions and recommendations. English translations of Italian position names are reported in italic.*

Skills (alphabetical order)	Known position	Top 5 recommendations
blogging, e-commerce, facebook, marketing communications, marketing strategy, social media, social media marketing, social networking	responsabile customer service (*customer service manager*)	(1) marketing manager (2) sales manager (3) titolare (*owner*) (4) export manager (5) agente di commercio (*sales agent*)
apache, arch linux, centos, cisco nexus,cisco switches, dns, f5 bigip, netbackup, network administration, radware,red hat linux, sophos,trend micro, vmware, vmware esx, vsphere	it system administrator	(1) system administrator (2) it manager (3) network administrator (4) system engineer (5) senior system engineer
adults, mental health, psychology, psychotherapy	psychologist	(1) psicologa (*psychologist*, woman) (2) psicoterapeuta (*psychotherapist*) (3) psicologa psicoterapeuta (woman) (4) psicologo (*psychologist*, man) (5) psicologo psicoterapeuta (man)

The methods have been tested using a set of public profiles extracted from LinkedIn, naturally subject to noise and inconsistencies; we only applied a couple of trivial preprocessing steps to them. Despite this, we managed to extract a clustering where most of the groups are actually meaningful, composed of related positions. Concerning recommendations, a quantitative experimental evaluation trivially based on real job positions shows promising results, where in half of the cases the exact actual occupation of a person is within the top 50 recommended positions out of more than 2,000 possibilities. By leveraging the folksonomy of positions extracted above and looking at some specific cases we see that, even when the exact position name is not hit, homonyms and similar occupations are generally suggested.

Such a recommendation system can be potentially useful to individuals seeking for occupations where their abilities can effectively be endorsed, as well as to recruiters which have to evaluate the best candidates for specific positions. Notably, the method has no parameter to be set apart from the number of recommendations to be returned, so it is simple and ready to use in practice. One potential direction for further research would be to devise a method which fits even better to a recruitment system, for example by swapping the roles of profiles and positions, so that a set of recommended candidates can be obtained for a given occupation.

Chapter 5

Genomic Mining

5.1 Gene Ontology

Among available biological data, those that describe in structured form existing knowledge about structural and functional properties of biomolecular entities (mainly genes and their protein products) are extremely valuable. They are called *controlled biomolecular annotations* and each of them consists in the association of a biomolecular entity with a controlled term, which defines a structural or functional biological property and is part of a terminology or ontology; such association states that the biomolecular entity has the property that the controlled term defines.

Several biomedical terminologies and ontologies exist (Bodenreider, 2004; Smith et al., 2007); among them the Gene Ontology[1] (GO) is the most considerable one (Gene Ontology Consortium et al., 2001). It is composed of three sub-ontologies, overall including more than 40,000 concepts, which characterize species-independent Biological Processes (BP), Molecular Functions (MF) and Cellular Components (CC). These concepts are described through controlled terms and are hierarchically related, mainly through *IS_A* or *PART_OF* relationships, within a Directed Acyclic Graph (DAG), designed to capture orthogonal features of genes and proteins. In the GO DAG, each node represents a GO term (i.e. a concept) and each directed edge from a node A to a node B represents a relationship existing from a child term A to its parent term B.

[1]`http://geneontology.org/`

5.2 Biomolecular Annotations Prediction

Controlled biomolecular annotations are profitably leveraged for biomedical interpretation of biomolecular test results, extraction of novel information useful to formulate and validate biological hypotheses, and also discovering of new biomedical knowledge. They are particularly valuable for high-throughput and computationally intensive bioinformatics analyses. Several computational tools take advantage of these annotations, such as those for *annotation enrichment analysis* (e.g. (Masseroli et al., 2004; Masseroli, 2007; Al-Shahrour et al., 2007; Huang et al., 2009)) or *semantic similarity analysis* (Pesquita et al., 2009; Schlicker et al., 2010; Jain and Bader, 2010; Tedder et al., 2010; Falda et al., 2012) of genes and proteins; they strongly rely on coverage and quality of the available controlled annotations. However, existing controlled biomolecular annotations are accurate only in part, contain errors (particularly those only derived computationally, without human curator supervision) and are incomplete; several biological properties and functions of genes and gene protein products are still to be discovered, especially for recently studied organisms. Furthermore, the available annotations are largely derived computationally, and often they do not have an associated significance level; only a few of them are reviewed by human curators and represent highly reliable information. Annotation curation is crucial for annotation quality; yet, the curation process is very time-consuming. To help and accelerate it, availability of prioritized lists of computationally predicted annotations is greatly effective.

In this scenario, computational techniques able to reliably predict new biomolecular annotations with an associated likelihood value are paramount; towards this aim several different methods have been proposed. King et al. (King et al., 2003) suggested the use of *decision trees* and *Bayesian networks* to predict annotations by learning patterns from available annotations. Tao et al. (Tao et al., 2007) proposed a *k-nearest neighbour* (k-NN) classifier to associate a gene with new annotations common among its functionally nearest neighbour genes, where functional distance between genes is computed according to the semantic similarity of the GO terms that annotate the genes. *Hidden Markov Model* (HMM) techniques were also used to model gene function evolution (Mi et al., 2013) or predict gene function from sequential gene expression data (Deng and Ali, 2004). *Support Vector Machine* (SVM) classifiers are also common in gene function prediction. Minneci and colleagues (Minneci et al., 2013) leveraged them to predict GO annotations for several eukaryotic protein sequences, whereas Mitsakakis et al. (Mitsakakis et al., 2013) used them to predict potential functions for previously un-annotated *Drosophila melanogaster* genes, through the analysis of a large dataset from gene expression microarray experiments.

Also biological *network analysis* is often used to predict gene functions. Warde-Farley and colleagues (Warde-Farley et al., 2010) developed GeneMANIA, a server for gene function prediction, where query gene sets are represented as networks with an associated weight, which is based on how well connected the genes in the query set are to each other compared with their connectivity to non-query genes. Differently, Li et al. (Li et al., 2007) used a kernel-based learning method and proposed a labeled graph kernel which is able to predict functions of individual genes on the basis of the function distributions in their associated gene interaction networks.

Using a latent semantic approach and basic linear algebra, Khatri and colleagues (Khatri et al., 2005; Done et al., 2010) proposed a prediction algorithm based on the *Singular Value Decomposition* (SVD) method of the gene-to-term annotation matrix; this is implicitly based on the count of co-occurrences between pairs of annotation terms in the available annotation dataset. Masseroli and colleagues enhanced this algorithm (Masseroli et al., 2011), by including gene clustering based on gene functional similarity computed on the GO annotations of the genes; then, they further extended it by automatically choosing the best SVD truncation level, according to the evaluated dataset (Chicco and Masseroli, 2013, 9). The SVD has also been used with annotation *weighting schemes*, built upon the gene and term frequencies (Done et al., 2010, 2007; Pinoli et al., 2014a). Based on simple matrix decomposition operations, these methods are independent of both the considered organism and term vocabulary used in the annotation set; yet, obtained results showed limited accuracy.

Other more sophisticated latent semantic analysis techniques, mainly related to *Latent Semantic Indexing* (LSI) (Dumais et al., 1988) which was formerly used for Natural Language Processing, have been suggested to predict biomolecular annotations on the basis of available annotations; they include the *probabilistic Latent Semantic Analysis* (pLSA) (Hofmann, 1999), which leverages the latent model of a set of annotation terms to increase robustness of annotation prediction results. In a previous work, we used this technique (Masseroli et al., 2012), also enhanced with weighting schemes (Pinoli et al., 2013, 92), and showed that it improves the SVD method of Khatri and colleagues (Khatri et al., 2005). Topic modeling has been leveraged also through the use of the *Latent Dirichlet Allocation* (LDA) algorithm (Blei et al., 2003). Bicego et al. (Bicego et al., 2010) and Perina et al. (Perina et al., 2010) applied LDA to separate gene expression microarray data in clusters. Lately, we took advantage of the LDA algorithm associated with the Gibbs sampling (Griffiths, 2002; Casella and George, 1992; Porteous et al., 2008) to predict gene annotations to GO terms (Pinoli et al., 2014b). These advanced techniques greatly overcome those based on linear algebra, but the underlying model complexity and the training procedure slowness make them inappropriate when the evaluated dataset size increases.

Recently, additional supervised methods were proposed also for gene annotation prediction. Cheng et al. (Cheng et al., 2014) formulated the prediction of gene functions as a multi-label top-down classification problem. They developed a method using the hierarchical relationships in the GO structure to mitigate the quantitative difference between positive and negative training samples; yet, they obtained only poor accuracy. Additionally, Stojanova et al. (Stojanova et al., 2013) considered that relations between genes generate autocorrelation in gene annotations and violate the assumption that annotations are independently and identically distributed; by explicitly considering this autocorrelation they obtained better predictive accuracy.

Novel gene annotations were also inferred by leveraging multiple data types or sources, also from different species. A general Bayesian framework was developed by Troyanskaya and colleagues (Troyanskaya et al., 2003) to integrate heterogeneous types of high-throughput biological data, and was applied for the prediction of *Saccharomyces cerevisiae* gene functions. Barutcuoglu et al. (Barutcuoglu et al., 2006) used gene expression levels from microarray experiments to train a SVM classifier for each gene annotation to a GO term; then, they enforced consistency among predicted annotation terms by means of a Bayesian network mapped on the GO structure. Conversely, Raychaudhuri et al. (Raychaudhuri et al., 2002) and Pérez et al. (Pérez et al., 2004) used text mining techniques to extract form the literature gene associated keywords which are then mapped to GO concepts. These approaches provide improved results with respect to similar methods applied on a single data type; yet, they require a preparatory data integration step which adds complexity, reduces flexibility and slows the prediction process.

Thus, previously proposed methods for biomolecular annotation prediction either are general and flexible, but provide only limited accuracy mainly due to the simple model used, or improve prediction performance by either leveraging a complex integrative analytical framework, which often is difficult and time consuming to be properly set up, or adopting a more complex model, which in turn significantly slows the prediction process particularly in the usual case of many data to be evaluated.

5.3 Discovering New Gene Functionalities from Random Perturbations of Known Gene Ontological Annotations

In this section a novel representation of the annotation discovery problem is proposed, so as to enable applying supervised algorithms to predict Gene Ontology annotations of different organism genes. In order to use supervised algorithms despite labeled data to train the prediction model are not available, is proposed a random perturbation method of the training set, which creates a new annotation matrix to be used to train the model to recognize new annotations. This approach has been presented in (Domeniconi et al., 2014a) and further analyzed in (Domeniconi et al., 2015a).

5.3.1 Genomic Annotation Datasets

In order to have easy access to subsequent versions of gene annotations to be used as input to the considered algorithms or to evaluate the results that they provide, we took advantage of the Genomic and Proteomic Data Warehouse (GPDW) (Canakoglu et al., 2012). In GPDW several controlled terminologies and ontologies, which describe genes and gene products related features, functionalities and phenotypes, are stored together with their numerous annotations to genes and proteins of many organisms. These data are retrieved from several well known biomolecular databases. In the context of developing and testing machine learning methods on genomic annotations, GPDW is a valuable source since it is quarterly updated and old versions are kept stored. We leveraged this feature in our method evaluation by considering differed versions of the GO annotations of the genes of three organisms. In GPDW they are available with additional information, including an *evidence code* that describes how reliable the annotation is.

We leveraged it by filtering out the less reliable annotations, i.e. those with *Inferred from Electronic Annotation* (*IEA*) evidence, from the datasets used for our evaluation.

In GPDW, as in any other biomolecular database, only the most specific controlled annotations of each gene are stored. This is because, when the controlled terms used for the annotation are organized into an ontology, as for the GO, biologists are asked to annotate each gene only to the most specific ontology terms representing each of the gene features. In this way, when a gene is annotated to a term, it is implicitly indirectly annotated also to all the more generic terms, i.e. all the ancestors of the feature terms involved in its direct annotations. This is called *annotation unfolding.*

All direct and indirect annotations of a set of genes can be represented by using binary matrices. Let $\mathcal{G}$ be the set of genes of a certain organism and $\mathcal{T}$ a set of feature terms. We define the annotation matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{G}|\times|\mathcal{T}|}$ as the matrix whose columns correspond to terms and rows to genes. For each gene $g \in \mathcal{G}$ and for each term $t \in \mathcal{T}$, the value of the $\mathbf{A}(g,t)$ entry of the annotation matrix is set according to the following rule:

$$\mathbf{A}(g,t) = \begin{cases} 1, & \text{if } g \text{ is annotated either to } t \\ & \text{or to any of } t \text{ descendants} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

Examples of two versions of these matrices are shown in Figure 5.1a and 5.1b, where $\mathbf{A}_1$ is an updated version of $\mathbf{A}_0$. Each GPDW update contains some number of new discovered annotations, namely new 1 in the matrix. Table 5.1 gives a quantitative description of the considered annotations.

5.3.2 Method

Data and Problem Modelling

Given a feature term t, we want to predict if a gene g is likely to be, or not to be, annotated to that term t, i.e. if the element $\mathbf{A}(g,t)$ of the annotation matrix is likely to be 1, or 0. This can be modelled as a supervised problem, in which the predicted class is a term, i.e. a column of the matrix, that can be 0 or 1 according to the presence or absence of annotation between the gene and the term, while all other annotations of the gene represent the features of the record, as in Figure 5.1c. Considering that predictions

Table 5.1 – *Quantitative characteristics of the nine considered annotation datasets. Figures refer to the sum of direct and indirect annotations not inferred from electronic annotation, i.e. without IEA evidence code.*

	Gallus gallus			*Bos taurus*			*Danio rerio*		
	CC	MF	BP	CC	MF	BP	CC	MF	BP
# genes	260	309	275	497	540	512	430	699	1,528
# terms	123	134	610	207	226	1,023	131	261	1,176
# ann. 2009	3,442	1,927	8,709	7,658	3,559	18,146	4,813	4,826	38,399
# ann. 2013	3,968	2,507	10,827	9,878	5,723	24,735	5,496	6,735	58,040
Δ annotations between GPDW versions									
#Δ ann.	526	580	2,118	2,220	2,164	6,589	683	1,909	19,641
%Δ ann.	15.28	30.10	24.32	29.00	60.80	36.31	14.19	39.56	51.15

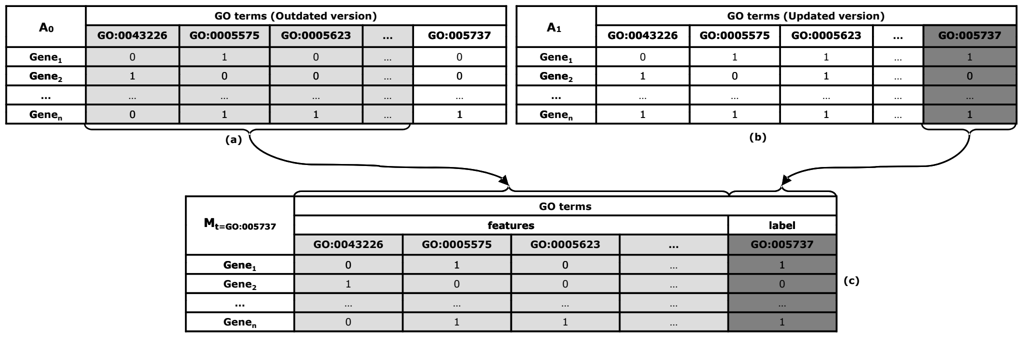


Figure 5.1 – *Illustrative diagram of the data representation. The data set (c) is created with an older annotation version* $\boldsymbol{A}_0$ *(a) for the features and an uptdated version* $\boldsymbol{A}_1$ *(b) for the labels.*

must be made for all the terms $t \in \mathcal{T}$, i.e. all the columns of the matrix, the problem can be modeled as a supervised multi-label classification, with the difference that we do not have a distinct set of features and labels, but we have a set of terms that are both classes and features. To address this problem, we use the most common approach in the literature, i.e. transform it into a set of binary classification problems, which can then be handled using single-class classifiers. Henceforth, for simplicity of exposition, we will refer to a single supervised task concerning the discovery of a new annotation of the gene g to the term t (for instance the term *GO:005737* in Figure 5.1), which is then repeated iteratively for all other genes and terms.

Let's now see how to assign a label to each instance of the data model.

Given an annotation matrix, our proposal is to use as input a version of the matrix with less annotations (referred as outdated matrix, since it may resemble an outdated annotation dataset version); then, to derive from such input matrix the features of the data model, and consider as label of each record the presence or absence of an annotation in a more complete matrix (referred as updated matrix, since it may resemble a newer annotation dataset version). This representation is sketched in Figure 5.1. Given the feature term t considered for the prediction, called *class-term*, the representation of the data is created by taking as features, for each gene, all the annotations to all the other terms in an outdated version of the matrix $\mathbf{A}_0$, while the label is given by the value of the class-term in the updated version of the matrix $\mathbf{A}_1$. Henceforth, we refer to this representation matrix as $\mathbf{M}_t$, where t is the class-term of the model.

This data representation is exactly the same as that of a supervised classification problem represented in a Vector Space Model. Thus, a classic supervised task could be envisaged by subdividing this new matrix $\mathbf{M}_t$ horizontally and using a part of the genes to train the model and the remaining part to test it. In this domain, however, this approach is not applicable because it implies the availability of at least the part of the updated matrix to train the model, but new datasets are only released as a whole and not partially. Thus, the purpose is to predict which annotations are missing in the entire matrix, rather than on some part of it. The data representation matrix $\mathbf{M}_t$ requires information from two different annotation dataset versions. Thus, since the aim is to make predictions over the entire dataset, to train the model we use a matrix $\mathbf{M}_t^{train}$ that is created by using the information from both the latest version currently available at training time, i.e. $\mathbf{A}_1$, and an older version of the matrix with missing annotations, i.e. $\mathbf{A}_0$. With this two different versions of the matrices, the training set is created by using the features derived from the outdated version $\mathbf{A}_0$ and the labels from the updated one $\mathbf{A}_1$. Then, the validation of the classification model has to be made by discovering new annotations, missing in the current state of the matrix. Therefore, the features regarding the current version $\mathbf{A}_1$ and labeled with the values of a future updated matrix $\mathbf{A}_2$ are used to create the validation matrix $\mathbf{M}_t^{validation}$. The training and validation data representation process is sketched in Figure 5.2.

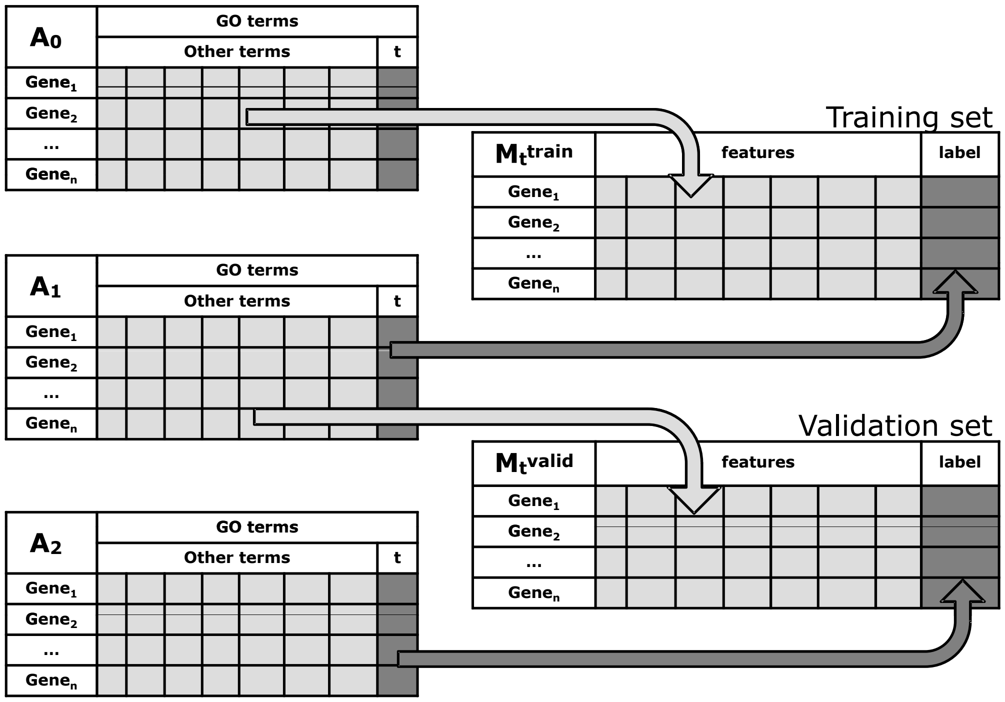


Figure 5.2 – *Illustrative diagram of the dataset representation for the prediction model of the annotations to a term t. The training set ($\boldsymbol{M}^{train}$) is created with an older annotation version $\boldsymbol{A}_0$ for the features and the current annotation version $\boldsymbol{A}_1$ for the labels. Similarly, the validation set ($\boldsymbol{M}^{valid}$) is created using $\boldsymbol{A}_1$ and a future updated annotation matrix $\boldsymbol{A}_2$.*

Random Perturbation

The supervised problem modelling described in the previous subsection requires, at training time, two versions of the annotation matrix to create the supervised model, i.e. $\mathbf{A}_0$ and $\mathbf{A}_1$. However, biologists typically have available only the most updated version of the annotation matrix, not keeping stored the outdated versions for space reasons, given the large amount of data. Thus, with reference to Figure 5.1, there is available only one version of the matrix, i.e. only the current version $\mathbf{A}_1$, with which the training data representation $\mathbf{M}_t^{train}$ is created.

To overcome the problem just mentioned, we start from the observation that also the input matrix $\mathbf{A}_1$ contains missing annotations. Therefore, we could use only this matrix to obtain the representation $\mathbf{M}_t$, assuming $\mathbf{A}_0 = \mathbf{A}_1$. However, the classification model will have to discover new gene-term annotations starting from an outdated matrix; thus, it will be more effective if it is trained with a training set in which the features are taken from an outdated matrix, with a greater number of missing annotations than the matrix version from which the labels of the instances are obtained. If we consider that the annotations of genes to features are discovered by teams of biologists that work independently from each other, a reasonable hypothesis is that the new annotations discovered by the entire scientific community, on the whole, do not have any kind of bond or rule. This should be equivalent to a random process of discovery of new annotations. Such considerations led to our thesis that new gene annotations can be better discovered by artificially increasing the number of missing annotations in the input matrix $\mathbf{A}_0$. Since, as mentioned, usually only the input matrix $\mathbf{A}_1$ is available, this can be achieved by randomly deleting known annotations in the matrix $\mathbf{A}_1$ to obtain a new matrix $\mathbf{A}_0$ artificially perturbed.

Thus, to get the data to train the classification model, we propose to randomly perturb the matrix $\mathbf{A}_1$ to create a new matrix $\mathbf{A}_0$, in which some annotations are eliminated with a probability p. In this way we obtain the matrix $\mathbf{A}_0 = \textit{random\_perturbation}(\mathbf{A}_1, p)$. Formally, for each gene g and term t, the perturbation is done as follows:

$$\mathbf{A}_0(g,t) = \begin{cases} 0 & \text{if } \mathbf{A}_1(g,t) = 1 \wedge \textit{random} \leqslant p \\ 1 & \text{if } \mathbf{A}_1(g,t) = 1 \wedge \textit{random} > p \\ 0 & \text{if } \mathbf{A}_1(g,t) = 0 \end{cases} \tag{5.2}$$

Once the perturbed matrix $\mathbf{A}_0$ is generated, to ensure its correctness with respect to the unfolding of the annotations, the matrix $\mathbf{A}_0$ is corrected by switching to 0 also all the annotations to the same gene of all the descendants of the ontological terms with modified gene annotation; we call this process *perturbation unfolding*. It is important to note that, depending on this correction, the percentage of the actual modified annotations of the matrix $\mathbf{A}_0$ will hence be greater than the percentage derived from p. The overall data representation process is the same as that shown in Figure 5.2, with the difference that the matrix $\mathbf{A}_0$ is created by perturbing randomly $\mathbf{A}_1$.

Considering the annotation unfolding in the GO, in order to avoid trivial predictions (i.e. 1 if a child is 1), in the set of features of the dataset $\mathbf{M}_t$ all the descendants or ancestors of the term t are not taken into consideration. Once created the training matrix $\mathbf{M}_t^{train}$, we can use any supervised algorithm, capable of returning a probability distribution, to train the prediction model and then validate it with $\mathbf{M}_t^{validation}$. The predicition model provides a probability distribution $pd(g,t)$, called *likelihood*, concerning the presence of an annotation of the gene g to the term t. To provide predictions of only new annotations, only those annotations that were missing in the outdated version of the matrix are taken into account. The supervised process described above is repeated for all the terms $t \in \mathcal{T}$, giving as final output a list of predictions of new gene annotations ordered according to their likelihood; the illustrated annotation discovery workflow is sketched in Figure 5.3.

Term Weighting

To increase the associative power of the gene-term matrix, we can give a weight to each existing association between genes and terms. This approach is similar to a classic term weighting in information retrieval (Section 3.2.4). Some weighting schemes have already been appliated to the prediction of genomic annotations (Pinoli et al., 2014a); here we tested these and other different schemes from information retrieval and data classification realms, in order to weigh each known annotation in the representation matrix $\mathbf{M}_\mathrm{t}$.

Fixed the *class-term* t_c of the representation matrix $\mathbf{M}_\mathrm{t}$, for each feature term $t \in \mathcal{T} : t \neq t_c$ four elements (A, B, C and D) shall be defined in order to describe the term weighting schemes: A denotes the number of genes associated both with t_c and t; B denotes the number of genes associated with t_c and not with t; C denotes the number of genes associated with t and not with t_c and D denotes the number of genes associated neither with t_c or t. The sum of all the genes is denoted with $N = A + B + C + D = |\mathcal{G}|$.

As already discussed, a term weighting scheme is generally based on three factors: i) *term frequency factor* or local weight; ii) *collection frequency factor* or global weight; iii) *normalization factor*. The *term frequency factor* measures how important a feature, namely an ontology term, is to a certain gene. For each gene g and feature term t, it can be expressed as $tf(g,t) = 1 + M$, where M is the number of descendant terms of t which are associated with the gene g, both directly or indirectly (i.e. derived from the unfolding procedure). Considering that this tf is measured in a different way than in the standard information retrieval methods, we also consider the case in which the local factor consists in a simple binary value, regarding the presence or the absence of the association between t and g.

The *collection frequency factor* may be taken from virtually all proposed weighting schemes in information retrieval or data mining. An interpretation of the common *idf* (Sparck Jones, 1972) is the igf (inverse gene frequency, (Pinoli et al., 2014a)); for each term t its value is:

$$igf(t) = \ln \frac{|\mathcal{G}|}{|\text{genes annotated to } t|} \equiv \log\left(\frac{N}{A+C}\right) \tag{5.3}$$

The combination of these two measures, *term* and *collection frequency factors*, with the *normalization factor* generates several possible weighting schemes ((Done et al., 2007)). The contribution of seven of these generated schemes to the gene annotation prediction using an unsupervised method is studied in ((Pinoli et al., 2014a)), where a substantial improvement is shown by using some of them. Out of all the schemes analyzed in that work, we focus on the combination regarding no-transformation in the tf factor and the maximum, cosine and none normalization (i.e. the schemes named NTM, NTC and NTN). In this work, we refer to these three schemes as $tf.igf^M$, $tf.igf^C$ and $tf.igf^N$, respectively. In our experiments, we also tested the igf alone, using only a binary value (*bin*) as term frequency factor; we refer to these schemes as igf^M, igf^C and igf^N. In addition, we tested also the two term frequency measures, i.e. tf and *bin*, alone, without a collection frequency factor.

Furthermore, we use some weighting schemes derived directly from the information retrieval ((Debole and Sebastiani, 2003)), such as χ^2 or ig (information gain). Finally, we also tested the *relevance frequency* (rf) scheme proposed by ((Lan et al., 2009)) for the text classification task. The rf of a term t is based on the idea that the higher the concentration of genes associated both with t and the *class-term*, the greater the contribution of t in the prediction model.

Likelihood Correction

As shown above, the output of the supervised model is a list of predicted annotations, each one with a likelihood degree. According to the hierarchical structure of GO, when a gene is annotated to an ontological term, it must be also annotated to all the ancestors of that term; this constraint is also known as *True Path Rule* (Tanoue et al., 2002). The supervised classifier, however, provides a likelihood for each gene annotation regardless of the predictions of the annotation of other GO terms to the same gene. This can result in possible cases of anomalies in which a gene shall be annotated to a term, but not to one or more of its ancestor terms, thus violating the True Path Rule. To obtain a likelihood that takes into account the hierarchy of the terms, once obtained the likelihood of each gene-term association, we proceed as follows:

1. For each novel gene-term annotation, to the probability given by the model we add the average of all the probabilities of the novel annotations of the gene to all the ancestors of the term. Note that, since the classification model provides in output a probability distribution ranging between 0 and 1, the hierarchical likelihood of each gene-term annotation shall be between 0 and 2, as follows:

$$pd^H(g,t) = \frac{\sum\limits_{t_a \in ancestors(t)} pd(g,t_a)}{|ancestors(t)|} + pd(g,t) \tag{5.4}$$

2. Once the likelihood is made hierarchical, the correction of the possible anomalies regarding the True Path Rule is taken into account. An iterative process is carried on from the leaf terms to the root term of the hierarchy, upgrading each likelihood with the maximum likelihood value of the descendant terms, as follows:

$$l(g,t) = \max\{pd^H(g,t), \max_{t_c \in children(t)} \{pd^H(g,t_c)\}\} \tag{5.5}$$

   In such a way, for each ontology term, the likelihood of a gene to be annotated to that term is always greater than or equal to the likelihood of the gene to be annotated to the term descendants.

Evaluation

Effectiveness of supervised methods in discovering new gene annotations based on the available ones can be fully evaluated by using gold standard

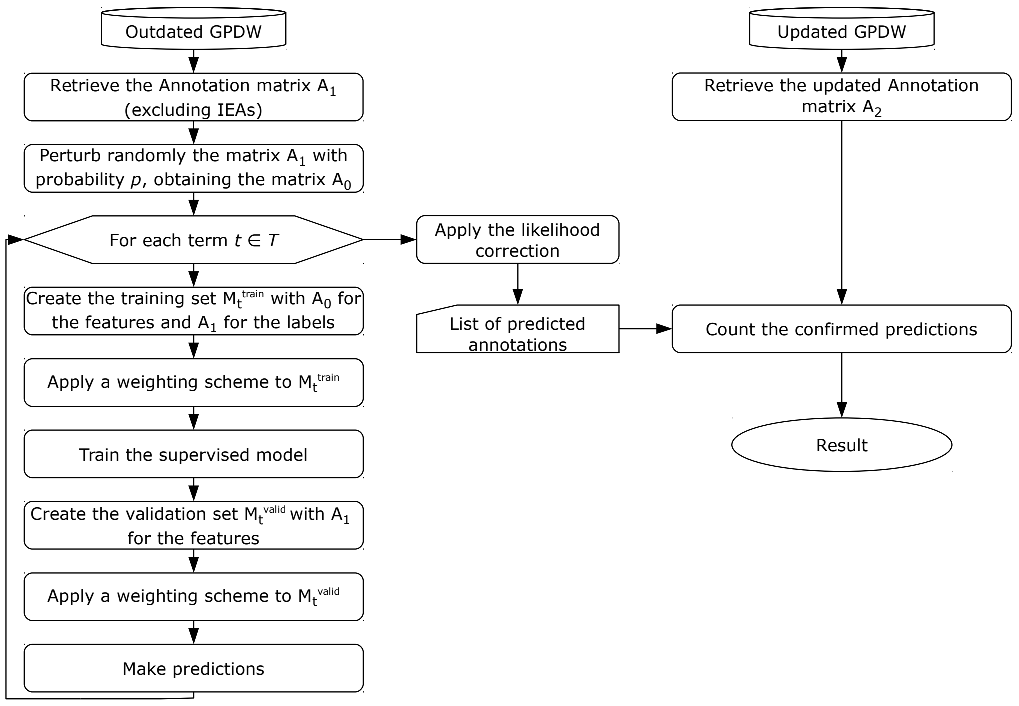

Outdated GPDW
Retrieve the Annotation matrix A_1 (excluding IEAs)
Perturb randomly the matrix A_1 with probability p, obtaining the matrix A_0
For each term $t \in T$
Create the training set M_t^{train} with A_0 for the features and A_1 for the labels
Apply a weighting scheme to M_t^{train}
Train the supervised model
Create the validation set M_t^{valid} with A_1 for the features
Apply a weighting scheme to M_t^{valid}
Make predictions
Apply the likelihood correction
List of predicted annotations
Updated GPDW
Retrieve the updated Annotation matrix A_2
Count the confirmed predictions
Result

Figure 5.3 – *Workflow of the training and validation processes.*

datasets; yet, unfortunately, such biological datasets generally are not available, and are difficult and very time-consuming to be created in a useful significant size. An alternative option is to consider two gene annotation versions, available at a given temporal distance, and use the older one to train the model and generate the annotation predictions, and the newer one to evaluate the model predictions; in fact, generally, available gene annotations of an organism increase over time while the organism is studied. It is important to note that, although much more feasible, this option allows only a lower estimate of the method efficacy; as less the temporal distance between the available gene annotation versions is, as more predictions could be correct, but not found confirmed simply because not yet included in the newer gene annotations available.

In our experiments we tested the effectiveness of supervised models in discovering new functional gene annotations from the available annotations. Since the proposed method is applicable to any supervised algorithm that returns a probability distribution, we tested different types of existing algorithms in order to measure their effectiveness, in particular: *Support Vector Machines*, *nearest neighbors*, *decision trees*, *logistic regressions* and *naive bayes*, using the implementations provided by Weka[2] in its 3.7.9 version. In the experiments we tested the Weka classifiers: *IBk* (with $k = 3$), *J48*, *Logistic*, *Naive Bayes* (*NB*), *Random Forest* (*RF*) and *SMO*. For each algorithm we used the default parameter settings provided by Weka; no tuning of parameters has been done for time reasons.

We measured the effectiveness of the predictions in the same way it was done in (Pinoli et al., 2014a), in order to be able to directly compare our results with those in that work; the overall procedure was as follows.

1. We extracted the input annotations from an outdated version of the GPDW (July 2009), excluding from those annotations the ones less reliable, i.e. with IEA *evidence* code.

2. We randomly perturbed the unfolded annotation matrix to get a modified version of it, with some missing annotations.

3. By running the prediction algorithm, we got a list of predicted annotations ordered by their confidence value (i.e. their corresponding likelihood $l(g, t)$).

4. We selected the top P predictions (we use $P = 250$) and we counted how many of these P predictions were found confirmed in the updated version of the GPDW (May 2013 version), regardless their evidence code.

5. For each experiment, steps 2, 3, 4 were repeated 10 times by varying the random seed. The effectiveness of each experiment was determined by averaging the counts obtained in all the experiment repetitions.

We depict the training and validation procedure workflows in Figure 5.3.

5.3.3 Experimental Results

Table 5.2 shows the results obtained by varying the supervised algorithm used to train the prediction model, always using a fixed random perturbation probability $p = 0.05$ and without weighting the term associations, i.e.

[2]http://www.cs.waikato.ac.nz/ml/weka/.

Table 5.2 – *Validation results of the predictions obtained by varying the supervised algorithm used to build the prediction model. The results show, for each of the nine considered datasets, the amount of the top 250 predicted gene annotations that have been found confirmed in the updated GPDW version. The setup of these experiments was done with random perturbation of the training matrix with probability* $p = 0.05$ *and the binary term weighting scheme. The first column (SIM) reports the results obtained in ((Pinoli et al., 2014a)) with the SIM best configuration. Each result is reported as the average and corresponding standard deviation of 10 experiments repeated by changing the random perturbation seed. In bold the best result for each dataset.*

Dataset	SIM	IBk	J48	Logistic	NB	RF	SMO
Gallus g. - BP	**86**	58.6±20.2	47.2±4.7	32.7±6.8	25.4±4.4	52.7±12.1	28.7±9.3
Gallus g. - MF	24	58.0±5.6	**79.7**±12.7	40.0±10.4	14.2±1.6	54.4±9.6	50.7±14.3
Gallus g. - CC	50	**81.5**±8.2	73.4±8.5	31.9±6.4	23.5±3.7	55.2±11.3	29.6±4.0
Bos t. - BP	55	48.9±6.8	49.7±5.1	37.0±6.5	28.4±4.2	**62.4**±7.6	31.2±4.6
Bos t. - MF	28	58.2±4.4	**58.8**±10.5	27.5±4.3	15.7±2.9	57.5±11.2	36.9±4.4
Bos t. - CC	91	**112.0**±9.7	94.3±9.8	38.2±5.3	8.2±2.0	93.7±10.4	48.4±6.8
Danio r. - BP	35	**70.9**±15.9	59.8±6.1	31.0±4.8	25.2±3.3	58.1±5.1	16.6±2.3
Danio r. - MF	35	77.5±10.3	75.8±7.1	54.4±11.0	41.2±2.7	**83.1**±9.6	79.7±8.7
Danio r. - CC	44	81.5±8.5	69.3±8.7	27.6±7.6	26.2±6.6	**92.3**±11.0	30.2±6.6
Total	447	**647.1**	608.8	320.3	207.9	609.4	352.0

using the *binary* scheme. State of art methods ((Pinoli et al., 2014a)) reach a total of 447 correct predictions; Table 5.2 shows that, with the proposed method, 3 out of 6 of the tested algorithms outperform them. Obtained results are excellent if we consider that they are obtained without any tuning of the algorithm parameters; therefore there is margin to improve them with an appropriate tuning. According to the results in Table 5.2, we can infer that using the standard parameterization provided by Weka, the algorithm that obtains the best results is *IBk*, with an improvement of 44.8% compared with the results of ((Pinoli et al., 2014a)). *IBk* results also 6.2% better than *Random Forest* and 6.3% better than *J48*, the other two supervised algorithms that result better than the state of art.

Table 5.3 shows the results obtained by using the *IBk* algorithm with different term weighting schemes in the representation matrix. Differently from the work of Pinoli and colleagues ((Pinoli et al., 2014a)), where weighting schemes improved their method, from our results we can infer that using weighting schemes does not lead to an improvement of the prediction effectiveness. The comparisons between the weighting schemes considered

Table 5.3 – *Validation results of the predictions obtained using IBk as supervised algorithm, $p = 0.05$ as probability of perturbation and varying the term weighting scheme.*

Dataset	*BIN*	tf	$tf.igf^N$	$tf.igf^C$	$tf.igf^M$	igf^N	igf^C	igf^M	χ^2	ig	rf
G.g.-BP	58.6	52.5	48.0	46.4	51.2	58.6	62.0	**66.6**	47.0	50.2	47.2
G.g.-MF	58	53.0	57.6	58.4	53.2	51.6	**64.4**	60.8	62.6	64.0	61.4
G.g.-CC	**81.5**	55.6	48.2	42.2	58.4	61.0	45.4	50.8	68.0	69.6	67.2
B.t.-BP	48.9	61.8	52.8	34.4	48.0	45.8	**72.6**	65.0	39.0	40.2	56.6
B.t.-MF	58.2	39.0	38.4	48.0	38.6	73.0	60.6	64.0	72.8	**75.4**	73.6
B.t.-CC	112.0	90.2	93.8	72.6	80.4	93.0	103.4	90.0	101.8	103.4	**121.4**
D.r.-BP	**70.9**	68.6	63.4	59.9	61.2	70.6	67.2	69.2	69.3	68.3	67.6
D.r.-MF	**77.5**	45.2	57.2	46.0	23.6	60.8	55.4	53.0	57.2	52.4	62.2
D.r.-CC	81.5	74.2	30.6	49.2	40.4	61.4	77.6	60.8	75.2	73.6	**82.6**
Total	**647.1**	540.1	490.0	457.1	455.0	575.8	608.6	580.2	592.9	597.1	639.8

Table 5.4 – *Validation results of the predictions obtained using the IBk supervised algorithm, binary term weighting scheme and varying the probability p of random perturbation of the training matrix.*

Dataset	$p=0$	$p=0.05$	$p=0.10$	$p=0.15$	$p=0.20$	$p=0.25$	$p=0.30$
Gallus g. - BP	42	**58.6**±20.2	54.8±16.2	51.3±12.5	55.9±10.4	50.2±10.2	47.4±9.7
Gallus g. - MF	50	58±5.6	61.8±11	59.5±13	58.3±10.2	**63.6**±13.5	64.2±8.4
Gallus g. - CC	75	81.5±8.2	77.5±9.7	**82.2**±8.1	78.1±7.5	73.3±13.2	78.8±12
Bos t. - BP	43	48.9±6.8	51.7±10.1	50.4±8.4	**53.1**±9.6	52±12.5	52.2±15.4
Bos t. - MF	58	58.2±4.4	62.7±7.7	71.4±10.9	73±12.6	74.7±11.6	**77**±13
Bos t. - CC	108	112±9.7	114.3±11	118.6±13	118.1±13	**119**±13.1	116.7±22
Danio r. - BP	55	70.9±15.9	70.6±16.5	74.8±13.9	85.7±25.6	83.1±16.3	**90.6**±19.4
Danio r. - MF	76	**77.5**±10.3	72.5±7.1	67.7±10.1	62±7.6	58.4 ±8.7	51.4±15.1
Danio r. - CC	79	81.5±8.5	84.7±8.7	**90.7**±10	85.6±13.5	83.3±14.5	75.8±19.9
Total	586	647.1	650.6	666.6	**669.8**	661.6	654.1

provided fluctuating results, but, in general, the best scheme appears to be the *binary* one. We can note that the schemes with the tf factor do not achieve good results, as well as, although with better performance, the information retrieval classical supervised measures, namely χ^2 and ig; the best weighting scheme results the rf, which however is, in total, slightly worst than the *binary* one.

The proposed method introduces a new parameter: the probability p of the random perturbation of the training matrix. Table 5.4 shows the results obtained by varying this probability p and using the best supervised algorithm from Table 5.2, namely IBk, and the *binary* weighting scheme.

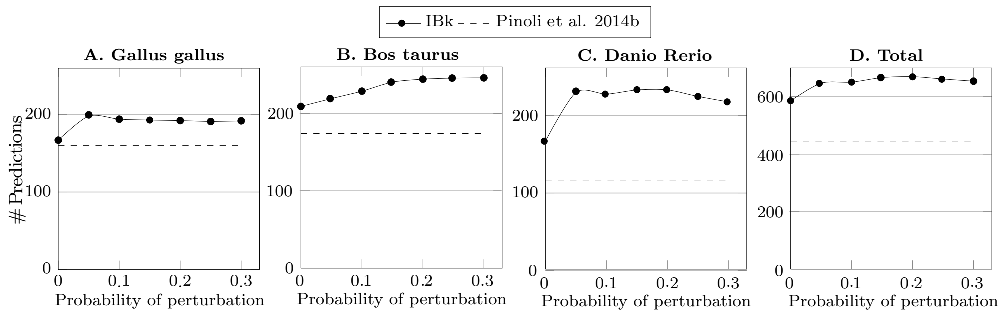


Figure 5.4 – *Validation results of the predictions obtained by varying the probability of perturbation p, compared with those obtained in ((Pinoli et al., 2014a)). The results show, for each organism in the A, B and C charts, the sum of the predicted annotations that have been found confirmed in the updated GPDW version of the three GO ontologies. The chart D shows the total values for all the organism.*

Table 5.4 results show that the best predictions are obtained with $p = 0.2$. Considering the *perturbation unfolding*, this p value leads to a perturbed matrix $\mathbf{A}_0$ with more than 20% of annotations less than in $\mathbf{A}_1$ (empirically they are about 30% less). Such percentage is very close to the average value of the variation of number of annotations between $\mathbf{A}_2$ and $\mathbf{A}_1$, i.e. 33.4%, notable in Table 5.1. Moreover, the probability p that gets the best results for each dataset seems to have a relationship with the dataset annotation variation between $\mathbf{A}_2$ and $\mathbf{A}_1$. This result leads to the conjectures that i) representing new annotations randomly leads to train a classifier able to predict the actual new annotations between two different annotation versions; ii) the more the amount of artificial missing annotations introduced in the training set is comparable to the actual missing annotations in the validation set, the more the predictions are accurate. Another result deducible from Table 5.4 is that using $p = 0$, namely the annotation matrix is not perturbed ($\mathbf{A}_0 = \mathbf{A}_1$), we get anyway good results, higher than those in ((Pinoli et al., 2014a)). This is important since it allows to avoid the parameter p and the tuning of the system for any considered dataset when not top performance is required. For a graphical view, the results discussed are also shown in Figure 5.4, grouped by organism. Our approach outperforms the best accuracy achieved in ((Pinoli et al., 2014a)) of 49.66%, in particular we obtain the highest improvement in large datasets, i.e. in the *Danio rerio* dataset there is an improvement of 104.56% of the correct annotations predicted.

5.3.4 Discussions

In this section we propose a method to discover new GO term annotations for genes of different organisms, based on available GO annotations of these genes. The idea is to train a model to recognize the presence of novel gene annotations using the obsolete annotation profile of the gene, labeling each term of an outdated annotation profile of a gene with a label taken from an updated version of it. This approach requires two different versions of the annotation matrix to build the training data representation. However, biologists typically have available only the most updated version of the gene annotation matrix. Given this constrain, we have proposed a method to overcome this lack; creating a different annotation matrix, representing an older version of the input one, by perturbing the known annotation matrix in order to randomly remove some of its annotations. This allows the use of supervised algorithms even in datasets without labels and the comparison with results obtained by unsupervised methods on the same originally unlabeled datasets.

Obtained results are very encouraging, since they show a great improvement compared with unsupervised techniques. Furthermore, these results could be even better with an appropriate tuning of the parameters of the supervised algorithms used; our purpose is to thoroughly investigate this aspect in the future. The extension, using weighted real values to represent gene-term associations, did not yield better results with respect to the binary value representation. Thus, we found that the computationally simpler scheme, namely the binary scheme, achieves results generally better than other more complex schemes.

From the obtained results we can see that, by increasing the number of perturbed (removed) annotations, the results improve, reaching a peak when the number of artificially missing annotations in the training set is comparable to the number of those in the validation set, i.e. when the variety of missing annotations has been fully mapped in the training set. Furthermore, it is noteworthy also the case where we do not perturb the training matrix, avoiding the tuning of the parameter p, which gets anyway good results.

5.4 Cross-Organism Learning Method to Discover New Gene Functionalities

In this section, we propose a novel cross-organisms learning approach to reliably predict new functionalities for the genes or proteins of an organism based on the known controlled annotations of the genes or proteins of another, evolutionarily related and better studied, organism. We leverage the representation of the annotation discovery problem and the random perturbation of the available controlled annotations, previously presented in Section 5.3. Taking advantage of the numerous gene annotations available for a well-studied organism, our cross-organisms learning method creates and trains better prediction models, which can then be applied to predict new gene annotations of a target organism.

5.4.1 Cross-Organism Annotation Data

For the development and validation of the proposed method, also in this case, we retrieved multiple gene annotation sets from the Genomic and Proteomic Data Warehouse (GPDW) (Canakoglu et al., 2011, 6) of the Genomic and Proteomic Knowledge Base (GPKB) (GPK, accessed on May 30, 2015).

To evaluate on a comparative genomic strategy the effectiveness of our cross-organism computational approach for reliably gene annotation prediction, from the GPDW we extracted the GO annotations there available for the genes of multiple eukaryotic organisms; we selected which of these organisms to consider for our approach evaluation on the basis of their number of gene GO annotations, annotated genes and annotation terms, as well as on their evolutionary distance. Towards this purpose, we took into account the evolutionary divergent eukaryotic species shown in Figure 5.5, which are considered in the Reactome pathway knowledge base project (Croft et al., 2014) for a similar orthology inference strategy. Such strategy aims at using available manually-curated human biomolecular reactions to electronically infer biomolecular reaction events in these evolutionary divergent eukaryotic species, for which high-quality comprehensive sequence data exist (Rea, accessed on May 30, 2015). Out of these species, we selected the *Homo sapiens*, the most studied organism with the highest number of gene GO annotations available, and other four evolutionarily related model

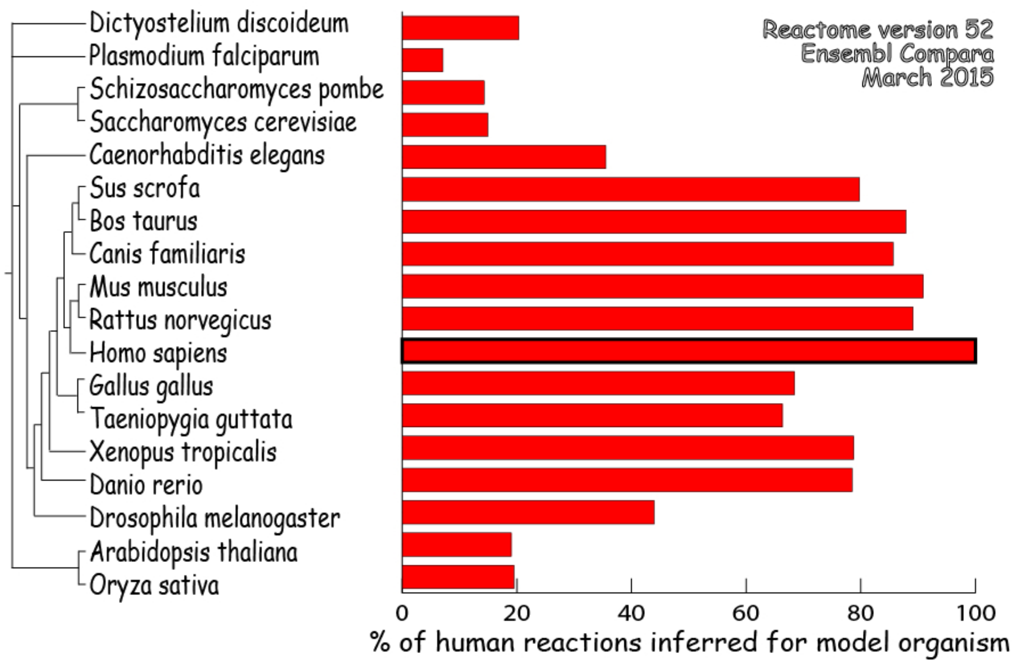


Figure 5.5 – *Phylogenetic dendrogram of evolutionarily divergent eukaryotic species (left) for which biomolecular reaction events have been electronically inferred in the Reactome pathway knowledge base project (Croft et al., 2014), using manually curated human reactions available. Success rate of such orthology inference is shown (right) as percentage of eligible reactions in the available Reactome human dataset for which an event can be inferred in the model organism (Rea, accessed on May 30, 2015).*

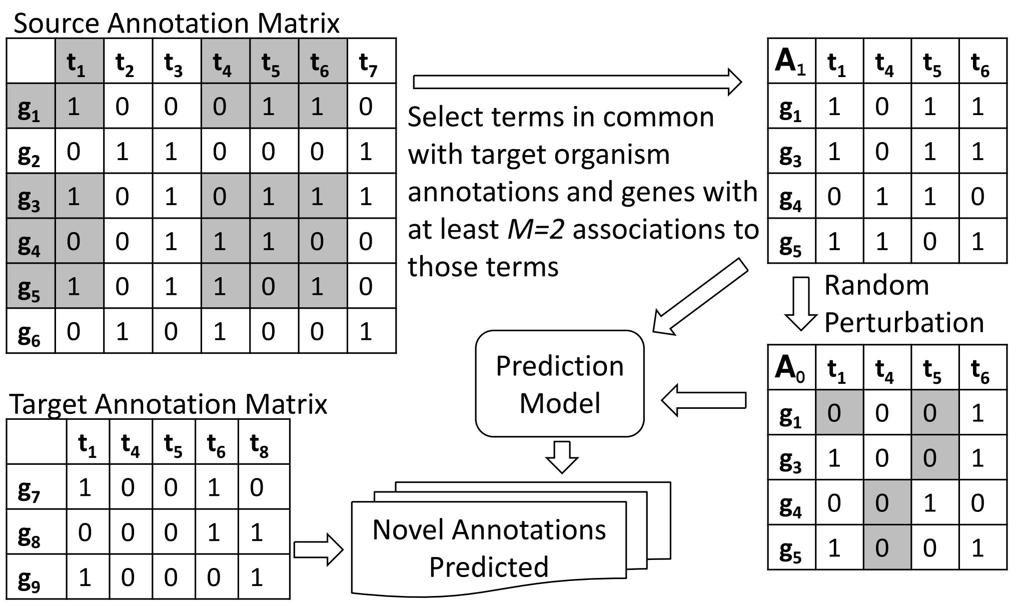
organisms: an evolutionary close organism with a high amount of gene GO annotations available (*Mus musculus*), an organism averagely distant evolutionarily (*Bos taurus*), another averagely distant organism (*Gallus gallus*), but much less studied, i.e. with much less gene GO annotations available, and an evolutionarily far related organism (*Dictyostelium discoideum*, a species of soil-living amoeba commonly referred as "slime mold"). Besides their different evolutionary distance, these organisms differ by the quantitative features of their gene GO annotations shown in Table 5.5.

To perform the defined evaluation procedure of our approach, we selected two temporally distant versions of the GO annotations available in the GPDW for the selected organism genes: an older version, as of July

Table 5.5 – *Counts of gene GO annotations (A) and their involved genes (G) and GO terms (T) used for the cross-organism approach evaluation. Only genes and GO terms shared between the July 2009 and March 2013 versions are counted. Both most specific and implicit gene GO annotations are counted. For the annotations of the July 2009 version (A_{09}), only the more reliable human curated annotations that were used for model training are reported; for the annotations of the March 2013 version, both curated (A_{13}) and total (At_{13}) annotations are reported.*

	Homo s.	*Mus m.*	*Bos t.*	*Gallus g.*	*Dict. d.*
G	9,937	9,265	646	321	1,762
T	3,322	3,366	749	403	1,016
A_{09}	345,259	319,402	21,305	8,846	65,421
A_{13}	353,679	606,239	26,194	11,339	63,621
At_{13}	955,341	826,033	47,237	17,744	118,695

2009, and a more recent one, as of March 2013. As most of the biomolecular databases, for each gene the GPDW stores only the available annotations to the most specific GO terms describing the gene functional features; all the other gene annotations to all the ancestor terms of these specific GO terms are left implicit and can be derived through the ontology structure. In building our annotation matrices we considered all gene GO annotations, including these implicit ones. Finally, we distinguished between more and less reliable annotations; this distinction has been based on the GO *evidence code* associated with each GO annotation and describing the procedure that produced the annotation. Each GO annotation can be associated with more than one evidence code, since there can be multiple procedures or assays that derived the annotation; we considered as less reliable all the GO annotations with only *Inferred from Electronic Annotation* (*IEA*) or *No biological Data available* (*ND*) evidence code, whereas we considered more reliable all the other GO annotations. We used the latter ones for the prediction model training and testing, whereas we used the newer available version (as of March 2013) of all the available annotations with any GO evidence code for the validation of the predicted annotations. The quantities of annotations and involved genes and terms for each organism and annotation version considered are listed in Table 5.5.


Source Annotation Matrix
Select terms in common with target organism annotations and genes with at least *M=2* associations to those terms
Random Perturbation
Prediction Model
Target Annotation Matrix
Novel Annotations Predicted

Source Annotation Matrix

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
g_1	1	0	0	0	1	1	0
g_2	0	1	1	0	0	0	1
g_3	1	0	1	0	1	1	1
g_4	0	0	1	1	1	0	0
g_5	1	0	1	1	0	1	0
g_6	0	1	0	1	0	0	1

A_1	t_1	t_4	t_5	t_6
g_1	1	0	1	1
g_3	1	0	1	1
g_4	0	1	1	0
g_5	1	1	0	1

A_0	t_1	t_4	t_5	t_6
g_1	0	0	0	1
g_3	1	0	0	1
g_4	0	0	1	0
g_5	1	0	0	1

Target Annotation Matrix

	t_1	t_4	t_5	t_6	t_8
g_7	1	0	0	1	0
g_8	0	0	0	1	1
g_9	1	0	0	0	1

Figure 5.6 – *Workflow for novel annotation predictions using the cross-organism learning method.*

5.4.2 Method

The Gene Ontology unifies the representation of gene and gene product features across species, i.e. GO terms describe gene functions independently from the organism considered. This lets assuming that the co-annotations of genes to GO terms depend only on the terms themselves and not on the gene organism. Consequently, given a supervised model able to discover relationships among gene annotations to terms of the GO, or of any other ontology with cross-species characteristics equivalent to those of the GO, it is conceivable to train such model on a largely studied organism (*source*), and then apply the trained model to predict novel gene annotations of a scarcely analyzed organism (*target*). With such a training, the used supervised algorithm can leverage a greater number of genes with a larger number of annotations already known, i.e. it can take advantage of a bigger and more dense training annotation matrix than that it would use by considering only the annotation matrix of the target organism. On the other hand, in this way the obtained model is not able to predict annotations to terms that are specific for the target organism and are not used for the source organism, since it can only handle annotations to terms used in the source organism annotations.

In order to create a cross-organism learning model, first it is necessary to select the genes and terms useful to predict novel gene annotations for the target organism. Let $\mathcal{T}_S$ and $\mathcal{T}_T$ be the two sets of feature terms used in the available gene annotations for the source and target organisms, respectively. We create the prediction model using the set $\mathcal{T}$ of those terms that the two organism annotations have in common, i.e. $\mathcal{T} = \mathcal{T}_S \cap \mathcal{T}_T$. Then, we also need to select the source genes to be used to train the model; in fact, there can be genes of the source organism that have no annotations to the terms in $\mathcal{T}$, thus being useless for the model training. Hence, we introduce a new parameter, called M, to represent the minimum number of annotations to the terms in $\mathcal{T}$ that any source gene needs to have in order to be used in the training process.

Defined the composition of the initial annotation matrix, as described in the Section 5.3, we can perturb it to create the training set and then proceed with the same prediction method used for a single organism; in principle, any supervised prediction algorithm can be used to this end. In Figure 5.6 we show a simple example of the described process: first, we select the set of common terms between source and target organisms, namely $\mathcal{T} = \{t_1, t_4, t_5, t_6\}$, and the source genes that have at least $M = 2$ annotations to the terms in $\mathcal{T}$. Once created the first training annotation matrix $\mathbf{A}_1$, an artificial older version $\mathbf{A}_0$ of it is created by randomly perturbing it; thus, we have the two matrices needed to train the prediction model, which is then applied to the genes of the target organism in order to predict novel annotations for them. If two original annotation versions of the source organism genes are available, they can be used to create the training annotation matrices, without needing any perturbation.

Evaluation

To measure the effectiveness of the defined approach, we considered the accuracy of the gene annotations predicted with likelihood greater than a threshold ρ (Ac_ρ). We calculated it as the number of gene annotations predicted for the target organism with likelihood greater than ρ that are found confirmed in an available newer version of the target organism's gene annotations, divided by the total number of target organism's gene annotations predicted with likelihood greater than ρ.

The overall evaluation procedure that we defined for our cross-organism learning method is as follows:

1. Create the annotation matrices $\mathbf{A}_{s_1}$ and $\mathbf{A}_{t_1}$ for the source and target organisms, respectively, by considering only all their reliable gene annotations (e.g. excluding their less reliable inferred electronic annotations) available at a given time T_1.

2. Select the set of terms $\mathcal{T}$ in common between such source and target organism annotations, avoiding considering the source genes with less than M annotations to the terms in the target organism annotations.

3. Create a second, more complete, annotation matrix $\mathbf{A}_{t_2}$ for the target organism, by considering a newer version of all its gene annotations (both more and less reliable) available at a time T_2, with $T_2 > T_1$.

4. Create a second, less complete, annotation matrix $\mathbf{A}_{s_0}$ for the source organism, by considering an older version of only all its reliable gene annotations available at a time T_0, with $T_0 < T_1$.

5. If such older annotation version is not available, randomly perturb the source annotation matrix $\mathbf{A}_{s_1}$ to get a modified version $\mathbf{A}_{s_0}$ of it with some missing annotations.

6. Train a supervised prediction model for each term in $\mathcal{T}$, by using the two annotation matrices $\mathbf{A}_{s_0}$ and $\mathbf{A}_{s_1}$ of the source organism.

7. Run the prediction algorithm on the annotation matrix $\mathbf{A}_{t_1}$ to get a list of predicted gene annotations for the target organism, ordered by their confidence value, i.e. by their likelihood.

8. Evaluate the obtained predictions by checking whether they are found confirmed in the updated version of the target organism's annotation matrix $\mathbf{A}_{t_2}$, and compute the Ac_ρ measure.

9. For each experiment with a perturbed training matrix $\mathbf{A}_{s_0}$, to avoid possible biases due to the random perturbation seed, repeat steps 5, 6, 7 and 8 multiple times (e.g. 5 times) by varying the random seed of the perturbation; then, determine the effectiveness of each experiment by averaging the Ac_ρ obtained in each experiment repetition.

Please note that, to discover new gene annotations for the target organism, the defined cross-organism approach strictly requires the availability of only one version of the source and target organisms' gene annotations; only the defined validation procedure requires the availability of an additional version of the considered annotations for the target organism, in order to automatically evaluate the accuracy of the predictions.

5.4.3 Experimental Results

In this Section we illustrate the performance results of our approach. First, we compare the performance obtained for single organism vs. cross-organism annotation prediction; following, we describe the assessment results of the usefulness of the perturbation method to better train the supervised algorithms; we then show the evaluation results of tuning the few parameters that our approach requires, i.e. the supervised algorithm to be used and the minimum number of shared annotation terms between the source and target organism; finally, as an example, we illustrate the full results obtained for the prediction of the GO Cellular Component annotations of a gene.

Single Organism vs. Cross-Organism Approach

We first evaluated the effective utility of the cross-organism learning method with respect to the single organism one. Results obtained in previous section show that using two equal annotation matrices to train the model (i.e. with $\mathbf{A}_0 = \mathbf{A}_1$) provides good effectiveness, avoiding the tuning of the perturbation parameter; accordingly, to obtain a first comparison between the single organism and cross-organism approaches, without perturbing the training annotation matrices we trained multiple prediction models with the gene annotations of the *Homo sapiens* and the other four selected organisms (*Mus musculus*, *Bos taurus*, *Gallus gallus* and *Dictyostelium discoideum*), and applied all the trained models to the gene annotations of each of such latter organisms. We focused the evaluation on the predicted annotations with good quality level, by setting the prediction likelihood threshold to the value $\rho = 0.8$.

Results in Table 5.6 show that models trained with the *Homo sapiens* gene GO annotations achieve much better accuracy than those obtained by using the gene GO annotations of the same organism both in training and testing (i.e. the single organism approach used in the previous section).

Table 5.6 – *Evaluation results of the cross-organisms learning method vs. the single organism one, obtained without perturbation of the training matrices and using IBk as supervised algorithm. In tests with different source and target organisms, only the source organism genes annotated to at least $M = 5$ terms of the target organism gene annotations are considered. Results show the accuracy ($Ac_{\rho=0.8}$) and number ($N_{\rho=0.8}$) of the novel gene GO annotations predicted with likelihood greater than $\rho = 0.8$, and their average level ($\overline{L}_{\rho=0.8}$) in the GO DAG (for the $\overline{L}_{\rho=0.8}$ computation, when the term of a predicted gene annotation belongs to multiple GO levels, only its lowest level was considered).*

Target	Source	$Ac_{\rho=0.8}$	$N_{\rho=0.8}$	$\overline{L}_{\rho=0.8}$	Target	Source	$Ac_{\rho=0.8}$	$N_{\rho=0.8}$	$\overline{L}_{\rho=0.8}$
Mus m.	*Homo s.*	**0.573**	6048	2.82	*Bos t.*	*Homo s.*	**0.538**	234	3.71
	Mus m.	0.407	205	2.93		*Mus m.*	0.137	388	3.55
	Bos t.	0.312	2184	2.73		*Bos t.*	0.250	4	3.00
	Gallus g.	0.221	2576	2.78		*Gallus g.*	0.314	35	3.45
	Dict. d.	0.305	5285	2.81		*Dict. d.*	0.273	220	3.27
Gallus g.	*Homo s.*	**0.390**	136	3.64	*Dict. d.*	*Homo s.*	**0.675**	765	3.99
	Mus. m.	0.100	281	3.68		*Mus m.*	0.217	327	3.96
	Bos t.	0.281	32	3.26		*Bos t.*	0.209	187	3.10
	Gallus g.	-	0	-		*Gallus g.*	0.284	81	3.96
	Dict. d.	0.205	156	2.97		*Dict. d.*	0.297	37	4.18

The accuracy ($Ac_{\rho=0.8}$) of the annotations predicted with likelihood greater than the threshold $\rho = 0.8$ shows an improvement almost always greater than 100% (except in the case of *Mus musculus*, where the improvement is just above 40%). Moreover, it is notable that this accuracy improvement is due to the use of the most studied organism to train the models. Note also that, with no effect on the predicted annotation accuracy, the number ($N_{\rho=0.8}$) of predicted annotations with likelihood greater than $\rho = 0.8$ is mainly related to the size of the annotation matrices of both the source and target organisms. For instance, with the biggest annotation matrices of both the source and target organisms (*Homo sapiens* and *Mus musculus*, respectively) we obtained the highest amount of annotations predicted with likelihood greater than $\rho = 0.8$; conversely, with the smallest annotation matrix of both source and target organism (*Gallus gallus*), there was no annotation predicted with likelihood greater than $\rho = 0.8$.

The average depth (i.e. GO level) of the gene GO annotations predicted with likelihood greater than $\rho = 0.8$ ($\overline{L}_{\rho=0.8}$) varied around levels 3 and 4 of the GO DAG; this shows that, on average, the predicted gene annotations

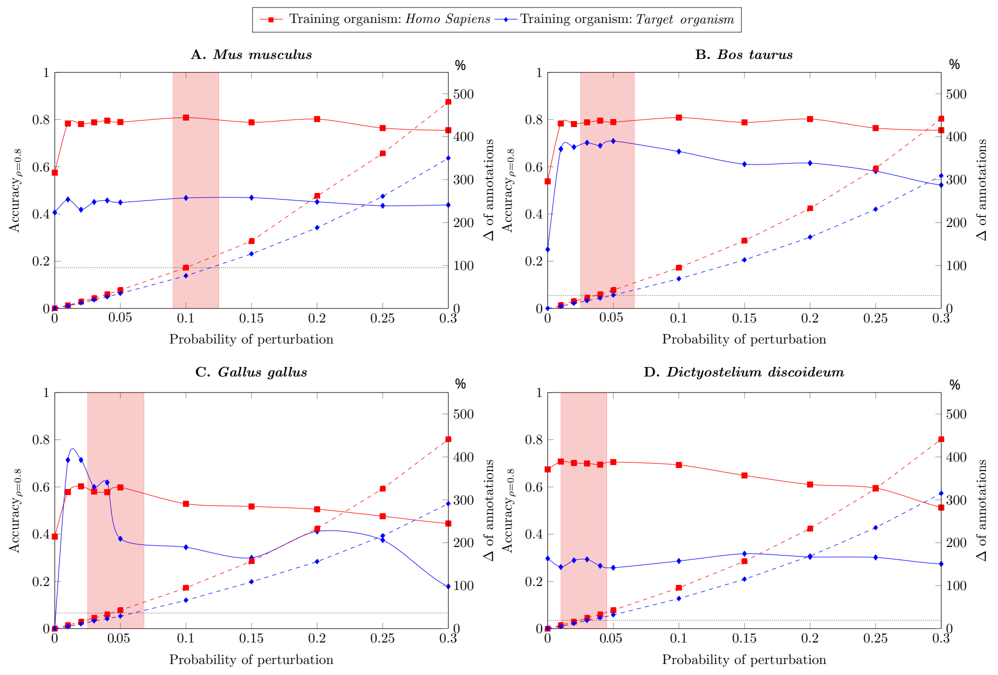


Figure 5.7 – *Evaluation results of the gene GO annotations predicted by varying the probability of the random perturbation of the training annotation matrix.*

were not very specific, but neither too generic (i.e. they were generally valuable). The not high $\overline{L}_{\rho=0.8}$ values obtained are mainly due to the fact that the annotations more reliably predicted for a gene (as those with likelihood greater than $\rho = 0.8$) are those to the GO terms that are children of terms of known GO annotations of the gene, whose far majority is at low GO levels (note that we consider also the implicit ontological annotations, not only the most specific ones).

Original and Perturbed Source Organism Annotation Matrices

Results in Table 5.6 show that the cross-organism approach using a prediction model trained on the most largely studied organism (i.e. the one with the highest number of known annotations) generally provides better predictions; thus, we further investigated the matter, testing whether the use of a random perturbation of the training annotation matrix improves the classification accuracy of the models. Figure 5.7 shows the accuracy of the predicted annotations by varying the perturbation probability of the training annotation matrix. Each figure graph regards a target organism and shows, for the training with the known gene GO annotations of the *Homo sapiens* (red, squares) or of the target organism (blue, diamonds), i) the accuracy $Ac_{\rho=0.8}$ of the novel gene GO annotations predicted with likelihood greater than $\rho = 0.8$ (solid lines), ii) the difference in percentage (Δ) of gene GO annotations in the two training matrices (dashed lines), and iii) the percentage of novel annotations (Δ) in the updated matrix of the target organism with respect to the outdated one (black, dotted line). On these graphs it can be easily observed that, also perturbing the training matrices, better results are almost always achieved with the cross-organism approach using models trained with *Homo sapiens* annotations. Moreover, the highlighted vertical bars in Figure 5.7 show the perturbation ranges where the Δ of annotations between the two training and the two target organism annotation matrices are comparable; one can notice that the tests with these perturbation ranges almost always achieve best accuracy results, or are very close to them. This confirms that greater accuracy of the model is achieved with a richer representation of the validation problem in the training set; in our scenario such richer representation can be measured by the similarity between the Δ of annotations in the training and validation matrices.

Two Original Matrices of a Well Annotated Source Organism

The above test results show a great improvement of the predicted annotation accuracies when prediction models were trained with *Homo sapiens* gene annotations. Considering this enhancement, we could conceive a tool based on *Homo sapiens* gene annotations and applicable to discover novel annotations for the genes of target organisms of interest. Furthermore, having available two original versions of *Homo sapiens* gene annotations at distinct times, it would be possible using them as source organism annotation matrices to train the prediction model without the need of the perturbation.

To evaluate the efficacy of using two original annotation matrices, at a distance of time, of the *Homo sapiens* as source organism, we performed several tests and compared their results with those obtained with the perturbation approach. In Figure 5.8 we show the trend of the predicted an-

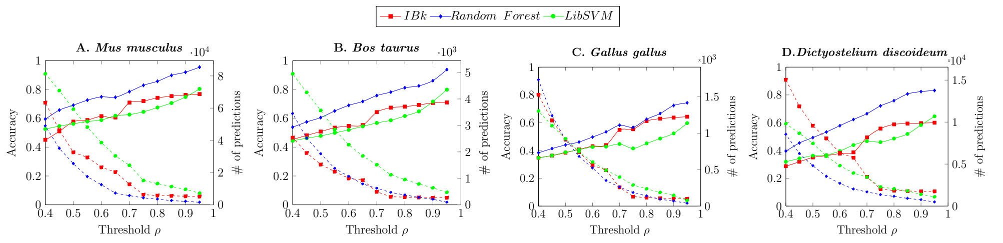


Figure 5.8 – *Evaluation results obtained by varying the threshold ρ on the likelihood of the gene GO annotations predicted using three different supervised algorithms (the IBk nearest neighbors with $K = 3$, the Random Forest decision tree and the LibSVM Support Vector Machine with radial basis kernel).*

notation accuracy Ac_ρ, while varying the threshold ρ on the likelihood of the annotations predicted using a model trained with two original matrices of *Homo sapiens* gene GO annotations, as of July 2009 and March 2013, respectively, with a percentage difference in number of annotations of $\Delta = 84.15\%$. For each target organism, we report a graph with the accuracies (solid lines, left vertical axis) and the related number of predicted annotations (dotted lines, right vertical axis) in relation to the minimum likelihood of the predicted gene GO annotations (horizontal axis). Figure 5.8 shows that the greater is the prediction likelihood threshold, the greater is the accuracy of the prediction model and the lower is the number of predicted annotations; thus, a biologist interested in very reliable, although few, novel annotations can raise the threshold as desired to obtain less, but comprehensively more reliable, predicted annotations. On the other hand, comparing results in Figure 5.8 with those in Figure 5.7, it appears that the perturbation approach provides generally better, or in few cases equivalent, results than using different original annotation versions (which are rarely available). This behavior shows that the perturbation process allows the correct representation of the discovery of new annotations; besides giving the possibility to vary the difference of annotations (Δ) between the training matrices, this allows the tuning of the models to better represent the problem.

We also evaluated the influence, on the predicted annotations, of the value of the parameter M, i.e. the minimum amount of overlapping between the annotation terms of the source and the target organisms. In

Table 5.7 we show the results obtained by varying this minimum overlapping, i.e. by considering for the prediction model training only the source organism genes that are annotated to at least M of the GO terms in the target organism gene annotations. Results show that generally, independently on the evolutionary distance of the target organism from the *Homo sapiens* and without influence on the average level in the GO DAG of the terms in the novel gene GO annotations predicted, the higher is the minimum overlapping, the lower is the prediction effectiveness. This brings to conjecture that even genes annotated to few terms in common with the gene annotations of the target organism are useful to better train the prediction model; as mentioned above, an higher number of training genes, terms and annotations (i.e. a wider and more dense source organism annotation matrix) leads to create a better prediction model. In the considered cases (i.e. *Homo sapiens* as source organism and the four considered model organisms as target organisms), on average the best accuracy results are obtained by training the prediction model with all the source organism genes that are annotated to at least $M = 5$ GO terms in common with the target organism gene annotations.

Multiple Supervised Algorithm Comparison

Our proposed cross-organism learning approach is totally independent from the specific supervised learning algorithm used to build the prediction model. To test which type of supervised algorithm can provide better accuracy results, we evaluated three different well-known supervised algorithms, namely *Support Vector Machines* (*LibSVM*, with radial basis kernel), *nearest neighbors* (*IBk*, with $K = 3$) and *decision trees* (*Random Forest*), using the implementations and default parameter values provided by Weka in its 3.7.9 version. In Figure 5.8, we report the obtained results. It can be noted that the Random Forest algorithm always provides better accuracy; however, the time needed to build the classification models using Random Forest is extremely higher than that required by a nearest neighbor classifier like IBk. On the other hand, the LibSVM algorithm often gives a higher number of predicted annotations, but less reliable (i.e. with lower accuracy).

Table 5.7 – *Evaluation results of the predicted gene GO annotations obtained by varying the minimum number M of GO terms in common between source and target organism gene annotations.*

Organism	Measure	$M = 50$	$M = 25$	$M = 10$	$M = 5$	$M = 1$
Mus m.	$Ac_{\rho=0.8}$	0.610	0.672	0.735	0.742	**0.743**
	$N_{\rho=0.8}$	8040	5398	5911	5799	5818
	$\overline{L}_{\rho=0.8}$	3.22	3.20	3.12	3.12	3.11
Bos t.	$Ac_{\rho=0.8}$	0.556	0.683	**0.702**	0.682	0.686
	$N_{\rho=0.8}$	419	338	265	292	280
	$\overline{L}_{\rho=0.8}$	3.72	3.87	3.99	3.94	3.95
Gallus g.	$Ac_{\rho=0.8}$	0.342	0.438	0.569	**0 .612**	0.573
	$N_{\rho=0.8}$	222	153	123	113	110
	$\overline{L}_{\rho=0.8}$	3.74	3.75	3.60	3.68	3.95
Dict. d.	$Ac_{\rho=0.8}$	0.426	0.570	**0.593**	0.589	0.589
	$N_{\rho=0.8}$	3230	2266	1891	1854	1856
	$\overline{L}_{\rho=0.8}$	3.57	3.62	3.67	3.68	3.69
Average	$Ac_{\rho=0.8}$	0.484	0.591	0.650	**0.656**	0.648
	$N_{\rho=0.8}$	2977.7	2038.7	2047.5	2014.5	2016.0
	$\overline{L}_{\rho=0.8}$	3.56	3.61	3.60	3.60	3.68

Predicted Gene Annotation Example

As an example, in Figure 5.9 we report the Directed Acyclic Graph of the GO Cellular Component terms of all annotations predicted and known of the *baculoviral IAP repeat containing 5* (*BIRC5*) gene of the *Bos taurus* organism. For readability reasons, we show a GO Cellular Component DAG, which generally includes less nodes than other GO sub-ontology DAGs; furthermore, we do not show all the GO terms that have no known association or low predicted association likelihood l, i.e. with $l < 0.2$, with the *Bos taurus BIRC5* gene. The predicted annotations shown were obtained by training the prediction model with two original versions of *Homo sapiens* gene GO annotations (as of July 2009 and March 2013, respectively), with $M = 5$ minimum common terms between the source and target organism gene annotations and using IBk as learning algorithm. In the DAG, ellipses represent GO terms of the known annotations of the *Bos taurus BIRC5* gene as of July 2009, whereas rectangles indicate GO terms of the predicted annotations with likelihood $l \geq 0.2$: as higher is the likelihood, as darker is

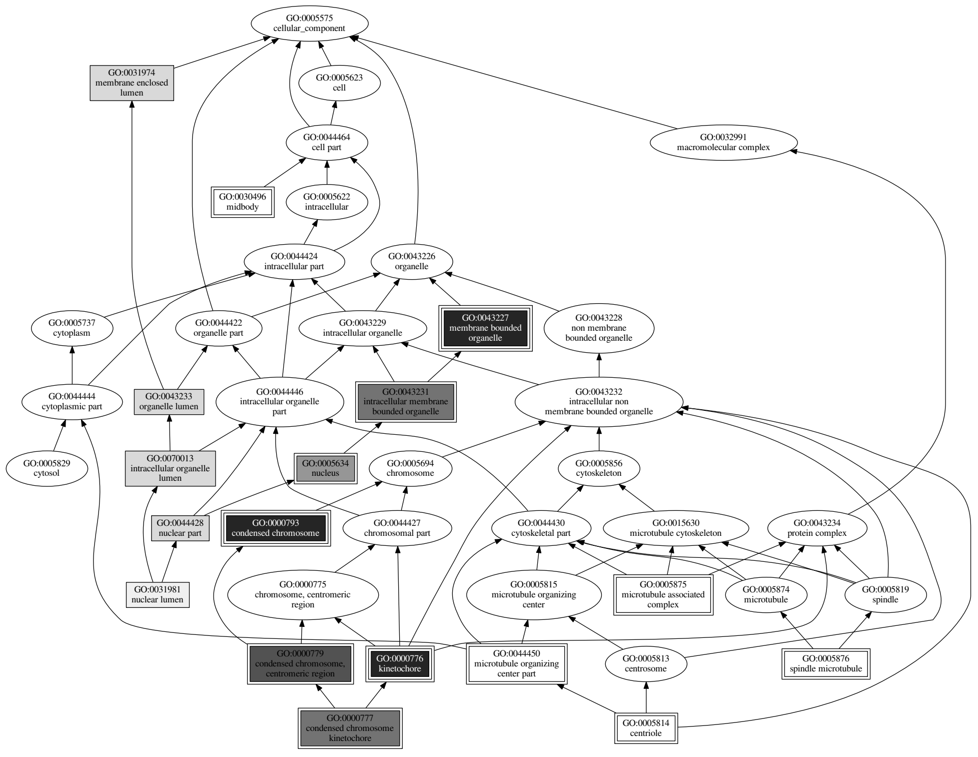

GO:0005575 cellular_component
GO:0031974 membrane enclosed lumen
GO:0005623 cell
GO:0044464 cell part
GO:0032991 macromolecular complex
GO:0030496 midbody
GO:0005622 intracellular
GO:0044424 intracellular part
GO:0043226 organelle
GO:0005737 cytoplasm
GO:0044422 organelle part
GO:0043229 intracellular organelle
GO:0043227 membrane bounded organelle
GO:0043228 non membrane bounded organelle
GO:0044444 cytoplasmic part
GO:0043233 organelle lumen
GO:0044446 intracellular organelle part
GO:0043231 intracellular membrane bounded organelle
GO:0043232 intracellular non membrane bounded organelle
GO:0005829 cytosol
GO:0070013 intracellular organelle lumen
GO:0005634 nucleus
GO:0005694 chromosome
GO:0005856 cytoskeleton
GO:0044428 nuclear part
GO:0000793 condensed chromosome
GO:0044427 chromosomal part
GO:0044430 cytoskeletal part
GO:0015630 microtubule cytoskeleton
GO:0043234 protein complex
GO:0031981 nuclear lumen
GO:0000775 chromosome, centromeric region
GO:0005815 microtubule organizing center
GO:0005875 microtubule associated complex
GO:0005874 microtubule
GO:0005819 spindle
GO:0000779 condensed chromosome, centromeric region
GO:0000776 kinetochore
GO:0044450 microtubule organizing center part
GO:0005813 centrosome
GO:0005876 spindle microtubule
GO:0000777 condensed chromosome kinetochore
GO:0005814 centriole

Figure 5.9 – *Example of Directed Acyclic Graph for the Gene Ontology Cellular Component terms involved in the annotations known and predicted for the baculoviral IAP repeat containing 5 (BIRC5) gene of the Bos taurus organism.*

the rectangle background. GO terms of predicted annotations confirmed in the validation dataset, which includes the GO annotations of the *Bos taurus BIRC5* gene as of March 2013, are in double perimeter rectangles, while in triple perimeter rectangles are the GO terms of the annotations predicted with likelihood greater than $\rho = 0.8$ (all of them are confirmed in the validation dataset). The octagons represent the few remaining GO terms that were associated with the *BIRC5* gene in the updated *Bos taurus* annotations available (as of March 2013), but that were not predicted.

The DAG in Figure 5.9 suggests several, very interesting, novel GO Cellular Component annotations for the *Bos taurus BIRC5* gene, which is part of the *inhibitor of apoptosis* (*IAP*) gene family whose members encode negative regulatory proteins preventing apoptotic cell death. Some of its predicted annotations, i.e. 7 out of the 12 annotations (58.33 %) predicted with likelihood l greater than $\rho = 0.8$ (involving the double and triple perimeter darker rectangles in the DAG), including the annotation to the *nucleus* (*GO:0005634*) term, were found confirmed in the 45 month later updated version of the *Bos taurus* gene annotations; others, such as the annotations to the *membrane-enclosed lumen* (*GO:0031974*) and to four of its descendant terms, are known for the human ortholog gene and could be confirmed for this *Bos taurus* gene in the future.

In the DAG, it can be also seen what would be the effect of lowering the threshold on the predicted annotation likelihood: GO terms within single and double perimeter rectangles might become included in the predicted annotations of the *Bos taurus BIRC5* gene, in accordance with their likelihood; a lower threshold could add many predicted annotations, but both confirmed (double perimeter rectangles) and not confirmed (single perimeter rectangles) ones, although the latter ones could be confirmed in a subsequent updated version of the known GO annotations of the gene. The high likelihood of the annotations confirmed, among those that the method predict with any likelihood, shows the high potential of our method in correctly predicting new annotations for a less studied organism by transferring them from a more studied one used to train the prediction model. The five GO annotations predicted for the *Bos taurus BIRC5* gene that have not (yet) been confirmed (single perimeter rectangles in Figure 5.9) regard GO terms all hierarchically related and are induced by the annotation to the most specific of such terms (*nuclear lumen - GO:0031981*). Note that the same *BIRC5* gene in human (i.e. the human ortholog) is directly annotated to a child of such most specific term (i.e. to *nuclear chromosome - GO:0000228*) with reliable Inferred from Direct Assay (IDA) evidence (Vong et al., 2005); thus, the *BIRC5* human ortholog gene is also indirectly annotated to all the GO terms involved in the annotations that our method predicts for the *Bos taurus BIRC5* gene which are not yet confirmed in the available *Bos taurus* gene annotations.

5.4.4 Discussions

The rationale of the cross-organism approach is that the richer and denser is the training annotation set, the greater is the accuracy of the annotation prediction model. Through the several performed evaluations, we proved that the richer knowledge of an organism can be used to build a better prediction model for other less-known organisms, which provides more accurate results than an equivalent model built using the lesser information available for the organism to be predicted. This approach is feasible because of our proposed data representation: the classification models are guided by the co-occurrences of annotation terms in the considered gene annotation profiles, indiscriminately from the genes, and thus from the organisms, in which they occur.

The proposed cross-organism approach allows training prediction models using any term involved in the annotation of the source organism; this implies the possibility of discovering new annotations for the target organism to terms representing biological concepts that have newer been associated with the target organism. Although such predicted annotations could be very valuable, guiding biologists to discover novel features completely unknown for the target organism, they could include annotations that are not meaningful for the target organism; furthermore, such predicted annotations could not be automatically validated using updated known annotations become available for the target organism. Thus, for the methodological purpose of this work, in the performed test we predicted new gene annotations only to those terms already included in the gene annotations of the target organism.

We tested our approach with both the same and different organisms as source and target organism, using gene annotation datasets of different sizes and densities and regarding organisms at different evolutionary distances. In all the conducted tests, the prediction models trained on a different more studied organism always achieved best accuracy of the predicted annotations, even when the source and target organisms were distantly related evolutionarily. The accuracy obtained using the most studied organism, the *Homo sapiens*, as source organism almost always overcomes by more than 100 % the accuracy obtained using the same organism as source and target organism. Moreover, the performed tests show that our introduced perturbation method permits to further improve prediction results, particularly when the difference in number of annotations between the two training annotation matrices is comparable to the one between the test and validation annotation versions. Validation results also confirmed that even without perturbation, i.e. avoiding the tuning of the perturbation parameter, our cross-organism approach ensures good annotation predictions.

Our approach provides ranked lists of predicted annotations that describe novel gene functionalities and have an associated likelihood value. They are very valuable, for instance, to complement available annotations for better coverage of the several gene and protein functionalities, or to quicken the annotation curation process by focusing it on the prioritized novel annotations predicted. Although our approach cannot be applied to discover new features of completely unannotated genes, the evaluation results show that it can reliably predict novel annotations also of genes which are only very limitedly annotated to the terms included in the gene annotations of the well annotated source organism used.

Finally, we point out that, despite the here presented results refer to Gene Ontology annotations of genes, the proposed approach can be equally applied also to protein annotations and it is not bound to Gene Ontology annotations, but it can be applied to any type of controlled annotations, from an ontology or even a flat terminology.

5.5 GO Term Annotation of Biomedical Literature

Nowadays, GO is the de facto standard for functional annotation of genes (Lewis, 2005; Rubin et al., 2008). The two main efforts of the GO project involve: i) the development and maintenance of a controlled vocabulary (ontologies) of functional attributes; ii) the annotation of genes in terms of the their associated attributes.

The majority of GO annotations are assigned by using computational methods (du Plessis et al., 2011; Radivojac et al., 2013; Barutcuoglu et al., 2006; Tao et al., 2007), although electronically inferred annotations are usually considered as inaccurate and unreliable (Lomax, 2005; Škunca et al., 2012). At the state of the art, GO annotations derived from manual curation of scientific literature can be still regarded as the *gold-standard* in terms of quality and specificity. However, the manual annotation step is extremely time-consuming, and thus it is one of the major bottlenecks in GO curation. The annotation task has become an even harder challenge in the post-genomic era, which has been characterized by an unprecedented growth in the production of biomedical literature. As an example, The Arabidopsis Information Resource's curation team (TAIR) reports that, in the recent years, it has been able to curate only a relatively small fraction ($\sim$ 30%) of the newly published literature on Arabidopsis Thaliana (Li et al., 2012a). Due to this enormous growth of biological information in form of unstructured text, there is an increasing interest in text mining tools that can aid GO curators during the labor-intensive annotation task (Hirschman et al., 2012).

The main reference for the state-of-the-art in automated GO curation is the text-mining challenge task for literature-based GO annotation at the BioCreative experiments (Blaschke et al., 2005; Mao et al., 2014). The main effort of BioCreative experiments is to provide as much as possible realistic biological scenarios for performance assessment of automated annotation tools. The two GO annotation-specific subtasks at the most recent BioCreative IV (Mao et al., 2014) were aimed at assessing automated GO recognition, given as input full-text articles with relevant gene information: i) Task A. Retrieve GO evidence text for relevant genes (text retrieval task); ii) Task B. Predict GO terms for relevant genes (concept-recognition task). The performances were assessed with both gold-standard GO annotations and the help of expert GO curators. The overall conclusions of BioCreative's assessors are that, despite the improvement over the last decade, even the best performing methods are still not accurate enough to aid manual GO curation.

In this work we focus on GO annotation of biomedical literature, namely the automatic assignment of GO terms to a scientific publication. This problem is closely related to BioCreative's Task B, which further requires to identify the associations between GO annotations related to a publication and genes within its text. GO annotation of biomedical literature is itself a relevant sub-problem of the most general gene annotation task. First of all, divide-and-conquer strategies often reduce complexity for difficult problems. Therefore, decoupling the GO annotation task (for publications) from the GO association task (for genes within the publication) leads to two simpler subproblems, which could be approached with ad hoc techniques. Also, the unsatisfactory results obtained so far in automated annotation could be due to a lack of easily accessible gold-standard training data, such as full-text articles and evidence sentences related to GO terms and gene names. Conversely, public web repositories contain a growing number of heterogeneous metadata and annotations, which can be automatically processed by text annotation tools. Furthermore, the literature annotation problem is of interest in itself for the development of ontology based search engines, such as GoPubMed (Doms and Schroeder, 2005), which could be used as a pre-filter by human curators.

In the Information Retrieval (IR) community, GO annotation of biomedical literature can be seen as a *hierarchical multi-label classification* problem, where GO terms represent (a large number of) categories in which biomedical publications have to be classified (Section 3.3.1).Generally, classical IR approaches are *topic-centric*, in that they rely on indexed representations of the categories. Indexing is typically performed by using a controlled vocabulary of terms contained in pre-labeled documents. The training of these methods becomes impractical at increasing number of categories and the classification accuracy drops down as the categories are poorly represented, thus state-of-the-art IR methods are best suited on hierarchies consisting of relatively few and well-represented categories. Some attempts have been made to apply supervised text classification methods in large taxonomies (Liu et al., 2005; Bennett and Nguyen, 2009; Tao et al., 2012), but the results are still quite poor in comparison to those obtained in smaller settings. To our knowledge, the only available tool that directly addresses the hierarchical multi-label classification over GO categories is GOCat (Gobeill et al., 2013). Differently from classical IR approaches, GOCat is *document-centric*, in that it uses a k-Nearest Neighbor (k-NN) strategy (Altman, 1992). While in topic-centric approaches a query document is classified by means of direct comparisons with categories, with GOCat, annotations for a query abstract are inferred from the k most similar pre-annotated publications in a knowledge base. The k-NN approach proved to be valuable and best performing in one of the GO annotation subtasks (Task B) at BioCreative IV (Gobeill et al., 2014).

This section introduces several new ideas for GO annotation of biomedical literature. First, we exploit a novel approach that combines the document-centric and topic-centric strategies. Second, we experiment with novel measures for assessing the similarity between documents and against category membership. Our resulting annotation tool, called GOTA[3] (**GO t**erm **a**nnotation) and presented in (Lena et al., 2015), makes use of publication title, abstract, references and year of publication (all readily available in public repositories, such as PubMed), although the approach itself is easily scalable to incorporate more (or less) information. We test the classification capabilities on a quite large set of scientific publications (15.000 documents)

[3]`http://gota.apice.unibo.it`

and with respect to different evaluation metrics. By comparison with GO-Cat, our approach shows better performance over all considered metrics. As a general consideration, the experimental results are encouraging and, in some aspects, surprising. In summary, on the average almost half of the gold standard annotations can be recovered for a query publication. This is a quite satisfactory result in comparison to other general-purpose hierarchical multi-label classification tools. More specifically, the annotations recovered for a publication, although not precise, are usually semantically close to the gold-standard annotations. Furthermore, we found that the classification capabilities improve over specie-specific knowledge bases. This suggests that the GO curation task could benefit from developing species-centered annotation tools. To our opinion, the most interesting findings are related to the smallest source of information that can aid classification. It is not unexpected that the biological content of a paper is summarized in few short sentences within the text, while the rest is background noise. In our experiments, given the information available for classification, it comes out that the strongest signal comes from the publication title.

5.5.1 Benchmark Set

The Gene Ontology vocabulary was retrieved from Gene Ontology Consortium web resource[4]. At the time of the retrieval, the GO vocabulary consisted of 39,399 distinct terms partitioned into three main categories, structured as directed acyclic graphs (DAG) with a unique root: 26,099 terms of type Biological Process (BP), 9,753 of type Molecular Function (MF) and 3,547 of type Cellular Component (CC). The literature-based GO annotations were retrieved from the UniProt-GOA web resource[5]. We downloaded all the single Gene Association files, corresponding to set of proteins in different species/knowledge bases (30 overlapping sets). The single files were parsed in order to extract only the associations between pairs of GO terms and PubMed identifiers (PMIDs), discarding the gene product references. We ended up with a benchmark set of 328,357 pairs, consisting of 115,402 distinct PMIDs annotated with 22,839 distinct GO terms. The title, abstract and reference information related to the PMIDs in the benchmark set were downloaded in XML format from PubMed[6].

[4]`http://geneontology.org/ontology/go-basic.obo`
[5]`http://www.ebi.ac.uk/GOA/downloads`
[6]`http://www.ncbi.nlm.nih.gov/pubmed`

For 1256 out of 115,402 documents (1%) the information downloaded from PubMed consists of the publication title only. For a total of 45,341 documents (39%) also the cited references are available. Out of 22,839 GO identifiers in benchmark set, 14,889 are of type BP, 5,951 of type MF and 1,999 of type CC. The number of GO annotations for each PMID ranges from 1 to 1,309 terms, with an average of 2.8. The distribution of annotations in the benchmark set is not uniform. In particular, except for a small subset, the number of annotations per PMID is quite small: 39% of PMIDs have a single GO annotation and 99% at most 10 distinct annotations.

Our method requires a preprocessing of the underlying Knowledge Base (KB) and a parameter tuning phase (see Text preprocessing Section and Tuning of parameters Section below). Both these phases are extremely time-consuming, which prevents the possibility of performing multiple rounds of cross validation in reasonable time. For this reason, for performance evaluation, we randomly selected a large sample of $15,000$ publications from the benchmark set derived from UniProt-GOA. Publications in this set are used for testing the classification performances, while the remaining $\sim 100,000$ publications constitute the underlying KB of our method. Almost 6% of the terms represented in the test set have no associated publication in the KB. Among the missing terms, there is a higher number of BP terms (67%) in comparison to MF (25%) and CC (8%) terms.

5.5.2 Methods

We use the combination of a publication-centric and term-centric approach to capture the relation between a scientific publication and a GO term:

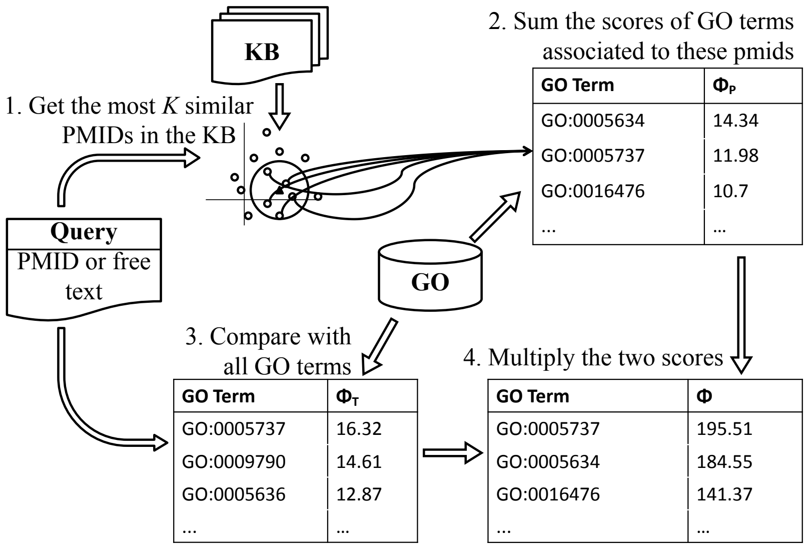
1. *Publication-centric*: the relation between a GO term t and a query q is inferred from the query's similarity with annotated publications in the underlying KB.

2. *Term-centric*: the relation between the topics covered by a query q and a GO term t is determined by direct comparison.

The likelihoods obtained from these two approaches are simply combined into a single relatedness score:

$$\Phi(q,t) = \Phi_P(q,t) \cdot \Phi_T(q,t) \tag{5.6}$$

where Φ_P and Φ_T are the *publication-centric* and the *term-centric* similarity scores, respectively. The meaning of Eq. 5.6 is quite simple (see Figure 5.10). The Φ_P function makes use of a k-NN strategy to select a ranked list of GO terms (only terms associated to some publication in the KB can be selected). The Φ_T function provides a re-weighting of the selected terms.


KB
1. Get the most K similar PMIDs in the KB
2. Sum the scores of GO terms associated to these pmids
Query
PMID or free text
GO
3. Compare with all GO terms
4. Multiply the two scores

GO Term	Φ_P
GO:0005634	14.34
GO:0005737	11.98
GO:0016476	10.7
...	...

GO Term	Φ_T
GO:0005737	16.32
GO:0009790	14.61
GO:0005636	12.87
...	...

GO Term	Φ
GO:0005737	195.51
GO:0005634	184.55
GO:0016476	141.37
...	...

Figure 5.10 – GOTA *workflow: Graphical representation of Eq. 5.6.*

The details of the preprocessing and tuning phase, as well as the exact definition of the two similarity functions in Eq. 5.6 are given in the rest of this Section.

Text preprocessing.

We perform typical preprocessing operations to transform each publication and GO term into a structured representation more manageable for querying. We make use of the quite common *Bag of Words* (BoW) representation of documents. The elements of the vector are weighted in order to balance their relative importance. Other than BoW representation, we experimented with different features that can be automatically extracted from PubMed and GO knowledge bases. In detail, a publication p is indexed by the following information:

- $\mathbf{w}_p$ (*text*): BoW built from the abstract and title of the publication p. More generally, this feature represents the BoW built from unstructured text associated to the publication.

- $\mathbf{t}_p$ (*title*): BoW built from the title of the publication p.

- $\mathbf{r}_p$ (*references*): weighted vector of references (PMIDs). As for the BoW, each entry of the vector indicates the presence (or absence) of a reference within the publication. The references are weighted according to their importance.

- $\mathbf{y}_p$ (*year*): year of publication of p.

All the features above can be easily extracted for documents indexed into PubMed. Some of these features could not be available for some publication (references, for example), particularly for query publications. We assume that at least the $\mathbf{w}_p$ feature vector is always non-null (i.e. contains entries greater than zero). A GO term t is indexed with the following information (with abuse of notation, we use the same symbol when the term-specific and publication-specific features are related):

- $\mathbf{w}_t$ (*text*): BoW built from unstructured text associated to the term t. In this case, the text associated to t includes the term name, synonyms and description (all available in the GO knowledge base), as well as all the titles and abstracts of those publications in the KB associated to term t (these informations could be absent).

- $\mathbf{t}_t$ (*name*): BoW built from the name and synonyms of the term t.

- $\mathbf{y}_t$ (*year*): average year of the publications annotated with term t.

The $\mathbf{w}_t$ and $\mathbf{t}_t$ BoWs are always non empty, while the $\mathbf{y}_t$ feature can be unspecified (if there is no publication in the KB annotated with t).

We use the following procedure to create a pool of words (BoW) appearing in titles and abstracts of publications in the KB, as well as title, synonyms and descriptions of GO terms. Single words occurring in the text are extracted, discarding punctuation. A stopword list is used to filter-out all the words that are not enough informative about the text (such as, *actually, after,* etc). The standard Porter stemming algorithm (Porter, 1980) is used to group words with common stems. As the method works accurately even without filtering or selecting features, which generally are costly steps that require further complex tuning of parameters, we do not perform such task in this paper. However, we use the common *tf-idf* scheme (previously described in Section 3.2.4), to assign a relevance weight to words related to each feature.

To automatically extract weighted vectors of references $\mathbf{r}_p$, we use the same criteria of the *idf* measure. Publications indexed in PubMed are identified with a unique PMID and cited references are represented as a list of PMIDs. Thus, the *idf* measure for references can be easily calculated by:

$$idf_{\text{ref}}(\text{PMID}) = \log \frac{\text{total number of publications with bibilography}}{\text{number of publications citing PMID}}$$

With the BoW representation, the similarity between two feature vectors x, y can be measured by means of the commonly used *cosine similarity* (Eq. 3.1).

Publication-centric score.

The first similarity score between a publication query and a GO term is recovered by comparison with annotated literature, namely the publications in the underlying KB. This is basically a k-NN approach. First, the publication query is compared with all the publications in the KB, in order to detect those that appear to be most closely related to the query (Step 1 in Figure 5.10). Second, a ranked list of GO annotations for the query is inferred from the recovered publications (Step 2 in Figure 5.10).

The comparison score between a query publication q and a target publication p in the KB is given by the product of four different similarity measures:

$$\phi_P(q,p) = \Pi_{i=1}^{4}(1 + f_i(q,p))^{m_i} \tag{5.7}$$

where $0 \leq f_i(q,p) \leq 1$ gives a similarity score associated to a single publication-related feature and the power m_i represents its relative weight. The four similarity functions $f_i(q,p)$ are defined by:

1. (*Text similarity*) Cosine similarity between BoW of the query and target publications:

$$f_1(q,p) = \cos(\mathbf{w}_q, \mathbf{w}_p)$$

2. (*Title similarity*) Cosine similarity between the title-related BoW of the query and target publications:
$$f_2(q, p) = \cos(\mathbf{t}_q, \mathbf{t}_p)$$

3. (*References similarity*) Cosine similarity between the weighted vector of references of the query and target publications:
$$f_3(q, p) = \cos(\mathbf{r}_q, \mathbf{r}_p)$$

4. (*Year similarity*) Distance between the publication year of the query and target. We decided to normalize this value with a maximum distance of 50 years:
$$f_4(q, p) = \begin{cases} 0 & \text{if } |\mathbf{y}_q - \mathbf{y}_p| > 50 \\ (50 - |\mathbf{y}_q - \mathbf{y}_p|)/50 & \text{otherwise} \end{cases}$$

We remark that the four features described above can be used for the classification only if the query publication is indexed into PubMed. In all the other cases, i.e. when the query consists of unstructured text only, the only available feature is $f_1(q, p)$. Furthermore, also for publications indexed in PubMed some information can be missing (abstract and/or bibliography, for example). However, due to the definition of Eq. 5.7, the missing information does not affect the calculation of the comparison score. Symmetrically, the comparison score can be easily extended to incorporate more features.

The measure $\phi_P(q, p)$, computed by Eq. 5.7, is used to rank the publications in the KB according to their similarity with the query. The gold-standard GO annotations of the top-K ranked publications are then transferred to the query, by using $\phi_P(q, p)$ as their relative weight. The *publication-centric similarity score* between a query q and a GO term t is then given by:

$$\Phi_P(q, t) = \sum_{\text{top-}K \text{ ranked } p \in KB} \begin{cases} \phi_P(q, p) & \text{if } p \text{ has annotation } t \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

Term-centric score.

The second similarity score between a publication query and a GO term is based on the direct comparison between a query q and a term t (Step 3 in Figure 5.10). Also in this case, the score is given by the the product of different similarity measures:

$$\Phi_T(q,t) = \Pi_{i=1}^{5}(1 + g_i(q,t))^{n_i} \tag{5.9}$$

where $0 \leq g_i(q,t) \leq 1$ is the score associated to a single GO term-related feature and the power n_i represents its relative weight. The five similarity functions $g_i(q,t)$ are defined by:

1. (*Text similarity*) Cosine similarity between the BoW of the query and term:
$$g_1(q,t) = \cos(\mathbf{w}_q, \mathbf{w}_t)$$
2. (*Title similarity*) Cosine similarity between the title-related BoW of the query and term:
$$g_2(q,t) = \cos(\mathbf{t}_q, \mathbf{t}_t)$$
3. (*Name frequency*) Percentage of the words in the title-related BoW of the term that are contained in the BoW of the query:
$$g_3(q,t) = \frac{|\mathbf{t}_t \cap \mathbf{w}_q|}{|\mathbf{t}_t|}$$
4. (*Name occurrence*) Normalized number of occurrences of the name or synonym of the term in the query. In this case, we seek the full name of the term (and its synonyms) in the entire query text. The number of occurrences c is normalized to 4.
$$g_4(q,t) = \begin{cases} 1 & \text{if } c > 4 \\ c/4 & \text{otherwise} \end{cases}$$
5. (*Year similarity*) Distance between the publication year of the query and term:
$$g_5(q,t) = \begin{cases} 0 & \text{if } |\mathbf{y}_q - \mathbf{y}_t| > 50 \\ (50 - |\mathbf{y}_q - \mathbf{y}_t|)/50 & \text{otherwise} \end{cases}$$

Tuning of parameters

The computation of Eq. 5.6 requires a preprocessing of the underlying KB, as well as the tuning of a few parameters. The parameter tuning has been performed over the 100,402 publications in the KB, which has been randomly extracted from the 115,402 publications of the benchmark text set. Moreover we randomly split the KB into two text sets: a training set of 90,402 publications and a validation set of 10,000 publications. Generally the tuning of parameters requires performing a cross-validation strategy, however, as previously explained, we do not use such a strategy because of (i) the computational time required for preprocessing a so large data set and because (ii) the amount of data, randomly selected for the validation set, is representative of the entire benchmark text set. In fact, according to the sampling theory (Wonnacott and Wonnacott, 1972), in the worst case the representativeness (or sampling) error of this validation set is less than 0.98%, with a confidence level of 95%. We also remark that the effectiveness of the method does not strongly depend from the parameter tuning. In fact, without it the drop in classification accuracy over the validation set is less than 2% (data not shown).

The parameters involved in the tuning process belong to the two functions that calculate the two similarity scores. In particular the computation of the similarity score $\Phi_P(q,t)$ in Eq. 5.8 requires the tuning of the power parameters m_i in Eq. 5.7 and the K parameter in Eq. 5.8. We tested all the combinations of the m_i ranging from 0 to 4 each. Note that a 0-value leads to not considering the related feature. The selected power parameters are: $m_1 = 4, m_3 = 3, m_2 = m_4 = 1$. While the power parameters seem to give more importance to some similarity scores, such as $f_1(q,p)$, they actually play a balancing role, since the actual values of the four similarity scores follow different distributions. We also remark that, when the input query consists uniquely of unstructured text, the classification relies only on the $f_1(q,p)$ similarity score. In such situations, the power parameter m_1 has only a limited effect on the scoring. We also tested the method by varying the number (K) of most similar publications considered in the Eq. 5.8. We experimented that varying K from 50 to 1000, we obtained the best results with K ranging from 50 to 300, with a maximum peak at $K = 150$.

As for Eq. 5.8, we tuned the power parameters n_i for the Eq. 5.9 testing all combinations with each value ranging from 0 to 4. The selected parameters are: $n_1 = 4, n_3 = 2$, $n_2 = n_4 = n_5 = 1$. When the input query consists just of unstructured text, the only power parameters used are n_1, n_3 and n_4.

We further considered the problem of assigning a confidence threshold (low, medium, high) to the predicted annotations. The aim of such filter is to provide to the user a confidence level, mainly for those queries that have a very low biological content. The confidence score is not considered for the experimental testing in the paper, but it is made available on the online Web application.

Evaluation metrics

In a real world scenario, a text mining tool for GO classification should assist a database curator for manual extraction of GO annotations from scientific literature. From this point of view, a classification tool can be useful whether it can provide a small set of annotations that accurately cover the biological content of a scientific publication. Thus, for performance assessment, we particularly focus on the evaluation of the top-10 ranked terms only (recall that 99% of the publications in our benchmark set have at most 10 distinct GO annotations).

Furthermore, in order to make the results more accessible to a heterogeneous community, we adopt different metrics from different domains. In particular, we use specific evaluation metrics from the IR domain, as established at TREC experiments (Voorhees, 2002), and metrics exploited for performance assessment in biologically-related domains, such as BioCreative (Mao et al., 2014) and CAFA (Radivojac et al., 2013) experiments. The topic covered at CAFA is the evaluation of protein function prediction methods. As in our setting, also at CAFA the performances are assessed over the GO hierarchy. We further introduce a third set of metrics, based on recent approaches for measuring semantic similarity between set of terms/concepts (Schlicker et al., 2006).

In the rest of this section, we adopt the following notation. For a given query, we denote with T and P its related set of gold-standard (from GOA) and predicted (from the classifier) GO annotations, respectively. We use lowercase letters, such as t and p, to denote single terms within T and P, respectively. We can assume that predicted terms in P are ranked according to the classifier score. We denote with P_k the top-k ranked terms in P. With abuse of notation, we use P_ρ to denote also the subset of predicted terms in P with a score greater than or equal to score threshold ρ. A detailed description of the metrics follows.

TREC metrics.

The *reciprocal rank* metric evaluates the precision at first correctly predicted term by computing the reciprocal of its rank:

$$RR_k(T,P) = \frac{1}{\min\{i \mid T \cap P_i \neq \emptyset, 1 \leq i \leq k\}}. \tag{5.10}$$

The MRR_k is computed by averaging the RR_k over the entire set of evaluated queries. The *recall at rank k* metric evaluates the average fraction of relevant GO terms in the top-k predictions returned by the classifier:

$$R_k(T,P) = \frac{|T \cap P_k|}{|T|}. \tag{5.11}$$

The RR_k and R_k measures are strictly greater than zero only if $T \cap P_k \neq \emptyset$. For performance evaluation, here we consider only the mean reciprocal rank and recall at the top-10 predictions, MRR_{10} and R_{10}, respectively.

CAFA/BioCreative metrics.

These metrics, introduced to reflect the hierarchical nature of GO, are based on a variant of the standard precision/recall measures, called *hierarchical precision/recall* (Verspoor et al., 2006). The hierarchical measures are some of the metrics adopted at the BioCreative IV experiments. The hierarchical-precision at rank k is defined by

$$hP_k(T,P) = \frac{|\mathcal{A}(T) \cap \mathcal{A}(P_k)|}{|\mathcal{A}(P_k)|}, \tag{5.12}$$

and the hierarchical-recall at rank k by

$$hR_k(T,P) = \frac{|\mathcal{A}(T) \cap \mathcal{A}(P_k)|}{|\mathcal{A}(T)|}, \tag{5.13}$$

where $\mathcal{A}(X)$ denotes the set of all the ancestors of terms in X, recursively propagated up to the root(ρ) of the GO hierarchy. The set $\mathcal{A}(X)$ contains also X, i.e. $X \subseteq \mathcal{A}(X)$. The hierarchical-precision and hierarchical-recall can be combined into a single metric, the hierarchical F-measure (harmonic mean):

$$hF_k(T, P) = \frac{2 \cdot hP_k(T, P) \cdot hR_k(T, P)}{hP_k(T, P) + hP_k(T, P)} \tag{5.14}$$

The $hP_k(T, P)$ measure tends to assign high scores when P contains very generic terms, such as those at the top of the hierarchy. Thus, the $hP_k(T, P)$ measure is not very robust over GO, since it gives high scores even when P consists of just few non-informative terms, such as the three roots of the GO hierarchy. Symmetrically, the $hR_k(T, P)$ measure tends to assign high scores when P contains very specific terms, such as those at the bottom of the hierarchy. Since the GO hierarchy contains many leaf terms, the hR_k measure is more robust than the hP_k over GO. Furthermore, if we choose a fixed k (= 10 or 20), even if $T \subset P$, $hP_k(T, P)$ would generally provide a poor estimation of the classification capabilities, due to the highly unbalanced number of annotations per publication in our benchmark set. Conversely, the unbalanced number of annotations does not affect the hR_k metric. The hF_k metric is more robust than hP_k but it still suffers from the unbalanced amount of annotations in our dataset. For these reasons, here we consider only the hierarchical recall at rank 10, hR_{10}.

With small modifications, the hierarchical measures have been adopted also at CAFA experiments. In detail, at CAFA, the top-ranked predictions are selected with respect to a given score threshold ρ, not a rank k. The hierarchical precision and recall in Eq. 5.12 and Eq. 5.13, respectively, just need to be recoded with respect to score thresholds. Note that, in $\mathcal{A}(P_\rho)$ only the predicted terms with score greater than or equal to ρ are propagate up to the root(ρ) of the ontology. Furthermore, the hierarchical precision $hP_\rho(T, P)$ is assumed to be equal to zero if $|P_\rho| = 0$. For a given dataset D consisting of pairs of true and predicted annotations, and for a given score threshold ρ, CAFA's average hierarchical precision at ρ is defined by

$$hP_\rho(D) = \frac{1}{m_\rho(D)} \sum_{(T,P)\in D} hP_\rho(T, P),$$

where $m_\rho(D) = |\{P|(T, P) \in D, |P_\rho| > 0\}|$ is the number of non-empty predictions at threshold ρ. Asymmetrically, the average hierarchical recall at threshold ρ is averaged over all pairs in the dataset, and it is defined by

$$hR_\rho(D) = \frac{1}{|D|} \sum_{(T,P)\in D} hR_\rho(T,P).$$

These last two measure can be combined into a single metric, the F-measure (harmonic mean), at different score thresholds. The main evaluation metric at CAFA, which we also consider here, is the maximum value of the harmonic mean over all thresholds:

$$hF_{max}(D) = \max_\rho \left\{ \frac{2 \times hP_\rho(D) \times hR_\rho(D)}{hP_\rho(D) + hR_\rho(D)} \right\}. \tag{5.15}$$

Thanks to the choice to adopt a score cutoff, CAFA's hierarchical F-measure is not affected by the unbalanced number of annotations in the benchmark set.

Information-Theoretic metrics.

These metrics can be considered the information-theoretic counterparts of precision/recall, and rely on the information content of individual terms t in the GO hierarchy:

$$ic(t) = -\log Pr(t)$$

where $Pr(t)$ is the relative frequency of term t with respect to some background distribution. We adopt as background distribution the entire set of gold-standard annotations in our benchmark set. The Resnik's similarity (Resnik, 1995) between two terms t and p is defined as the maximum information content among the common ancestors of t and p:

$$sim_{Resnick}(t,p) = \max_{a\in\mathcal{A}(\{t\})\cap\mathcal{A}(\{p\})} \{ic(a)\}$$

The Lin's similarity (Lin, 1998) between two terms t and p is the normalized version of the Resnick's similarity:

$$sim_{Lin}(t,p) = \frac{2 \times sim_{Resnick}(t,p)}{ic(t) + ic(p)}$$

Following the approach in (Schlicker et al., 2006), we can extend the information-theoretic similarities to sets of annotations. In this way, it is possible to obtain the information-theoretic counterpart of precision at the top-k ranked terms:

$$iP_k(T, P) = \frac{1}{|\mathcal{L}(P_k)|} \sum_{p \in \mathcal{L}(P_k)} \max_{t \in \mathcal{L}(T)} \{sim_{Lin}(t, p)\}, \tag{5.16}$$

and recall at the top-k ranked terms:

$$iR_k(T, P) = \frac{1}{|\mathcal{L}(T)|} \sum_{t \in \mathcal{L}(T)} \max_{p \in \mathcal{L}(P_k)} \{sim_{Lin}(t, p)\} \tag{5.17}$$

where $\mathcal{L}(X)$ denotes the set of *leaf terms* in X, i,e. $\mathcal{L}(X) \subseteq X$ is the largest subset of X such that $\forall u, v \in \mathcal{L}(X), u \notin \mathcal{A}(v)$. As it happens for the hierarchical-precision hP_{10}, the information-theoretic precision iP_{10} is slightly affected by the unbalanced number of annotations in our benchmark set. Conversely, the iP_1 metric is more robust, since it just estimates the quality of the very top predicted annotation. For these motivations, here we consider only the information-theoretic precision at the top ranked term (iP_1) and the information-theoretic recall at rank 10 (iR_{10}), which can be seen as complementary to the two TREC metrics MRR_{10} and R_{10}.

5.5.3 Experimental Results and Discussions

In the following sections, performances are assessed over the entire GO hierarchy, without considering separately the three main ontologies BP, MF and CC. As a general consideration, the classification accuracy is overall better if we asses the results separately for the three ontologies. Furthermore, it is higher over the MF and CC in comparison to BP. These results are not completely unexpected, since the entire GO hierarchy contains much more categories/terms than its main sub-ontologies. Equivalently, the number of distinct categories is much lower in MF and CC than in BP. Furthermore, also baseline random classifiers have surprisingly good classification performances over MF and CC. This may suggest that there is some manual-annotation bias toward few specific MF and CC terms. Conversely, the performance gap between random and non-random classifiers is much more pronounced over the entire GO hierarchy. The results over the complete GO are thus more representative of the true classification limits of the different approaches.

Table 5.8 – *Performances over a test set of 15,000 publications*

Method[a]	Info[b]	IT[c]		CAFA[c]	BC[c]	TREC[c]	
		iP_1	iR_{10}	hF_{max}	hR_{10}	MRR_{10}	R_{10}
GOTA	PM	**0.43**	**0.64**	**0.43**	**0.69**	**0.40**	**0.46**
GOTA	T+A	0.42	**0.64**	**0.43**	0.68	0.39	0.45
GOTA	T	0.41	0.63	0.42	0.68	0.39	0.44
RandFR	N/A	0.20	0.33	0.20	0.33	0.18	0.15
RandIC	N/A	0.21	0.27	0.18	0.31	0.03	0.08
GOTA Φ_P	PM	0.37	0.64	0.41	0.67	0.38	0.44
GOTA Φ_P	T+A	0.35	0.62	0.40	0.66	0.36	0.41
GOTA Φ_P	T	0.35	0.62	0.40	0.66	0.36	0.41
GOTA Φ_T	PM	0.28	0.41	0.30	0.49	0.16	0.17
GOTA Φ_T	T+A	0.24	0.37	0.27	0.46	0.11	0.12
GOTA Φ_T	T	0.22	0.35	0.26	0.44	0.09	0.10

[a] Method used for the classification. RandFR and RandIC are baseline predictors, based on the distribution of GO terms in the training set.
[b] Informations used in prediction: PM = title, abstract, references and publication year (PubMed); T+A = title and abstract; T = title; N/A = no information.
[c] Metrics definitions are in the Evaluation metric Section. In top section of the table, for each metric, the best result is highlighted in bold.

Overall classification performances

Classification performances on our test set of 15,000 publications (see Benchmark set Section) are shown in the top section of Table 5.8. In order to clarify how the classification is affected by different sources of information, in Table 5.8 we show the performances of our best-parameter model, which makes use of the detailed information available in PubMed (PM), and those of two more versions that make use of unstructured text only: title plus abstract (T+A), and only title (T). In these last two models, the classifier is queried with unstructured text and it does not know whether the input text is related to the publication title and/or abstract. We include in the comparison also two naive baseline predictors obtained from the distribution of GO terms in our KB. Both naive predictors assign exactly the same predictions to all query targets. The first naive predictor, RandFR, ranks the GO terms according to their relative frequency in the underlying KB. The second naive predictor, RandIC, ranks each GO term according to its average information-theoretic precision (Eq. 5.16), with respect to all the publications in the underlying KB.

As shown in Table 5.8, GOTA classification capabilities are much better than those of the two naive predictors, over all considered metrics. By observing in more detail the best performances (first row) in Table 5.8, according to the MRR_{10} metric, on the average a prediction includes a gold standard annotation at rank between 2 and 3. According to the R_{10} metric, 46% of the gold standard annotations are included in the top-10 predicted terms. Although not directly comparable, these results are better that those reported at BioCreative IV Task B (Mao et al., 2014) and for other large hierarchical multi-label classification settings (Tao et al., 2012). The information-theoretic measures provide more interesting insights into the classification capabilities. According to the iR_{10} metric, the average semantic similarity between the gold standard annotations with respect to the top-10 predicted terms is 0.64. According to the iP_1 measure, the average semantic similarity of the top predicted term is 0.46. By themselves these scores say little about the quality of the predicted annotations. However, we can easily calculate some statistics to asses whether these values are meaningful or not. For 72% of the gold-standard terms (represented in the test set) there are less than 1% terms (represented in the KB) with semantic similarity ≥ 0.46. That is, two terms with semantic similarity ≥ 0.46 are very likely to be closely related. We also bootstrapped (10,000 draws of 10 terms each from the KB) the sample mean and standard deviation of the iR_{10} and iP_1 scores for each test-publication. At a confidence level of 10^{-5}, GOTA's iR_{10} and iP_1 scores are significantly higher (according to a Z-test) than the samples means for 88% and 57% of the test publications, respectively. The conclusion is that, on the average: i) the top-10 predicted terms include GO terms that are significantly similar to the gold standard annotations. ii) such classifications are very unlikely to be observed by random selection of GO terms from the KB. With respect to the hR_{10} metric, we can observe a satisfactory performance (although not directly comparable) with respect to the results reported at BioCreative IV Task B (Mao et al., 2014). Finally, although protein and literature annotation are different but related problems, the hF_{max} measurements for BP and MF are higher in comparison to the best performing methods at CAFA (Radivojac et al., 2013). With respect to CAFA, we can report better results also if we restrict to subsets related to specific species. This may suggest that the automated GO annotation task from protein sequences is an even harder problem than literature-based annotation.

Although the classifier that makes use of more detailed information (PM) has the best performances, its classification capabilities are not much higher than those of two models that make use of unstructured text only (T+A and T). We can asses whether the improvement is statistically significant with a paired Student's t-test over all selected metrics but hF_{max} (which is not an averaged score). At standard significance level of 5%, the performances of the full model (PM) are significantly better over all metrics in comparison to the title plus abstract-based model (T+A) and the title-based model (T). However, the performances of the two text-based classifiers are indistinguishable for some of the adopted metrics (data not shown). This is somewhat surprising and suggests that publication title is the most important source of information for GO annotation of scientific literature.

Another interesting question is whether authorship can introduce some bias in the experimental testing. In particular, we ask to which extent having in test and training papers from the same author(s) can affect the classification performance. In this context, the authorship information can be relevant for two main reasons: i) the way in which an author writes can be repetitive in some parts and it could affect the text similarities extracted by an automatic classifier, and ii) it is conceivable that the same authors could work and publish more than one papers on similar topics. We experimented the authorship information, we omit here for space reason the results, the overall conclusion of these tests is that authorship information alone provides better results than random classification. On the other end, such information does not affect significantly GOTA's performances.

To conclude, in the bottom section of Table 5.8 we show the average performances of the two comparison scores (Φ_P and Φ_T) used in the GOTA classifier (see Approach Section). It is evident that publication-centric approach (Φ_P) is more accurate than the topic-centric approach (Φ_T). However, their combination provides overall better results. This shows that the two approaches are not significantly affected by collinearity problems. In fact, their two BoWs are built using different techniques.

Table 5.9 – *Performance comparison over species-specific knowledge bases.*

Species[a]	KB[b]	Info[c]	IT[d]		CAFA[d]	BC[d]	TREC[d]	
			iP_1	iR_{10}	hF_{max}	hR_{10}	MRR_{10}	R_{10}
Human	Human	PM	**0.45**	**0.62**	**0.46**	**0.69**	**0.49**	**0.49**
Human	Full	PM	0.44	**0.62**	0.44	**0.69**	0.44	0.48
Human	Human	T	0.42	0.60	0.45	0.66	0.46	0.47
Human	Full	T	0.44	0.61	0.44	0.68	0.45	0.47
Mouse	Mouse	PM	**0.45**	**0.63**	**0.45**	**0.67**	**0.45**	**0.44**
Mouse	Full	PM	**0.45**	0.61	0.44	0.66	0.43	0.42
Mouse	Mouse	T	0.42	**0.63**	0.44	0.65	0.43	0.42
Mouse	Full	T	0.44	0.60	0.43	0.64	0.42	0.41
Rat	Rat	PM	**0.38**	**0.64**	**0.41**	**0.69**	**0.36**	**0.44**
Rat	Full	PM	0.34	0.61	0.37	0.67	0.33	0.42
Rat	Rat	T	0.37	0.62	0.40	0.67	0.34	0.42
Rat	Full	T	0.33	0.61	0.37	0.66	0.33	0.42
Yeast	Yeast	PM	**0.45**	**0.72**	**0.47**	**0.77**	**0.42**	**0.50**
Yeast	Full	PM	0.43	0.70	**0.47**	0.75	0.39	0.49
Yeast	Yeast	T	0.41	0.68	0.44	0.74	0.37	0.45
Yeast	Full	T	0.41	0.68	0.44	0.73	0.35	0.46

[a] Only publications related to the specified Species are considered for the evaluation.
[b] Knowledge base used for prediction. Full = all available publications in the KB. Human/Mouse/Rat/Yeast = only related publications.
[c] Informations used in prediction: PM = title, abstract, references and publication year (PubMed); T = title.
[d] Metrics definitions are in the Evaluation metric Section.

Performances on species-specific knowledge bases

A natural question concerning GO annotation of literature is whether we can improve the classification accuracy by restricting to species-specific knowledge bases. For instance, if we know a-priori that some publication is related to some specific species, for annotation purposes is it better to compare it with a smaller set of publications on the same species or with a larger set of publications on different species?

In order to answer this question, here we considered four species for which we found a sufficiently high number of annotated publications in GOA database: Human (27,133 distinct publications, 24% of the entire benchmark set), Mouse (21,581, 19%), Rat (18,321, 16%) and Yeast (10,162, 9%). We extracted from our test set of 15,000 publications the subsets related to the selected species. For each one of such subsets we compare the performances we obtain when the underlying KB is the same as in previous Section (Full) and when it consists of species-specific subsets only (Human, Mouse, Rat and Yeast). The species-specific subsets have been constructed by selecting from the KB only the PMID-GO associations listed in the species-related GOA files. The results are summarized in Table 5.9, where we can observe that the classification performances over species-specific KBs are overall better than those on the full KB. However, the improvement is statistically significant (as assessed by a paired t-test at significance level of 5%) over almost all the adopted metrics only on the set of publications related to Rat and Yeast. In the remaining cases, the improvement is not uniformly significant for all the considered metrics, but only some of them. These results indicate that, in some specific cases, specie-related KB can help to improve dramatically the classification accuracy. To some extend, these results are not completely unexpected, since specie-specific KBs identify specie-specific subsets of GO terms. For instance, the rat-specific KB collects 9,090 distinct terms, in comparison to the 25,622 terms in the full KB. Thus, the classification of rat-related publications is less challenging, since the rat-specific KB drastically reduces the number of possible categories represented in the full KB.

As a final remark, as shown in Table 5.9, also for species-specific classification tasks, the publication title is the most important source of information. This is true irrespectively of the considered species.

Performance comparison with related approaches

We compare the performances of our tool with GOCat (Gobeill et al., 2013) (ML classifier), which is, to date, the only publicly available tool closely related to our work. For comparison purposes, we extracted from our test set of 15,000 publications the set of 412 PMIDs not included in the KB of GOCat's latest release. GOCat's predictions over the 412 queries have been obtained by using the publication title only (T) and publication title plus abstract (T+A), in form of unstructured text. For a fair comparison, we removed from GOCat's predictions all the terms not included in our GO hierarchy.

The results are summarized in Table 5.10. The performances of both GOTA and GOCat are significantly higher than those of the two naive

Table 5.10 – *Performance comparison with different approaches*

Method[a]	Info[b]	IT[c]		CAFA[c]	BC[c]	TREC[c]	
		iP_1	iR_{10}	hF_{max}	hR_{10}	MRR_{10}	R_{10}
GOTA	PM	**0.42**	**0.69**	**0.42**	**0.73**	**0.39**	**0.49**
GOTA	T+A	0.37	0.68	0.41	**0.73**	0.35	0.48
GOTA	T	0.39	0.66	0.39	0.70	0.34	0.44
GOCat	T+A	0.34	0.64	0.37	0.69	0.29	0.40
GOCat	T	0.30	0.64	0.36	0.69	0.28	0.40
RandFR	N/A	0.08	0.21	0.10	0.23	0.03	0.05
RandIC	N/A	0.22	0.23	0.19	0.30	0.00	0.01

[a] Method used for prediction. RandFR and RandIC are baseline predictors, based on the distribution of GO terms in the training set.
[b] Informations used in prediction: PM = title, abstract, references and publication year (PubMed); T+A = title and abstract; T = title; N/A = no information.
[c] Metrics definitions are in the Evaluation metric Section. For each metric, the best result is highlighted in bold.

classifiers RandFR and RandIC. GOTA performances with full information (PM) are significantly better then those of GOCat (T+A) over all the considered metrics, as assessed by a paired t-test at significance level of 5%.GOTA makes use of more specific information than GOCat, such as references and publication year. Anyway, even when exactly the same information is used in prediction (T+A and T), GOTA classification capabilities are significantly superior to those of GOCat for almost all the considered metrics (data not shown). As a further analysis, we calculate what is the fraction of publications for which the top-10 predictions contain at least one gold standard term. We have an a amount of 62% publications with GOTA against 52% of GOCat.

Analyzing in further detail GOTA and GOCat performances, we can verify whether they behave in similar or completely different way. In Table 5.11, for each evaluation metric (but hF_{max}) we show the fraction of publications on which GOTA gets a score exactly equal to (GOTA=GOCat), strictly higher than (GOTA>GOCat) or strictly lower than (GOTA<GOCat) GOCat's. Although GOTA's performances are overall better, from Table

Table 5.11 – *1-to-1 comparison between* GOTA *(PM) and GOCat (T+A)*

Metric[a]	GOTA=GOCat[b]	GOTA>GOCat[c]	GOTA<GOCat[d]
iP_1	0.44 (0.13)	0.36 (0.15)	0.20 (0.07)
iR_{10}	0.29 (0.16)	0.41 (0.15)	0.30 (0.07)
hR_{10}	0.41 (0.25)	0.36 (0.17)	0.23 (0.09)
MRR_{10}	0.42 (0.13)	0.37 (0.16)	0.20 (0.07)
R_{10}	0.61 (0.21)	0.25 (0.17)	0.13 (0.08)

[a] Metrics definitions are in the Evaluation metric Section.
[b] Fraction of publications on which GOTA and GOCat get exactly the same score. In parenthesis, fraction of publications on which the score is equal to 1 (maximum).
[c] Fraction of publications on which GOTA gets a score strictly higher than GOCat's. In parenthesis, fraction of publications on which the score is equal to 1 (maximum).
[d] Fraction of publications on which GOCat gets a score strictly higher than GOTA's. In parenthesis, fraction of publications on which the score is equal to 1 (maximum).

5.11 it is clear that there is a non-trivial fraction of publications on which GOCat performs better than GOTA.

Finally, interestingly enough, also with GOCat there is not a dramatic improvement in the classification capabilities when the input query consists of title and abstract (T+A) versus publication title only (T). This is a further confirmation that, independently of the particular approach, titles provide most of the information about the biological content of scientific publications. This may suggest that automated GO annotation tools could benefit from a preprocessing phase aimed at selecting short sentences within the text, in order to filter-out the background noise.

Chapter 6

Conclusions

In this dissertation, several techniques for data and text mining have been discussed. Some of them are based on a novel representation method of a data mining problem, through which were applied standard machine learning techniques. The main idea is to use a pair of objects, or a pair object-class, in each row of the representation matrix (i.e. instance of the model), instead of a single object like in the standard methods; each column (i.e. feature) is related to a relation between the paired objects. This approach has been applied in text categorization problems, both on flat and hierarchical domains, showing a potentially independence of the created models with respect the training topics, and allowing, for instance, the efficient addition of new categories during the classification phase. Furthermore, the same idea has been used to extract conversational threads from a pool of messages and to classify the biomedical literature based on the genomic features treated.

The hierarchical categorization based on this approach has resulted in an tool called Chonxi. It works pairing documents and categories and training a Semantic Relationship Scoring model, namely a probabilistic knowledge model that is independent on the document words because it employs, as features, quantitative information about WordNet semantic relationships among relevant terms rather than the specific words. Given an adequately large training set, is expected it to be topic-independent, and therefore valid even for topics missing in the training phase. With extensive experiments on well known hierarchical text datasets, Chonxi has proved to achieve an accuracy equal to or greater than the hierarchical categorization state of arts methods. Overall, the experiments show that classifying documents with CHONXI using knowledge extracted from a different domain, yields an accuracy only few percentage points below that which would be obtained by using a model extracted from the same domain. This partly supports the hypothesis of the general validity of the semantic relatedness model. It is expected that some improvements could be made with further tuning of parameters and especially by using models extracted from larger datasets, possibly providing more rich knowledge.

The application of this representation "in pairs", yielded a novel method for the automatic detection of conversational threads from an unregulated pool of any type of textual messages, for instance from social network chats, mailing list, email boxes, chats, forums etc. The problem has been addressed by using a three-dimensional representation for each message, which involved textual semantic content, social interactions and creation time. Messages were paired, creating features that describe different relationships between them, based on the three dimensions. These features were firstly combined with a simple multiplication to compute a distance measure between each pair of messages, which was used in a clustering algorithm to detect the threads. Then, a novel proposal was the use of a supervised model which combines these features using the probability to be in the same thread estimated by the model as a distance measure between two messages. Experiment on seven different datasets showed that the use of the supervised model leads to higher accuracy in thread detection, outperforming all earlier approaches.

A last application of this representation idea was a novel approach for GO annotation of biomedical literature. The resulting annotation tool, called GOTA, made use only of information that is readily available from public repositories and it is easily expandable to handle novel sources of information. GOTA is based on a combination of document-centric (i.e. pairs of biomedical papers) and topic-centric (i.e. pairs of biomedical papers and GO terms) strategies to create a similarity score between biomedical papers and GO terms, allowing the annotation (i.e. classification) of the biomedical literature. The classification accuracy of GOTA was reasonably good in comparison to the state of the art in multi-label classification over large taxonomies and the experimental tests have provided some interesting insights into the potential improvement of automated annotation tools. In particular: i) the classification capabilities improved if the approach was tested over specie-specific knowledge bases. This suggested that GO curation task could benefit from developing species-centered annotation tools; ii) the publication title was the most important source of information for GO classification biomedical literature. This resulted is a strong indication that the biological content of a paper is well-summarized in few short sentences within the text, while the rest is background noise. In this view, the suggestion is that GO annotation tools for literature could benefit from a preprocessing phase aimed at filtering-out background sentences within the text.

Another contribution of this thesis addressed cross-domain text categorization, where a knowledge model to classify documents in a target domain is obtained with the support of labeled documents of a source domain. A novel method has been proposed, based on nearest centroid classification, where a profile for each category is extracted from training documents and each new document is classified with the category having the closest profile. In the cross-domain case, such profiles are extracted from the source domain and, through successive refining steps, they are adapted to the target domain and then used to classify relevant documents. Experiments on recurring benchmark datasets produced good results, with accuracy higher or comparable to those reported by state of art works and also fast running times, all despite the relative simplicity of the approach. A variant has been proposed to cut down the number of iterations with minimal loss of accuracy, while another one has been proposed to obtain from the source domain a regression model mapping measurable similarity to effective relatedness.

A specific application of text categorization is the classification of textual opinions in positive, negative or neutral polarity, generally called sentiment classification. A novel Markov chain-based method to accomplish sentiment classification in both in-domain and cross-domain has been proposed. The main idea is to use terms and also categories (i.e. positive/negative/neutral) as nodes of the MC, creating weighted links between them based on the co-occurrences of terms in the documents. This approach has proven to accomplish a knowledge flow from the terms of a document to classify, to the other terms of the dataset and to the classes, allowing the classification of test documents in both in-domain and cross-domain problems. This approach has been proved to achieve comparable performance in term of effectiveness with the state of ar methods. Furthermore, lower parameter tuning was required than previous works, since only the pre-processing parameters need to be calibrated. Finally, in spite of having a comparable computational complexity, much fewer terms were demanded to obtain good accuracy.

A recommendation system is a tool that provides ranked lists of most suitable products or services for an user, based on the user preferences and constraints. Here has been proposed a novel job recommendation system based on exploiting known co-occurrences between the skills of a person and the job positions, which are elaborated by means of LSA to discover latent relationship between them. Furthermore, has been also showed how the same data can be used to automatically build a folksonomy of job positions by means of hierarchical clustering, in order to discover groups of related occupations. The methods have been tested using a set of 50.000 public real italian profiles extracted from LinkedIn, naturally subject to noise and inconsistencies. Concerning recommendations, a quantitative experimental evaluation trivially based on real job positions, has showed promising results, where in half of the cases the exact actual occupation of a person was within the top 50 recommended positions out of more than 2,000 possibilities. By leveraging the folksonomy of positions extracted above and looking at some specific cases we saw that, even when the exact position name was not exactly hit, homonyms and similar occupations were generally suggested. Such a recommendation system can be potentially useful to individuals seeking for occupations where their abilities can effectively be endorsed, as well as to recruiters which have to evaluate the best candidates for specific positions. Notably, the method has no parameter to be set apart from the number of recommendations to be returned, so it is simple and ready to use in practice.

A final contribution of this dissertation is related to reliably predict new functionalities for genes or proteins of an organism. The idea is to train a model to recognize the presence of novel gene annotations using the obsolete annotation profile of the gene, labeling each term of an outdated annotation profile of a gene with a label taken from an updated version of it. This approach requires two different versions of the annotation matrix to build the training data representation. However, biologists typically have available only the most updated version of the gene annotation matrix. Given this constrain, a novel method has been proposed to overcome this lack, artificially creating a new annotation matrix, representing an older version of the input one, by perturbing the known annotation matrix in order to randomly remove some of its annotations. This allowed the use of supervised algorithms even in datasets without labels and the comparison with results obtained by unsupervised methods on the same originally unlabeled datasets. Experiments on four different organisms have been encouraging, the proposed supervised method based on random perturbation has achieved accuracies always higher than the state of arts unsupervised methods. Experiments varying term weighting schemes have been made, to give different importances to each entry of the matrix (a couple gene-GO term), showing that a simple binary weighting permits to obtain better performance.

Once demonstrated the goodness of the random perturbation approach, has been considered the hypothesis of evidence conservation in gene expression or proteomic data from multiple organisms; in other words the assumption that genes sharing similar biomedical annotation profiles have similar functionalities. This allowed the use of knowledge available for well-studied and better-known organisms and transferring it to less studied one. In particular, it allowed taking advantage of the many available and relevant structured descriptions of the existing knowledge, which leverage the several controlled terminologies and ontologies that have been developed and are used to express the existing valuable biological information and knowledge in a way that lets easily and effectively using them for new discoveries.

From this assumptions, a cross-organism approach here has been proposed. The basic idea is that: the richer and denser is the training annotation set, the greater is the accuracy of the annotation prediction model. Namely prediction models have been trained using any term involved in the annotation of a source organism; this implied the possibility of discovering new annotations for the target organism to terms representing biological concepts that have newer been associated with the target organism. Through several performed evaluations, was proved that the richer knowledge of an organism can be used to build a better prediction model for other less-known organisms, which provides more accurate results than an equivalent model built using the lesser information available for the organism to be predicted. This approach is feasible because of the proposed data representation: the classification models are guided by the co-occurrences of annotation terms in the considered gene annotation profiles, indiscriminately from the genes, and thus from the organisms, in which they occur.

6.1 Future Research

A limit of the presented work, concerning both the semantic model in Chonxi and GOTA and in the cross-domain categorization method, is the centroid-based nature, which could be a cause of accuracy loss. Works on text categorization show that methods based on nearest centroid classification usually bring sub-optimal efficacy due to the fact that categories cannot usually be accurately represented with single centroids. Future works might be dedicated to the research of a more robust model for representation of categories, to be fit in the proposed approach.

For what regards text categorization method based on the domain-independent semantic model, further works may include, other than experiments on larger datasets, test of variants in different parts of the method, such as the selection of example couples and the classification process. The goal is to obtain improvements on the generality of the obtained models and on the overall accuracy of classification of documents.

Other possible directions include the application of the proposed pairs-based representation approach to different tasks, such as performing clustering of documents, by using the semantic model to measure mutual distances and relationships between documents. The distinction of multiple relatedness modalities, like distinguishing strictly related documents from those treating more general or more specific topics, may be useful in more advanced applications, such as hierarchical clustering and induction of a taxonomy of categories.

As regards the prediction of novel biological functionalities, although the results obtained are very encouraging, they could be inadequate to deal with big data sets, being constrained by memory limitations and unable to exploit parallelism. A future research could face this problem, overcoming such limitations with the development of new incremental and parallel data and text mining algorithms, according to deep learning approaches. The expected result will be to greatly expand the applications of these methods to big data, in terms of both instances and features, which represent a de facto requirement in real data sets, such as in those that associate patients, pathologies and genomics.

Bibliography

Genomic and Proteomic Knowledge Base, accessed on May 30, 2015. URL `http://www.bioinformatics.deib.polimi.it/GPKB/`.

Reactome project. Computational inferred events, accessed on May 30, 2015. URL `http://www.reactome.org/pages/documentation/electronically-inferred-events/`.

Paige H. Adams and Craig H. Martell. Topic detection and extraction in chat. In *Proceedings of the 2th IEEE International Conference on Semantic Computing (ICSC 2008), August 4-7, 2008, Santa Clara, California, USA*, pages 581–588, 2008.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer, 2012.

Fatima Al-Shahrour, Pablo Minguez, Joaquín Tárraga, Ignacio Medina, Eva Alloza, David Montaner, and Joaquín Dopazo. FatiGO+: a functional profiling tool for genomic data. Integration of functional annotation, regulatory motifs and interaction data with microarray experiments. *Nucleic Acids Res*, 35(Web Server issue):W91–W96, 2007.

IBM AlchemyAPI. http://www.alchemyapi.com/.

Jim Allen and Rolf Van der Velden. Educational mismatches versus skill mismatches: effects on wages, job satisfaction, and on-the-job search. *Oxford economic papers*, pages 434–452, 2001.

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

C Anuradha and T Velmurugan. A comparative analysis on the evaluation of classification algorithms in the prediction of students performance. *Indian Journal of Science and Technology*, 8(15), 2015.

Chidanand Apté, Fred Damerau, and Sholom M Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

Erik Aumayr, Jeffrey Chan, and Conor Hayes. Reconstruction of threaded conversations in online discussion forums. In *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*, 2011.

Libby Barak, Ido Dagan, and Eyal Shnarch. Text categorization from category name via lexical reference. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 33–36, 2009.

Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

Mathieu Bastian, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, Hyungjin Kim, Sal Uryasev, and Christopher Lloyd. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 1–8. ACM, 2014.

Jason Baumgarten and Claudia Kelly. Internet based employee/executive recruiting system and method, May 17 2001. US Patent App. 09/858,881.

Paul N. Bennett and Nam Nguyen. Refined experts: improving classification in large taxonomies. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 11–18, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-483-6. doi: 10.1145/1571941.1571946. URL `http://doi.acm.org/10.1145/1571941.1571946`.

Manuele Bicego, Pietro Lovato, Barbara Oliboni, and Alessandro Perina. Expression microarray classification using topic models. In *Proc ACM Symp Appl Comput*, pages 1516–1520. ACM, 2010.

Christian Blaschke, Eduardo A Leon, Martin Krallinger, and Alfonso Valencia. Evaluation of biocreative assessment of task 2. *BMC bioinformatics*, 6(Suppl 1):S16, 2005.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

Stephan Bloehdorn and Andreas Hotho. Boosting for text classification with semantic features. In Bamshad Mobasher, Olfa Nasraoui, Bing Liu, and Brij Masand, editors, *Advances in Web Mining and Web Usage Analysis*, volume 3932 of *Lecture Notes in Computer Science*, pages 149–166. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-47127-1.

Olivier Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res*, 32(Database issue):D267–D270, 2004.

Harold Borko and Myrna Bernick. Automatic document classification. *J. ACM*, 10(2):151–162, April 1963. ISSN 0004-5411. doi: 10.1145/321160.321165. URL `http://doi.acm.org/10.1145/321160.321165`.

Athman Bouguettaya, Qi Yu, Xumin Liu, Xiangmin Zhou, and Andy Song. Efficient agglomerative hierarchical clustering. *Expert Syst. Appl.*, 42(5):2785–2797, 2015.

James A Breaugh. Employee recruitment: Current knowledge and important areas for future research. *Human Resource Management Review*, 18(3):103–118, 2008.

James A Breaugh and Mary Starke. Research on employee recruitment: So many studies, so many remaining questions. *Journal of management*, 26(3):405–434, 2000.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Andrea Broughton, Beth Foley, Stefanie Ledermaier, and Annette Cox. The use of social media in the recruitment process. *Retrieved May*, 2:2014, 2013.

Patrick Buckley, Kathleen Minette, Dennis Joy, and Jeff Michaels. The use of an automated employment recruiting and screening system for temporary professional employees: A case study. *Human Resource Management*, 43(2-3): 233–241, 2004.

Ricardo Buettner. A framework for recommender systems in online social network recruiting: An interdisciplinary call to arms. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 1415–1424. IEEE, 2014.

Lijuan Cai and Thomas Hofmann. Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 182–189, 2003. ISBN 1-58113-646-3.

Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 78–87, 2004.

Arif Canakoglu, Giorgio Ghisalberti, and Marco Masseroli. Integration of genomic, proteomic and biomolecular interaction data to support biomedical knowledge discovery. In *Proc Int Meet Comput Intell Methods Bioinforma Biostat (CIBB 2011)*, volume 8, pages 1–10, 2011, 6.

Arif Canakoglu, Giorgio Ghisalberti, and Marco Masseroli. Integration of biomolecular interaction data in a genomic and proteomic data warehouse to support biomedical knowledge discovery. In *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 112–126. Springer, 2012.

George Casella and Edward I George. Explaining the Gibbs sampler. *Am Stat*, 46(3):167–174, 1992.

William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

Michelangelo Ceci and Donato Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78, 2007.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Hierarchical classification: combining Bayes with SVM. In *Proceedings of the 23rd international conference on Machine learning*, pages 177–184, 2006a.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54, 2006b.

Liangxi Cheng, Hongfei Lin, Yuncui Hu, Jian Wang, and Zhihao Yang. Gene function prediction based on the Gene Ontology hierarchical structure. *PloS One*, 9(9):e107187, 2014.

Chia-Fen Chi. A study on job placement for handicapped workers using job analysis data. *International Journal of Industrial Ergonomics*, 24 (3):337 – 351, 1999. ISSN 0169-8141. doi: http://dx.doi.org/10.1016/S0169-8141(98)00041-9. URL `http://www.sciencedirect.com/science/article/pii/S0169814198000419`.

Davide Chicco and Marco Masseroli. A discrete optimization approach for SVD best truncation choice based on ROC curves. In *Proc IEEE Int Conf Bioinformatics Bioeng (BIBE 2013)*, pages 1–4. IEEE, 2013, 9.

Vikas Chirumamilla, Sruthi T Bhagya, Velpula Sasidhar, and Sunkara Indira.

Novel approach to predict student placement chance with decision tree induction. *nternational Journal of Systems and Technologies*, 7(1):78–88, 2014.

Kristof Coussement and Dirk Van den Poel. Improving customer complaint management by automatic email classification using linguistic style features as predictors. *Decision Support Systems*, 44(4):870–882, 2008.

David Croft, Antonio Fabregat Mundo, Robin Haw, Marija Milacic, Joel Weiser, Guanming Wu, Michael Caudy, Phani Garapati, Marc Gillespie, Maulik R Kamdar, et al. The Reactome pathway knowledgebase. *Nucleic Acids Res*, 42 (Database issue):D472–D477, 2014.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM, 2007a.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the AAAI '07, 22nd national conference on Artificial intelligence*, pages 540–545, 2007b.

Stephen D'Alessio, Keitha Murray, Robert Schiaffino, and Aaron Kershenbaum. The effect of using hierarchical classifiers in text categorization. In *Proceedings of 6th International Conference Recherche d'Information Assistee par Ordinateur*, pages 302–313, 2000.

Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, 2007.

H Kristl Davison, Catherine Maraist, and Mark N Bing. Friend or foe? the promise and pitfalls of using social networking sites for hr decisions. *Journal of Business and Psychology*, 26(2):153–159, 2011.

Olivier De Vel, Alison Anderson, Malcolm Corney, and George Mohay. Mining e-mail content for author identification forensics. *ACM Sigmod Record*, 30(4): 55–64, 2001.

Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Proceedings of the 18th ACM Symposium on Applied Computing*, pages 784–788, 2003.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41 (6):391–407, 1990.

Mostafa Dehghani, Azadeh Shakery, Masoud Asadpour, and Arash Koushkestani. A learning approach for email conversation thread reconstruction. *J. Information Science*, 39(6):846–863, 2013.

C Deisy, M Gowri, S Baskar, SMA Kalaiarasi, and N Ramraj. A novel term weighting scheme midf for text categorization. *Journal of Engineering Science and Technology*, 5(1):94–107, 2010.

Xutao Deng and Hesham Ali. A hidden markov model for gene function prediction from sequential expression data. In *Proc IEEE Comput Sys Bioinform Conf*, pages 670–671. IEEE, 2004.

Zhi-Hong Deng, Shi-Wei Tang, Dong-Qing Yang, Ming Zhang Li-Yu Li, and Kun-Qing Xie. A comparative study on feature weight in text categorization. In *Advanced Web Technologies and Applications*, pages 588–597. Springer, 2004.

Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923, October 1998. ISSN 0899-7667. doi: 10.1162/089976698300017197. URL `http://dx.doi.org/10.1162/089976698300017197`.

Kumar A Dinesh and V Radhika. A survey on predicting student performance. *International Journal of Computer Science and Information Technologies*, 5 (5):6147–9, 2014.

Giacomo Domeniconi, Marco Masseroli, Gianluca Moro, and Pietro Pinoli. Discovering new gene functionalities from random perturbations of known gene ontological annotations. In *KDIR 2014 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Rome, Italy, 21 - 24 October, 2014*, pages 107–116, 2014a. doi: 10.5220/0005087801070116. URL `http://dx.doi.org/10.5220/0005087801070116`.

Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. Cross-domain text classification through iterative refining of target categories representations. In *KDIR 2014 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Rome, Italy, 21 - 24 October, 2014*, volume 553 of *Communications in Computer and Information Science*, pages 31–42. Springer, 2014b. doi: 10.5220/0005069400310042. URL `http://dx.doi.org/10.5220/0005069400310042`.

Giacomo Domeniconi, Marco Masseroli, Gianluca Moro, and Pietro Pinoli. Random perturbations of term weighted gene ontology annotations for discovering gene unknown functionalities. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management - 6th International Joint Conference, IC3K 2014, Rome, Italy, October 21-24, 2014, Revised Selected Papers*, pages 181–197, 2015a. doi: 10.1007/978-3-319-25840-9_12. URL `http://dx.doi.org/10.1007/978-3-319-25840-9_12`.

Giacomo Domeniconi, Gianluca Moro, Andrea Pagliarani, and Roberto Pasolini. Markov chain based method for in-domain and cross-domain sentiment classification. In *Proceedings of the 7th International Joint Conference on Knowledge*

Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), volume 1: KDIR, pages 127–137, 2015b.

Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. A study on term weighting for text categorization: A novel supervised variant of tf.idf. In *DATA 2015 - Proceedings of 4th International Conference on Data Management Technologies and Applications, Colmar, Alsace, France, 20-22 July, 2015.*, pages 26–37. SciTePress, 2015c. doi: 10.5220/0005511900260037. URL `http://dx.doi.org/10.5220/0005511900260037`.

Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. Iterative refining of category profiles for nearest centroid cross-domain text classification. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management - 6th International Joint Conference, IC3K 2014, Rome, Italy, October 21-24, 2014, Revised Selected Papers*, volume 553 of *Communications in Computer and Information Science*, pages 50–67. Springer, 2015d. doi: 10.1007/978-3-319-25840-9_4. URL `http://dx.doi.org/10.1007/978-3-319-25840-9_4`.

Andreas Doms and Michael Schroeder. Gopubmed: exploring pubmed with the gene ontology. *Nucleic acids research*, 33(suppl 2):W783–W786, 2005.

Bogdan Done, Purvesh Khatri, Arina Done, and Sorin Draghici. Semantic analysis of genome annotations using weighting schemes. In *Proc IEEE Symp Comput Intell Bioinforma Comput Biol (CIBCB 2007)*, pages 212–218. IET, 2007.

Bogdan Done, Purvesh Khatri, Arina Done, and Sorin Draghici. Predicting novel human gene ontology annotations using semantic analysis. *IEEE/ACM Trans Comput Biol Bioinform*, 7(1):91–99, 2010.

Louis du Plessis, Nives Škunca, and Christophe Dessimoz. The what, where, how and why of gene ontologya primer for bioinformaticians. *Briefings in bioinformatics*, page bbr002, 2011.

Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263, 2000.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, 1998. ISBN 1-58113-061-9.

Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proc ACM SIGCHI Conf Hum Factors Comput Syst*, pages 281–285. ACM, 1988.

Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In *Advances in neural information processing systems*, pages 385–392, 2008.

Andreas Eckhardt, Sven Laumer, and Tim Weitzel. Extending the architecture for a next-generation holistic e-recruiting system. In *CONF-IRM 2008 Proceedings*, page 27, 2008.

Sudheep Elayidom, Sumam Mary Idikkula, and Joseph Alexander. A generalized data mining framework for placement chance prediction problems. *International Journal of Computer Applications (0975–8887) Volume*, 2011.

Shai Erera and David Carmel. Conversation detection in email systems. In *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*, pages 498–505, 2008.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, 1996.

L. Shuler T. Wu F. M. Khan, T. A. Fisher and W. M. Pottenger. Mining chatroom conversations for social and semantic interactions. In *Technical Report LU-CSE-02-011, Lehigh University*, 2002.

Marco Falda, Stefano Toppo, Alessandro Pescarolo, Enrico Lavezzo, Barbara Di Camillo, Andrea Facchinetti, Elisa Cilia, Riccardo Velasco, and Paolo Fontana. Argot2: a large scale function prediction tool relying on semantic similarity of weighted Gene Ontology terms. *BMC Bioinformatics*, 13(Suppl 4):S14, 2012.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

Pedro Fialho, Sérgio Curto, Ana Cristina Mendes, and Luısa Coheur. Wordnet framework improvements for NLP: Defining abstraction and scalability layers. Technical report, Technical Report 8113, Spoken Language Systems Lab, Technical University of Lisbon, 2011.

BJ Field. Towards automatic indexing: automatic assignment of controlled-language indexing and classification from free indexing. *Journal of Documentation*, 31(4):246–265, 1975.

Mark Alan Finlayson. Java libraries for accessing the princeton wordnet: Comparison and evaluation. In *Proceedings of the 7th Global Wordnet Conference, Tartu, Estonia*, 2014.

Lena Katharina Flecke. Utilizing facebook, linkedin and xing as assistance tools for recruiters in the selection of job candidates based on the person-job fit. 2015.

George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305, 2003.

William J Frawley, Gregory Piatetsky-Shapiro, and Christopher J Matheus. Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57, 1992.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput and Syst Sci*, 55(1): 119–139, 1997.

Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1048–1053, 2005.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611, 2007.

Luigi Galavotti, Fabrizio Sebastiani, and Maria Simi. Experiments on the use of feature selection and negative evidence in automated text categorization. In *Research and Advanced Technology for Digital Libraries*, pages 59–68. Springer, 2000.

Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 283–291. ACM, 2008.

Gene Ontology Consortium et al. Creating the gene ontology resource: design and implementation. *Genome res*, 11(8):1425–1433, 2001.

Kristin Glass and Richard Colbaugh. Toward emerging topic detection for business intelligence: Predictive analysis of `meme' dynamics. *CoRR*, abs/1012.5994, 2010.

Julien Gobeill, Emilie Pasche, Dina Vishnyakova, and Patrick Ruch. Managing the data deluge: data-driven go category assignment improves while complexity of functional annotation increases. *Database*, 2013:bat041, 2013.

Julien Gobeill, Emilie Pasche, Dina Vishnyakova, and Patrick Ruch. Closing the loop: from paper to protein annotation using supervised gene ontology classification. *Database*, 2014:bau088, 2014.

Tom Griffiths. Gibbs sampling in the generative model of Latent Dirichlet Allocation. *Standford University*, 518(11):1–3, 2002.

Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Using kNN model for automatic text categorization. *Soft Comput.*, 10(5):423–430, 2006. ISSN 1432-7643.

Arpan Gupta and Deepak Garg. Applying data mining techniques in job recommender system for considering candidate job preferences. In *Advances in Computing, Communications and Informatics (ICACCI), 2014 International Conference on*, pages 1458–1465. IEEE, 2014.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

Donna Harman. Overview of the first trec conference. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 36–47. ACM, 1993.

Samer Hassan and Carmen Banea. Random-walk term weighting for improved text classification. In *In Proceedings of TextGraphs: 2nd Workshop on Graph Based Methods for Natural Language Processing. ACL*, pages 53–60, 2006.

Yulan He, Chenghua Lin, and Harith Alani. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 123–131. Association for Computational Linguistics, 2011.

Lynette Hirschman, Gully AP C Burns, Martin Krallinger, Cecilia Arighi, K Bretonnel Cohen, Alfonso Valencia, Cathy H Wu, Andrew Chatr-Aryamontri, Karen G Dowell, Eva Huala, et al. Text mining for the biocuration workflow. *Database*, 2012:bas020, 2012.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression.* John Wiley & Sons, 2004.

Derek Hao Hu, Vincent Wenchen Zheng, and Qiang Yang. Cross-domain activity recognition via transfer learning. *Pervasive and Mobile Computing*, 7(3):344–358, 2011.

Da Wei Huang, Brad T Sherman, and Richard A Lempicki. Bioinformatics Enrichment tools: Paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res*, 37(1):1–13, 2009.

Jiayuan Huang, Alexander J Smola, Arthur Gretton, Karsten M Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2007.

Jiuming Huang, Bin Zhou, Quanyuan Wu, Xiaowei Wang, and Yan Jia. Contextual correlation based thread detection in short text message streams. *J. Intell. Inf. Syst.*, 38(2):449–464, 2012.

David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–291. Springer, 1994.

Shobhit Jain and Gary D Bader. An improved method for scoring protein-protein interactions using semantic similarity within the Gene Ontology. *BMC Bioinformatics*, 11(1):562, 2010.

Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 143–151, 1997.

Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.

Pawel Jurczyk and Eugene Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 919–922, 2007.

Kimberley Kasper. Half of all job seekers arent in it for the long haul, jobvite job seeker nation study shows, 2015. Retrieved from: http://www.jobvite.com/press-releases/2015/half-job-seekers-arent-long-haul-jobvite-job-seeker-nation-study-shows/, 09-09-2015.

Purvesh Khatri, Bogdan Done, Archana Rao, Arina Done, and Sorin Draghici. A semantic analysis of the annotations of the human genome. *Bioinformatics*, 21(16):3416–3421, 2005.

Tony Kinder. The use of the internet in recruitmentcase studies from west lothian, scotland. *Technovation*, 20(9):461–475, 2000.

Oliver D King, Rebecca E Foulger, Selina S Dwight, James V White, and Frederick P Roth. Predicting gene function from patterns of annotation. *Genome Res*, 13(5):896–904, 2003.

Man Lan, Sam-Yuan Sung, Hwee-Boon Low, and Chew-Lim Tan. A comparative study on term weighting schemes for text categorization. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 1, pages 546–551. IEEE, 2005.

Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):721–735, 2009. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.110.

Christine Largeron, Christophe Moulin, and Mathias Géry. Entropy based feature selection for text categorization. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 924–928, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982389. URL `http://doi.acm.org/10.1145/1982185.1982389`.

In Lee. An architecture for a next-generation holistic e-recruiting system. *Communications of the ACM*, 50(7):81–85, 2007.

Pietro Lena, Giacomo Domeniconi, Luciano Margara, and Gianluca Moro. Gota: Go term annotation of biomedical literature. *BMC Bioinformatics*, 16(1):346, 2015. ISSN 1471-2105.

David D Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50. ACM, 1992.

David D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 246–254, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6. doi: 10.1145/215206.215366. URL `http://doi.acm.org/10.1145/215206.215366`.

Suzanna E Lewis. Gene ontology: looking backwards and forwards. *Genome Biol*, 6(1):103, 2005.

Donghui Li, Tanya Z Berardini, Robert J Muller, and Eva Huala. Building an efficient curation workflow for the arabidopsis literature corpus. *Database*, 2012:bas047, 2012a.

Lianghao Li, Xiaoming Jin, and Mingsheng Long. Topic correlation analysis for cross-domain text classification. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012b.

Xiao Li, Da Kuang, and Charles X. Ling. Active learning for hierarchical text classification. In *Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, volume 1, pages 14–25, 2012c.

Xin Li, Zhu Zhang, Hsinchun Chen, and Jiexun Li. Graph kernel-based learning for gene function prediction from gene interaction network. In *Proc IEEE Int Conf Bioinformatics Biomed (BIBM 2007)*, pages 368–373. IEEE, 2007.

Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

Xiao Ling, Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Spectral domain-transfer learning. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 488–496. ACM, 2008a.

Xiao Ling, Gui-Rong Xue, Wenyuan Dai, Yun Jiang, Qiang Yang, and Yong Yu. Can chinese web pages be classified with english data source? In *Proceedings of the 17th international conference on World Wide Web*, pages 969–978. ACM, 2008b.

Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations Newsletter*, 7(1):36–43, 2005.

YC Liu, XL Wang, ZM Xu, and Yi Guan. A survey of document clustering. *Journal of Chinese Information Processing*, 20(3):55–62, 2006.

Ying Liu, Han Tong Loh, and Aixin Sun. Imbalanced text classification: A term weighting approach. *Expert Syst. Appl.*, 36(1):690–701, January 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2007.10.042. URL `http://dx.doi.org/10.1016/j.eswa.2007.10.042`.

Jane Lomax. Get ready to go! a biologist's guide to the gene ontology. *Briefings in bioinformatics*, 6(3):298–304, 2005.

H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1(4):309–317, October 1957. ISSN 0018-8646. doi: 10.1147/rd.14.0309. URL `http://dx.doi.org/10.1147/rd.14.0309`.

Hans Peter Luhn. A business intelligence system. *IBM Journal of Research and Development*, 2(4):314–319, 1958.

Qiming Luo, Enhong Chen, and Hui Xiong. A semantic term weighting scheme for text categorization. *Expert Syst. Appl.*, 38(10):12708–12716, September 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.04.058. URL `http://dx.doi.org/10.1016/j.eswa.2011.04.058`.

Jochen Malinowski, Tobias Keim, Oliver Wendt, and Tim Weitzel. Matching people and jobs: A bilateral recommendation approach. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 6, pages 137c–137c. IEEE, 2006.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Yuqing Mao, Kimberly Van Auken, Donghui Li, Cecilia N Arighi, Peter McQuilton, G Thomas Hayman, Susan Tweedie, Mary L Schaeffer, Stanley JF Laulederkind, Shur-Jen Wang, et al. Overview of the gene ontology task at biocreative iv. *Database*, 2014:bau086, 2014.

Melvin Earl Maron. Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417, 1961.

Marco Masseroli. Management and analysis of genomic functional and phenotypic controlled annotations to support biomedical investigation and practice. *IEEE Trans Inf Technol Biomed*, 11(4):376–385, 2007.

Marco Masseroli, Dario Martucci, and Francesco Pinciroli. GFINDer: Genome Function INtegrated Discoverer through dynamic annotation, statistical analysis, and mining. *Nucleic Acids Res*, 32(Web Server issue):W293–W300, 2004.

Marco Masseroli, Marco Tagliasacchi, and Davide Chicco. Semantically improved genome-wide prediction of Gene Ontology annotations. In *Proc IEEE Int Conf Intell Syst Design App (ISDA 2011)*, pages 1080–1085. IEEE, 2011.

Marco Masseroli, Davide Chicco, and Pietro Pinoli. Probabilistic Latent Semantic Analysis for prediction of Gene Ontology annotations. In *Proc Int Joint Conf Neural Netw (IJCNN 2012)*, pages 2891–2898. IEEE, 2012.

Andrew McCallum. Text classification by bootstrapping with keywords, EM and shrinkage. In *ACL99 Workshop for Unsupervised Learning in Natural Language Processing*, pages 52–58, 1999.

Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*, volume 752, pages 41–48, 1998.

Robert J McGovern, James A Winchester Jr, Andrew B Evans, Brian E Farmer, Jennie A Koffman, and Aaron P Walker. Employment recruiting system and method using a computer network for posting job openings and which provides for automatic periodic searching of the posted job openings, April 9 2002. US Patent 6,370,510.

Gina J Medsker, Larry J Williams, and Patricia J Holahan. A review of current practices for evaluating causal models in organizational behavior and human resources management research. *Journal of Management*, 20(2):439–464, 1994.

Huaiyu Mi, Anushya Muruganujan, and Paul D Thomas. PANTHER in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees. *Nucleic Acids Res*, 41(Database issue):D377–D386, 2013.

George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

Hokey Min and Ahmed Emam. Developing the profiles of truck drivers for their successful recruitment and retention: a data mining approach. *International Journal of Physical Distribution & Logistics Management*, 33(2):149–162, 2003.

Thomas P Minka. A comparison of numerical optimizers for logistic regression. `http://research.microsoft.com/en-us/um/people/minka/papers/logreg/`, 2003.

Federico Minneci, Damiano Piovesan, Domenico Cozzetto, and David T Jones. FFPred 2.0: Improved homology-independent prediction of Gene Ontology terms for eukaryotic protein sequences. *PloS One*, 8(5):e63754, 2013.

Nicholas Mitsakakis, Zak Razak, Michael D Escobar, and J Tim Westwood. Prediction of *Drosophila melanogaster* gene function using Support Vector Machines. *BioData Min*, 6(1):8, 2013.

Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.

Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 1320–1326, 2010.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Sinno Jialin Pan, James T Kwok, and Qiang Yang. Transfer learning via dimensionality reduction. In *Proceedings of the AAAI '08, 23rd national conference on Artificial intelligence*, pages 677–682, 2008.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM, 2010.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 2004.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

Ioannis Paparrizos, B Barla Cambazoglu, and Aristides Gionis. Machine learned job recommendation. In *Proceedings of the fifth ACM Conference on Recommender Systems*, pages 325–328. ACM, 2011.

Xiaogang Peng and Ben Choi. Document classifications based on word semantic hierarchies. In *Proceedings of the International Conference on Artificial Intelligence and Applications*, pages 362–367, 2005.

Antonio J Pérez, Carolina Perez-Iratxeta, Peer Bork, Guillermo Thode, and Miguel A Andrade. Gene annotation from scientific literature using mappings between keyword systems. *Bioinformatics*, 20(13):2084–2091, 2004.

Alessandro Perina, Pietro Lovato, Vittorio Murino, and Manuele Bicego. Biologically-aware latent dirichlet allocation (balda) for the classification of expression microarray. In *Pattern Recognition in Bioinformatics*, pages 230–241. Springer, 2010.

Catia Pesquita, Daniel Faria, Andre O Falcao, Phillip Lord, and Francisco M Couto. Semantic similarity in biomedical ontologies. *PLoS Comput Biolog*, 5 (7):e1000443, 2009.

Robert M Peterson and Howard F Dover. Building student networks with linkedin: The potential for connections, internships, and jobs. *Marketing Education Review*, 24(1):15–20, 2014.

Owen Phelan, Kevin McCarthy, and Barry Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388. ACM, 2009.

P. Pinoli, D. Chicco, and M. Masseroli. Weighting scheme methods for enhanced genome annotation prediction. In *Computational Intelligence Methods for Bioinformatics and Biostatistics (CIBB), 2013 10th International Meeting on*, pages 76–89. LNBI, Springer International Publishing, 2014a.

Pietro Pinoli, Davide Chicco, and Marco Masseroli. Enhanced probabilistic latent semantic analysis with weighting schemes to predict genomic annotations. In *Proc IEEE Int Conf Bioinformatics Bioeng (BIBE 2013)*, pages 1–4. IEEE, 2013, 92.

Pietro Pinoli, Davide Chicco, and Marco Masseroli. Latent Dirichlet Allocation based on Gibbs Sampling for gene function prediction. In *Proc IEEE Symp Comput Intell Bioinforma Comput Biol (CIBCB 2014)*, pages 1–8. IEEE, 2014b.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed Gibbs sampling for latent Dirichlet allocation. In *Proc ACM SIGKDD Int Conf Knowl Discov Data Min (KDDM 2008)*, pages 569–577. ACM, 2008.

Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

Peter Prettenhofer and Benno Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127, 2010.

Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys*, 41(2):12:1–12:31, 2009.

Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, et al. A large-scale evaluation of computational protein function prediction. *Nature methods*, 10(3):221–227, 2013.

Rachael Rafter, Keith Bradley, and Barry Smyth. Personalised retrieval for online recruitment services. In *The BCS/IRSG 22nd Annual Colloquium on Information Retrieval (IRSG 2000), Cambridge, UK, 5-7 April, 2000*, 2000.

V Ramesh, P Parkavi, and P Yasodha. Performance analysis of data mining techniques for placement chance prediction. *International Journal of Scientific & Engineering Research*, 2(8):1, 2011.

Soumya Raychaudhuri, Jeffrey T Chang, Patrick D Sutphin, and Russ B Altman. Associating genes with gene ontology codes using a maximum entropy analysis of biomedical literature. *Genome Res*, 12(1):203–214, 2002.

Fuji Ren and Mohammad Golam Sohrab. Class-indexing-based term weighting for automatic text classification. *Inf. Sci.*, 236:109–125, July 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2013.02.029. URL `http://dx.doi.org/10.1016/j.ins.2013.02.029`.

Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

Jorge Ropero, Ariel Gómez, Alejandro Carrasco, and Carlos León. A fuzzy logic intelligent agent for information extraction: Introducing a new fuzzy logic-based term weighting scheme. *Expert Systems with Applications*, 39(4): 4567–4581, 2012.

Daniel L Rubin, Nigam H Shah, and Natalya F Noy. Biomedical ontologies: a functional perspective. *Briefings in bioinformatics*, 9(1):75–90, 2008.

Robert S Rubin, William H Bommer, and Timothy T Baldwin. Using extracurricular activity as an indicator of interpersonal skill: Prudent evaluation or recruiting malpractice? *Human Resource Management*, 41(4):441–454, 2002.

Miguel E. Ruiz and Padmini Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5:87–118, 2002.

G. Salton. *The SMART Retrieval System&#8212;Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

Andreas Schlicker, Francisco S Domingues, Jörg Rahnenführer, and Thomas Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC bioinformatics*, 7(1):302, 2006.

Andreas Schlicker, Thomas Lengauer, and Mario Albrecht. Improving disease gene prioritization using the semantic similarity of Gene Ontology terms. *Bioinformatics*, 26(18):i561–i567, 2010.

Sam Scott and Stan Matwin. Text classification using WordNet hypernyms. In *Use of WordNet in natural language processing systems: Proceedings of the conference*, pages 38–44, 1998.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. Thread detection in dynamic text message streams. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 35–42, 2006.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Jr. Carlos N. Silla and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72, January 2011. ISSN 1384-5810. doi: 10.1007/s10618-010-0175-9.

Zheng Siting, Hong Wenxing, Zhang Ning, and Yang Fan. Job recommender systems: a survey. In *Computer Science & Education (ICCSE), 2012 7th International Conference on*, pages 920–924. IEEE, 2012.

Nives Škunca, Adrian Altenhoff, and Christophe Dessimoz. Quality of computationally inferred gene ontology annotations. *PLoS Comput. Biol*, 8(5): e1002533, 2012.

Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol*, 25(11):1251–1255, 2007.

Ian Soboroff, Arjen P. de Vries, and Nick Craswell. Overview of the TREC 2006 enterprise track. In *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, USA, November 14-17, 2006*, 2006.

Sa-Kwang Song and Sung Hyon Myaeng. A novel term weighting scheme based on discrimination power obtained from past retrieval results. *Inf. Process. Manage.*, 48(5):919–930, September 2012. ISSN 0306-4573. doi: 10.1016/j.ipm.2012.03.004. URL `http://dx.doi.org/10.1016/j.ipm.2012.03.004`.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

Daniela Stojanova, Michelangelo Ceci, Donato Malerba, and Saso Dzeroski. Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction. *BMC Bioinformatics*, 14:285, 2013.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems 20*, volume 7, pages 1433–1440, 2007.

Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 521–528, 2001.

Aixin Sun, Ee-Peng Lim, Wee-Keong Ng, and Jaideep Srivastava. Blocking reduction strategies in hierarchical text classification. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1305–1308, 2004.

J Tanoue, M Yoshikawa, and S Uemura. The GeneAround GO viewer. *Bioinformatics*, 18(12):1705–1706, 2002.

Xiaohui Tao, Yuefeng Li, RaymondY.K. Lau, and Hua Wang. Unsupervised multi-label text classification using a world knowledge ontology. In *Advances in Knowledge Discovery and Data Mining*, pages 480–492. Springer, 2012.

Ying Tao, Lee Sam, Jianrong Li, Carol Friedman, and Yves A Lussier. Information theory applied to the sparse Gene Ontology annotation network to predict novel gene function. *Bioinformatics*, 23(13):529–538, 2007.

Philip MR Tedder, James R Bradford, Chris J Needham, Glenn A McConkey, Andrew J Bulpitt, and David R Westhead. Gene function prediction using semantic similarity clustering and enrichment analysis in the malaria parasite *Plasmodium falciparum*. *Bioinformatics*, 26(19):2431–2437, 2010.

Lori Foster Thompson, Phillip W. Braddy, and Karl L. Wuensch. E–recruitment and the benefits of organizational web appeal. *Computers in Human Behavior*, 24(5):2384 – 2398, 2008. ISSN 0747-5632. doi: http://dx.doi.org/10.1016/j.chb.2008.02.014. URL `http://www.sciencedirect.com/science/article/pii/S0747563208000393`. Including the Special Issue: Internet Empowerment.

Takenobu Tokunaga and Iwayama Makoto. Text categorization based on weighted inverse document frequency. In *Special Interest Groups and Information Process Society of Japan (SIG-IPSJ*. Citeseer, 1994.

Olga G Troyanskaya, Kara Dolinski, Art B Owen, Russ B Altman, and David Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiae*). *Proc Natl Acad Sci USA*, 100(14):8348–8353, 2003.

J. Ulrich, G. Murray, and G. Carenini. A publicly available annotated corpus for supervised email summarization. In *AAAI08 EMAIL Workshop*, 2008.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73 (2):185–214, 2008.

Karin Verspoor, Judith Cohn, Susan Mniszewski, and Cliff Joslyn. A categorization approach to automated ontological function annotation. *Protein Science*, 15(6):1544–1549, 2006.

Q P Vong, K Cao, H Y Li, P A Iglesias, and Y Zheng. Chromosome alignment and segregation regulated by ubiquitination of survivin. *Science*, 310(5753): 1499–1504, 2005.

Ellen M Voorhees. Overview of the trec 2001 question answering track. *NIST special publication*, pages 42–51, 2002.

D. Wang and H. Zhang. Inverse-category-frequency based supervised term weighting schemes for text categorization. *Journal of Information Science and Engineering*, 29(2):209–225, 2013. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84876262286&partnerID=40&md5=00897fb85a83bfd704cb2428254e1950`.

Pu Wang, Carlotta Domeniconi, and Jian Hu. Using Wikipedia for co-clustering based cross-domain text classification. In *ICDM '08, 8th IEEE International Conference on Data Mining*, pages 1085–1090. IEEE, 2008a.

Yi-Chia Wang, Mahesh Joshi, William W. Cohen, and Carolyn Penstein Rosé. Recovering implicit thread structure in newsgroup style conversations. In *Proceedings of the Second International Conference on Weblogs and Social Media, ICWSM 2008, Seattle, Washington, USA, March 30 - April 2, 2008*, 2008b.

David Warde-Farley, Sylva L Donaldson, Ovi Comes, Khalid Zuberi, Rashad Badrawi, Pauline Chao, Max Franz, Chris Grouios, Farzana Kazi, Christian Tannus Lopes, et al. The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function. *Nucleic Acids Res*, 38(Web Server issue):W214–W220, 2010.

Thomas H Wonnacott and Ronald J Wonnacott. *Introductory statistics*, volume 19690. Wiley New York, 1972.

Yejun Wu and Douglas W. Oard. Indexing emails and email threads for retrieval. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 665–666, 2005.

N. Zheng X. Wang, M. Xu and N. Chen. Email conversations reconstruction based on messages threading for multi-person. In *International Workshop on Geoscience and Remote Sensing (ETTANDGRS '08)*, volume 1, pages 676–680, 2008.

Evan Wei Xiang, Bin Cao, Derek Hao Hu, and Qiang Yang. Bridging domains using world wide knowledge for transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(6):770–783, 2010.

Gui-Rong Xue, Wenyuan Dai, Qiang Yang, and Yong Yu. Topic-bridged plsa for cross-domain text classification. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 627–634. ACM, 2008a.

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international*

ACM SIGIR conference on Research and development in information retrieval, pages 619–626, 2008b.

Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263, 1995.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

Jen-Yuan Yeh. Email thread reassembly using similarity matching. In *CEAS 2006 - The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA*, 2006.

Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the 21st International Conference on Machine Learning*, page 114. ACM, 2004.

Sarah Zelikovitz and Haym Hirsh. Using LSI for text classification in the presence of background text. In *Proceedings of the 10th international conference on Information and knowledge management*, pages 113–118, 2001.

Qiankun Zhao and Prasenjit Mitra. Event detection and visualization for social text streams. In *Proceedings of the First International Conference on Weblogs and Social Media, ICWSM 2007, Boulder, Colorado, USA, March 26-28, 2007*, 2007.

Qiankun Zhao, Prasenjit Mitra, and Bi Chen. Temporal and information flow based event detection from social text streams. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1501–1506, 2007.

Yu Zheng. Methodologies for cross-domain data fusion: An overview. *IEEE Transactions on Big Data*, September 2015. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=252114`.

Fuzhen Zhuang, Ping Luo, Hui Xiong, Qing He, Yuhong Xiong, and Zhongzhi Shi. Exploiting associations between word clusters and document classes for cross-domain text categorization. *Statistical Analysis and Data Mining*, 4(1): 100–114, 2011.

Julie Zide, Ben Elman, and Comila Shahani-Denning. Linkedin and recruitment: how profiles differ across occupations. *Employee Relations*, 36(5):583–604, 2014.

George Kingsley Zipf. Human behavior and the principle of least effort. 1949.