ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
COMPUTER SCIENCE AND ENGINEERING

Ciclo 33

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Deep Networks and Knowledge: from Rule Learning to Neural-Symbolic Argument Mining

Presentata da:

Andrea Galassi

Coordinatore Dottorato
Prof. Davide Sangiorgi

Supervisore
Prof. Paolo Torroni

Co-supervisori
Prof. Marco Lippi
Prof. Michela Milano

Esame Finale Anno 2021

To my families,
my true one and the ones I have found.

Abstract

The advent of Deep Learning has revolutionized the whole discipline of machine learning, heavily impacting fields such as Computer Vision, Natural Language Processing, and other domains of computer science concerned with the processing of raw inputs. Nonetheless, Deep Networks are still difficult to interpret, and their inference process is all but transparent to the end-user. Moreover, there are still challenging tasks for Deep Networks: contexts where the success depends on structured knowledge that can not be easily provided to the networks in a standardized way.

In this thesis, we aim to investigate the behavior of Deep Networks, assessing whether they are capable of learning complex concepts such as rules and constraints without explicit information, and then how to improve them by providing such symbolic knowledge in a general and modular way.

We start by addressing two tasks: learning the rule of a game, Nine Men's Morris, and learning to construct the solution to Constraint Satisfaction Problems. We provide the networks only with examples of moves of the game and possible variables assignments, without encoding any information regarding the task nor as input nor in the networks' architecture. We observe that the networks are capable of learning to play by the rules and to make feasible assignments in the CSPs.

Then, we move to Argument Mining, a complex NLP task which consists of finding the argumentative elements in a document and identifying their relationships. We deeply analyze Neural Attention, a mechanism widely used in NLP to improve networks' performance and interpretability, providing a taxonomy of its implementations. We exploit such a method to train an ensemble of deep residual networks and test them on four different corpora for Argument Mining. Our approach obtains satisfactory results, reaching or advancing the state of the art in most of the datasets we considered for this study.

Finally, we realize the first implementation of neural-symbolic argument mining. We use the Logic Tensor Networks framework to introduce logic rules during the training process and establish that they give a positive contribution under multiple dimensions.

Table of contents

List of figures

List of tables

List of publications

Part of the work in this thesis has previously appeared in the following works.
I hereby declare to have obtained the permission to use this material as part of my thesis.

[36] Chesani, F., Galassi, A., Lippi, M., and Mello, P. (2018). Can deep networks learn to play by the rules? A case study on nine men's morris. *IEEE Transactions on Games*, 10(4):344-353. IEEE. © 2018 IEEE.

[78] Galassi, A., Lombardi, M., Mello, P., and Milano, M. (2018). Model agnostic solution of CSPs via deep learning: A preliminary study. In *Proceedings of CPAIOR2018*, volume 10848 of *Lecture Notes in Computer Science*, pages 254–262. Springer. © 2018 Springer International Publishing AG, part of Springer Nature.

[77] Galassi, A., Lippi, M., and Torroni, P. (2018). Argumentative link prediction using residual networks and multi-objective learning. In *Proceedings of the 5th Workshop on Argument Mining @ EMNLP 2018*, pages 1–10. Association for Computational Linguistics. © 2018 Association for Computational Linguistics.

[79] Galassi, A., Kersting, K., Lippi, M., Shao, X., and Torroni, P. (2019). Neural-symbolic argumentation mining: An argument in favor of deep learning and reasoning. *Frontiers in Big Data*, 2:52. License CC BY 4.0.

[76] Galassi, A., Lippi, M., and Torroni, P. (2020). Attention in Natural Language Processing. Accepted in *IEEE Transaction on Neural Networks and Learning Systems*. IEEE. License CC BY 4.0.

[80] Galassi, A., Lippi, M., and Torroni, P. (2021). Multi-Task Attentive Residual Networks for Argument Mining. *Under review*. License CC BY 4.0.

Chapter 1

Introduction

In the last decade, Deep Networks (DNs) have become a standard tool for machine learning tasks, and Deep Learning (DL) has spread in every domain and task. Experimental studies indicate that such approaches seem to have the ability to create representations at different levels of abstraction in each of their layers, reaching high-level representations before the output [94]. This has been a crucial characteristic for tasks that must be performed on data with a low level of abstraction, such as the pixels of an image or sub-symbolic representation of words. Indeed, DNs have completely overtaken some research fields, replacing more traditional Artificial Intelligence (AI) techniques based on symbolic approaches, which were not capable of elaborating effectively such raw inputs [174].

But despite their impressive performance, DNs are still at the center of a debate, largely due to problems related to their interpretation. With the spread of AI in every aspect of everyday life, the public debate regarding how much an AI can be trusted has evolved, and the need to regulate their application and development has produced new requirements. For this reason, explainability has quickly become one of the most desired characteristics for modern AI systems, a requirement that clashes with the connectivist nature of neural networks [99]. Moreover, the fact that DNs are not naturally fit to perform formal reasoning limits their domain of application.

There are domains where the desired behavior of an AI is given by high-level concepts: rules and constraints involving multiple entities that are given to the networks as input, either simultaneously or sequentially. These tasks usually require reasoning and/or access to a knowledge base that contains the rules of the domain in a formal representation.

One such example can be seen in the domain of games, where an AI must attempt to reach an objective (usually, obtaining a victory), behaving however in a manner which does not violate the rules of the game. These rules may be the same for both players, be applied in every entity of the game, and hold for the whole game, as in the game Othello. But in more

complex cases, different rules may be applied to different elements of the game, as in Chess, or in different phases of the game, as in Nine Men's Morris. Typically, when DNs are used to play games, they are part of a more complex system that performs reasoning and prevents the network to take a decision that would go against the rules of the game.

Another domain where DNs have yet to see a large scale application, are Constraint Satisfaction Problems (CSPs). The creation of a solution for a CSPs usally requires knowing the specific constraints of the problem as well as the ability to reason about the links between different variables and their values. The task becomes even more complicated when there are soft constraints that make some solutions more desirable, especially in settings where they are not explicit.

Obviously, in real-world applications, it is fundamental to impose restrictions on the behavior of DNs, so that they do not violate rules or constraints. But investigating the ability of DNs to perform similar complex tasks without having access to formal knowledge about the domain might provide insights on their internal functioning and how they can be exploited in more complex systems.

This argument can be extended beyond the topic of rules and can be applied to knowledge in general. One domain where reasoning, knowledge representation, and numeric methods are linked is Argument Mining (AM) [27, 140, 159]. On one hand, computational argumentation [14, 212] is a well-known method of monotonic reasoning, that is used to represent knowledge. On the other hand, argumentation is a natural human activity, linked to the domain of Natural Language Processing, a domain where DNs have achieved remarkable results [206, 285]. AM is a branch of Natural Language Processing (NLP) that aims to find the argumentation present in speeches or written texts, through the identification and classification of argumentative entities and the relationships between them. The initial focus on the development of hand-crafted tailored features, built on specific models and datasets, is now leaving space for the use of modern neural architectures, already widely applied in NLP [285]. However, while such methods improve the generality of the approach and can result in better performance, they do not allow to enforce desired properties and to investigate the final behavior.

It is reasonable to believe that this research field would greatly benefit from a system capable of exploiting the ability of DNs to operate on low-level data, as well as a symbolic reasoning system capable of guiding, constraining, and validating the prediction of the networks. In the last years, the interest in neural-symbolic solutions has led to the development of many techniques and frameworks that aims to achieve this result. So far, the application to NLP has been scarce, and no attempt to use them for AM has been carried on yet.

1.1 Contribution

In the first part of this thesis, we explore the abilities of DNs in environments that are challenging due to the presence of complex constraints and rules. We train a system of DNs in a supervised fashion firstly to play a game and then to solve a CSP. In the first case, the networks are not provided with any information regarding the rules of the game, but they are simply trained to emulate the moves of an expert player. In the second experiment, the networks are trained to construct the solution of a CSP, emulating the construction of some specific solution. No information regarding the constraints of the problem or any other characteristic of the domain is provided. In both cases the networks are trained to emulate, but they are tested on whether they have learned to provide a prediction that does not violate the rules of the game or the constraints of the problem.

In the second part of the thesis, the task of Argument Mining is addressed. Firstly we approach the problem with a setting similar to the one used for the game and the CSP. The network is not provided with any information regarding the argumentative structure of the documents, and instead it is asked to simply classify two components and their relationship. We have additionally decided to avoid the use of most of the features that are usually exploited in this field. Instead, we have chosen to rely only on a sub-symbolic representation of the sentences and information regarding their distance. Finally, we face the problem of neural-symbolic argument mining, presenting examples of realization and conducting experimentation with the Logic Tensor Networks framework.

The domain of DNs is vast and in continuous evolution, with the constant development of new models and techniques. We have decided to focus on a specific type of DNs, Residual Networks (ResNets), since they have reached remarkable results in the field of NLP [19, 46], and they are designed to speed up the computation time, therefore they do require fewer computational resources compared to other architectural models.

The contribution of this thesis can be summarized as follows:

- We investigate the ability of Residual Networks to perform complex tasks without having access to contextual structured information. These investigations are carried on in three different domains characterized by the presence of rules and constraints: board games, CSPs, and Argument Mining.

- We propose to integrate DL and Reasoning techniques for the task of AM. We provide examples of possible frameworks and techniques that can be used and we offer experimental results obtained using the Logic Tensor Networks framework [59, 60, 228, 229].

1.2 Outline

Since the content of this thesis covers multiple topics, which range from game playing, solving constraint satisfaction problems, and Natural Language Processing, it would be confusing to offer a single "related works" section. Instead, we have decided to provide the proper background in every chapter, following the natural flow of the research.

Part I of this thesis presents the investigation of DNs. Chapter 2 and 3 report the results obtained respectively in the domain of board games and CSPs. Chapter 4 concludes this first part and paves the way for the next one.

In Part II we dive into the core of this research. Chapter 5 covers the domain of Argument Mining, providing information on the available resources and information regarding the current state-of-the-art. Chapter 6 provides background information regarding Neural Attention, a widely used technique that provides both an improvement in the performance of DNs and the possibility of getting insights regarding the behavior of the networks. Chapter 7 consist of experimentation on the use of attention-based feature-agnostic DNs on challenging datasets.

In Part III we face the challenge of neural-symbolic argument mining. Chaper 8 analyzes possible solutions for the integration of DL and reasoning techniques, while Chapter 9 present the first application of these techniques to this task.

Finally, Part IV concludes the thesis, summarizing our findings and outlining future works and research directions.

Part I

Deep Learning when Rules Matter

Chapter 2

Learning to Play Nine Men's Morris

Deep networks have been successfully applied to a wide range of tasks in artificial intelligence, and game playing is certainly not an exception. In this chapter, we present an experimental study to assess whether purely sub-symbolic systems, such as deep networks, are capable of learning to play by the rules, without any a-priori knowledge neither of the game, nor of its rules, but only by observing the matches played by another player. Similar problems arise in many other application domains, where the goal is to learn rules, policies, behaviours, or decisions, simply by the observation of the dynamics of a system. We present a case study conducted with residual networks on the popular board game of Nine Men's Morris, showing that this kind of sub-symbolic architecture is capable of correctly discriminating legal from illegal decisions, just from the observation of past matches of a single player. The content of this chapter is largely based on the work presented in Chesani et al. [36].

2.1 Introduction

Game playing has been the source of inspiration and the testbed for many advancements and discoveries in Artificial Intelligence (AI). In the last decade, neural networks and especially deep learning techniques [142] have brought a revolution within AI and games. The application of deep reinforcement learning techniques to the development of agents playing Atari video-games has been one of the most successful deep learning stories [188]. The design of AlphaGo [235, 236], a computer system capable of beating several Go world champions[1] has become a milestone in the history of AI.

The development of AI techniques for game playing has long known the traditional dichotomy between symbolic and sub-symbolic approaches [57]. Symbolic frameworks are

[1] https://deepmind.com/research/alphago/.

based on an explicit and formal representation of the domain (often in a human understandable way) like, for example, in the form of logic facts and rules. Background knowledge can be encoded in the system, and reasoning techniques can exploit it to derive additional information and take decisions: a characterizing aspect is that the reasoning is performed at the level of symbols. Sub-symbolic (also named connectionist) approaches consider instead reasoning as the result of the computation of a network of simple processing devices, without the need of explicitly representing knowledge through symbols. One major example of this class of models is clearly given by artificial neural networks. In game playing, both symbolic and sub-symbolic approaches have historically been widely applied.

A great research effort has been put on learning strategies for winning games: the majority of the approaches takes as granted the game rules, often expressed in some formal description language. Trying to learn game playing without any kind of external hint of which are the actual game rules, instead, has rarely been addressed in the literature of AI in games. Arthur Samuel, in his seminal paper in 1959 on the application of machine learning techniques to the game of checkers [223], stated:

> *"The rules of the activity must be definite and they should be known. Games satisfy this requirement. Unfortunately, many problems of economic importance do not. While in principle the determination of the rules can be a part of the learning process, this is a complication which might well be left until later."*

Even the most recent applications to game playing, such as AlphaGo, or the system developed by Clark and Storkey [39], although heavily relying on the computational power of deep networks, still encode either in the network architecture or in the input features some information about the rules.

In this work, we want to assess whether a deep learning approach can be exploited to learn to *play by the rules* a board game without any symbolic, background knowledge about the game rules. We consider as a case study Nine Men's Morris, a popular board game whose state space is not huge with respect to other games of the same kind. However, the complexity of the rules to be learned, and the large number of possible decisions to be taken by a player, make such a game a challenging benchmark.

Our main goal is not to be seen in terms of learning winning strategies, but rather in terms of learning legal moves. To this end, we have constructed a dataset of game states and possible legal moves, by observing the behavior of an AI player based on a symbolic approach. Such a dataset has been exploited to train a neural network system named *Neural Nine Men's Morris* (NNMM), based on residual networks [107]. For these reasons, we do not aim at comparing against any other system that exploits some (even partially) symbolic approach.

NNMM has been evaluated in terms of (percentage of) suggested legal moves, resulting in very good performance: in almost the totality of the cases the network suggests, as a first choice, a legal move. Moreover, we have also evaluated, through a quantitative analysis, the "level" of legality expressed by our network over the first N suggested decisions i.e., how much the networks are able to exhibit a compliant behavior with respect to the game rules. Again, results show that the presented approach is highly effective.

Natural extensions of this kind of analysis could be provided in other domains, such as decision making [122], anomaly detection [31], process mining [261]: when rules cannot be clearly identified or stated, a learning approach to compliance, based on observed behavior, could be an interesting alternative. Clearly, games represent the ideal scenario to first test our idea, as they provide frameworks where game rules are clearly defined and it is easy to check the legality of decisions.

The paper is structured as follows. In Section 2.2 we briefly revise the game rules, whereas in Section 2.3 we describe how we modeled the game, and which neural networks have been used in our experiments. In Section 2.4 we present the dataset built for our purposes, while Section 2.5 will report the experimental results. Section 2.6 will discuss related works, and finally Section 2.7 will conclude the paper.

2.2 Nine Men's Morris

Nine Men's Morris, also known as Mill Game, Merrils, or Cowboy Checkers, is a perfect-information strategy board game for two players. It is a very ancient game, dating back circa 1,400 B.C. [11], but still very popular in many countries around the world. This game has long been analyzed from the point of view of AI and game theory, and its solution has been proven to be a draw [84]. Recently, theoretical results have been found for ultra-strong and extended solutions [89].

There exist several game variants that differ for either the game board or for the rules. We hereby briefly describe the most common setting, that has been used in our experiments. The game board, depicted in Figure 2.1, consists in three concentric squares and four segments which connect the midpoints of the sides of the squares. The intersections of two or more lines create a grid of 24 points where checkers can be placed. Each player has nine checkers (also called stones or men). The game proceeds through three different phases, that define the allowed moves: (1) starting from an empty board, players alternately place a stone on an empty position; (2) when both players have placed all their stones, they must slide a checker along a line to a nearby empty position; (3) if a player remains with only three stones, then the

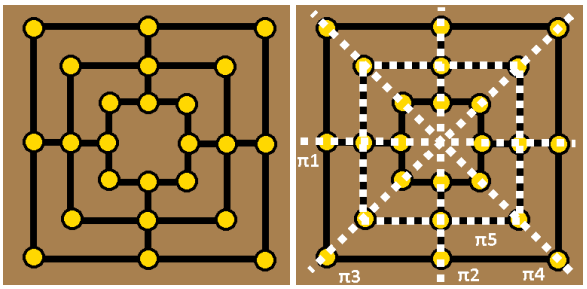


Figure 2.1: Nine Men's Morris game board (left) and its symmetry axes (right).

constraint to move to an adjacent position is removed: checkers can be moved to any empty position in the board (the checker can "fly" or "jump") .

When a player succeeds in aligning three checkers along a line ("closing a mill"), he/she removes an opponent's checker from the board among those checkers not belonging to any mill.[2] The removed checker is said to be "eaten" or "captured".

The game ends when one of these conditions occurs: (i) player A removes seven stones of player B, thus leaving B with less than three stones (A wins); (ii) player A cannot make any legal move (A loses); (iii) a configuration of the board is repeated (draw).[3] Whereas the first two ending conditions can be detected by observing the game state, recognizing the third condition requires to keep track of the game history.

If we take into account only the game states during phases (2) and (3), each of the 24 cells of the board can be either occupied by a white checker, or by a black checker, or it can be empty. Thus, an upper bound for the number of possible states is 3^{24}, that is approximately 2.8×10^{11}. However, there are further constraints that limit the number of possible states: for example, if a player has closed a mill, the opponent cannot have all the 9 checkers on the board. If board symmetries are considered too, it can be shown that the game has 7,673,759,269 possible states in phase (2) and phase (3) [84]. The number of possible game configurations is thus not dramatically huge: as a comparison, consider that the chess game allows between 10^{43} and 10^{50} different configurations [5].

The number of moves that a player can do is quite large. In the most general case, the player faces three decisions: the checker to move, where to place it and which adversary's stone to remove. As explained later in Section 2.3.1, these decisions can lead to a quite large number of alternative moves.

Summing up, the game enjoys the following characteristics:

[2] In the case that all the opponent's checkers are aligned in at least one mill, one aligned checker is allowed to be removed. If two mills are closed at the same time, still only one checker can be removed.

[3] Repetitions can happen only during phases (2) and (3).

1. Symbolic approaches have been proven to successfully solve and play it. Therefore is a well-known case of study and we can rely on background knowledge.

2. The dimension and complexity of the state space is not huge: as a consequence, the process of training a sub-symbolic approach does not require excessive resources in terms of time and hardware.

3. The choice of a move implies several decisions, and a legal move must satisfy constraints that affect both the single decisions and the move as a whole. As a consequence, the dimension of the space of legal moves is relatively small, when compared to the dimension of the space of possible moves, thus making the selection of legal moves a difficult and interesting problem (see Section 2.5.1 for details).

For these reasons, the Nine Men's Morris game represents a challenging case study for assessing whether a sub-symbolic system is capable of learning to play by the rules.

2.3 System Architecture

In this section we describe how we represent the game, and which architectures have been selected for the neural networks trained to play the game.

2.3.1 Game Modeling

The state of the game consists of four pieces of information: the board configuration and, for each player, the number of checkers he/she still has in his hand, the number of checkers that he/she has on the board and the phase of the game in which he/she is. The last two pieces of information can be deducted from the first two. The number of checkers in the hands of each player can obviously be represented with two numbers, each of which can assume values from 0 to 9. For the game board, several different representations could be chosen, by exploiting a one-dimensional array, a two-dimensional matrix, or a three-dimensional cube. For the sake of simplicity, we just used the most straightforward implementation, that is a plain enumeration of the board cells coded into a one-dimensional array (the i-th element of the array representing the i-th cell in the enumeration). Each cell can be occupied either by a white checker, or by a black checker, or it can empty. We can easily represent such configurations with three different values.

A move consists of three distinct pieces of information:

TO: In every game phase, a checker is always placed somewhere. This will be either a newly introduced one during phase 1, or a checker that is already present in the board during phases

2 and 3. We name this information "TO". It can assume 24 possible values (the cells of the board).

REMOVE: If placing the checker causes the closing of a mill, another information that must be encoded in the move is the adversarial checker to be removed. We name this information "REMOVE" and it can assume 25 possible values: the 24 cells of the board plus the none value, in the case that the move implies no removal. During a single match, the maximum number of moves which imply a removal is 13, that is 7 by the winner and 6 by the loser.

FROM: During phases 2 and 3, the checker is moved from one position to another, so we have to encode also the initial position. We name this information "FROM". Therefore, it can assume 25 possible values: the 24 cells of the board plus the none value, in the event that the game is in phase 1.

Without knowing the constraints on these three information, i.e. without any knowledge on the game's rules, the number of possible combination, and therefore of possible moves, is $24 \times 25 \times 25 = 15,000$. This number is quite big compared to other boardgames: for example, in the game of Go a single decision has to be taken ($19 \times 19 = 361$ positions on the board), while in the game of chess two decisions have to be taken[4] (initial and final positions of the moving piece – $64 \times 64 = 4,096$ possible couples of positions).

2.3.2 Neural Nine Men's Morris (NNMM)

Our system thus consists of three different neural networks, each predicting one part of the move (TO, REMOVE, FROM). We model the problem as a collection of three supervised learning tasks, where the target of each task is the partial decision to be taken by the player at a given board configuration. The output size of the TO network is 24 neurons, which represent the possible positions on the board. The REMOVE and FROM networks, instead, will also have an additional special output neuron (thus 25 output neurons) encoding the case in which no checker has to be moved or eaten, respectively. For this reason, we also let the TO network have 25 output neurons (with one extra neuron that is never used) so that the three networks have identical architectures. Any network will then provide a single position value among 25 possible ones. Note that exploiting a single neural network would have produced a number of output classes (all the possible moves) equal to 15,000 (see Section 2.3.1), which would have made the training almost unfeasible.

Features have been represented with a binary encoding, thus exploiting different bits to represent different possible values of the game state variables. The feature vector thus composed is described in Table 2.1. It simply consists of the board configuration and the number of

[4]The *castling* move can be indicated with the coordinates of the *king* involved. We are not taking into account the decision involved into *promotion*.

Table 2.1: Input features for the three networks in NNMM.

Feature	Bits	Description
Player's board	24	Position of player's checker on the board
Adversary's board	24	Position of adversary's checker on the board
Empty board	24	Empty position
Player's hand	9	The number of checkers in player's hand
Adversary's hand	9	The number of checkers in adversary's hand
First network choice[a]	25	Decision taken by the first network of the system
Second network choice[b]	25	Decision taken by the second network of the system

[a] Present only in the second and third networks
[b] Present only in the third network

checkers in hand for each player. In fact, the three networks operate in cascade, providing the positions chosen by the former ones as input to the latter ones. Since the output of each network is independent from the decision it has to make, its input/output structure of each network only depends on its position in the architecture. Since the decisions to be taken by the networks are clearly strongly correlated, exploiting independent networks rather than a cascade model, thus taking the three decisions (TO, FROM, REMOVE) independently one from the other, would have certainly lead to worse performance. Figure 2.3 illustrates the overall architecture of the system, which we name Neural Nine Men's Morris (NNMM), when the cascade of the three networks is in the order TO-FROM-REMOVE (TFR). In Section 3.3 we investigate also different arrangements of the networks, and evaluate through experiments their performance. We hereby underline the fact that, at this point, no choice has been made on the internal neural network used for each step.

Finally, we remark that, for any game state, there are legality constraints both on the choices of each of the three networks, but also on the whole move. That is, the legality of the parts does not guarantee the legality of the full move.

Although the chosen network architecture may look specifically tailored for Nine Men's Morris, its cascade structure (where the decision on the n-th part of the move depends on the first $n-1$) easily allows to generalize it to other board games. More precisely, it could be immediately applied to any board game where one has to choose a piece to move (FROM), a position where to place it (TO) and possibly also an opponent's piece to remove (REMOVE).

2.3.3 Residual Networks

A preliminary experimental study was conducted to choose the architecture for this case of study and to tune its parameters. As a result, we decided to adopt residual networks, which had achieved the best performance. We hereby describe the final system that will be adopted in the experimental evaluation.

Residual networks [107, 108] are a family of deep neural networks that achieved outstanding results in many machine learning tasks, in particular in computer vision applications such as medical imaging [287], but also natural language processing [19, 54, 263], crowd flow prediction [292], and game playing [28].

Residual networks have been designed to address the problem of vanishing or exploding gradient that affects most of deep neural architectures. Indeed, the use of non-linear functions combined with the large numbers of layers may make the gradient excessively small or large, resulting in a training process that is, respectively, ineffective or unstable.

The core idea behind residual networks, illustrated by Figure 2.2, is to create shortcuts that link neurons belonging to distant layers, whereas standard feed-forward networks typically link neurons belonging to subsequent layers only. This kind of architecture usually results in a speedier training phase, and it usually allows to train networks with a very large number of layers. The original architecture exploits convolutional layers, but it can be generalized so as to make use of different layers, such as dense (fully-connected) layers. The motivation behind residual networks is that if multiple non-linear layers can asymptotically approximate a complex function $H(x)$, they can also asymptotically approximate its residual function $F(x) = H(x) - x$. The original function is therefore obtained by simply adding back the residual value: $H(x) = F(x) + x$. Further improvements, such as the use of pre-activation of weight layers and of dropout inside the residual unit [108, 245], allow to achieve even better results than the original model on challenging computer vision tasks. The optimization problem results to be easier to solve with respect to traditional (non-residual) deep networks, in particular when the number of layers increases, allowing the gradient to "flow" through the connections, which typically lets these networks achieve better performance. A similar principle is followed also in the design of dense and highway networks [119, 299].

All the three networks in NNMM are residual networks, and each of them presents several blocks as that depicted in Figure 2.4. Each block follows a fully-connected layer and is made by two sub-blocks, each consisting of a rectifier linear unit [90] pre-activation layer, a dropout layer, and a fully-connected layer. Each network has an initial layer of 200 neurons, followed by a set of residual units stacked one onto the other. In each residual unit, the first layer contains 300 neurons, while the second 200. TO and FROM networks have been made by 10 residual units while REMOVE networks have been made by 30 units. Each network terminates with an

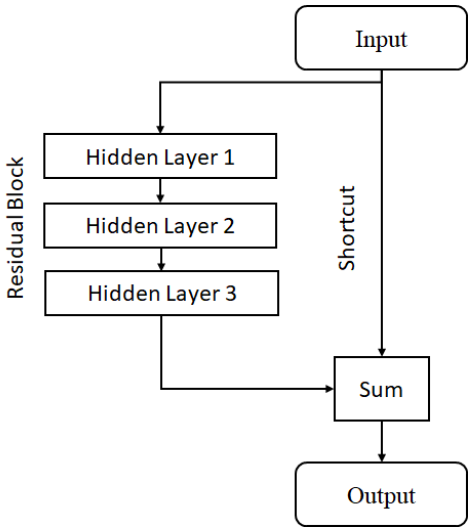


Figure 2.2: General schema of a residual network with a single residual block with three hidden layers.

output layer made of 25 neurons, to which the softmax function is applied, so that we obtain a probability distribution over the possible positions in the board. The total depth is thus 22 layers for the TO and FROM networks, and 62 for the REMOVE one.

2.4 Datasets

To evaluate the capability of deep networks to learn legal moves in Nine Men's Morris, we built a dataset of game matches to train our system. To this aim, we exploited a collection of AI players, all based on symbolic approaches. Such systems were developed by students for a competition within the "Foundations of Artificial Intelligence" course at the University of Bologna. The winner of the competition was chosen as the "trainer" of our deep networks. Thus, for each game board, the move of the trainer is used as the supervision target. This implies that the target moves in the dataset are not *optimal* moves according to some criterion, but rather *good* moves according to the trainer and, most importantly, they are *legal* moves. The decision to use this dataset, rather than sampling from a database in which the optimal moves are indicated, such as the one presented by Gévay and Danner [89], came from the intention to verify if the sub-symbolic system could learn to play by the rules, by simply observing matches played by another player.

For each state, the symmetries of the problem shown in Figure 2.1 have been exploited to increase the number of examples. This "Matches Dataset" is composed by 1,628,673 state-move pairs. Each state in the dataset is unique, therefore for any state only one move is present. To generate the dataset, the symbolic trainer has played 7,244 games against itself and the

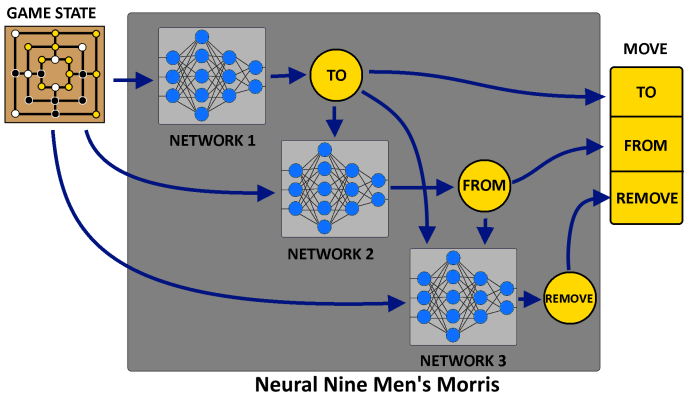


Figure 2.3: An illustration of our NNMM system, exploiting the TO-FROM-REMOVE configuration. Each of the three networks takes as input both the board and the decision taken from the previous network(s) in the cascade.

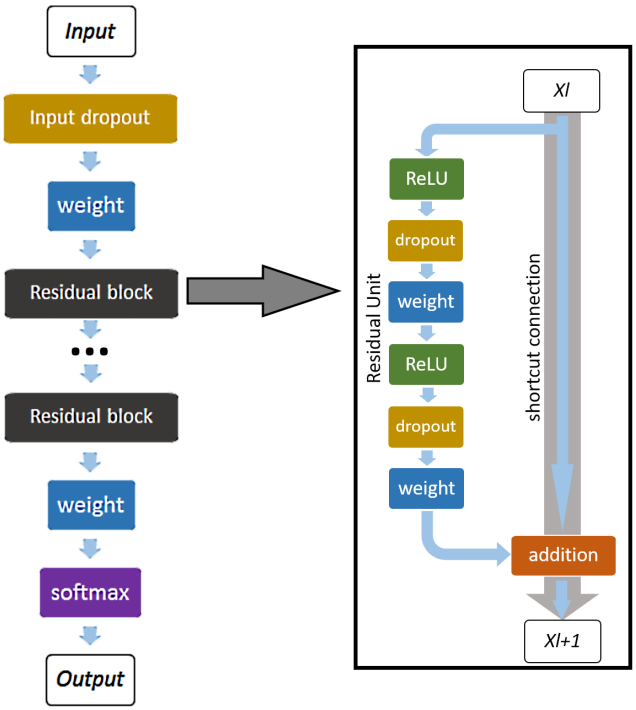


Figure 2.4: Illustration of the residual network used in our experiments.

other AI players. We let some of the games start from initial random configurations, rather than from the conventional one (i.e., empty board and nine checkers in each player's hand). Some of these random states, and therefore some of the states obtained during that match, are not reachable[5] in a proper game that starts from the conventional initial state. This feature of the dataset makes it more general for the task of learning game rules, as it limits the possible problem of overfitting on a subset of the whole state space. At the same time, the generation of the initial board state takes into account some characteristics of the game to ensure that the rules of the game, and therefore that legality definition, still hold. Some of the constraints imposed on the randomly generated configuration are: the next player to move is the white one, each player must have at most 9 and at least 3 checkers (among the hand and the board), both players must hold the same number of stones in their hands. With these constraints, the state of the game will always belong to one of the three phases described in Section 2.2. Moreover, the symbolic players used to generate the dataset have knowledge of the game rules, therefore they are capable of making only legal decisions.

As an additional test set, a second dataset was built. It contains random game states, that have been sampled as reachable board configurations, without any overlap with the ones present in the matches dataset. In this way, 2,085,613 states have been gathered. We name this second dataset "States Dataset". Such dataset has been used only as a further test set for our system, with the goal of evaluating the generalization capabilities of the system on generic states that have not been reached by a match played by the NNMM's symbolic teacher.

Both datasets are available online[6]. Each data point is represented as a textual string of 31, 33, or 35 characters. The first 24 characters describe the board state with a letter representing the state of each position: empty (O), occupied by a checker of the player (M), or occupied by an opponent one (E). The positions are represented in order as they appear from left to right and from top to bottom. A sequence of 4 numbers completes the state representation, where the first two numbers represent the number of checkers in the hand of the player and of their adversary, and the last two represent the number of checkers that the players and the adversary have on the board. A hyphen divides the game state from the move description, which is written as pairs of letter-number coordinates; the meaning of each coordinate depends on the game phase: the parts of the move are written (if present) in the order FROM, TO and REMOVE.

[5]For a reachable state we mean a state that can be reached, from the initial board configuration, with a sequence of legal moves only.

[6]https://github.com/AGalassi/NNMM/tree/master/datasets

2.5 Experiments

In this section we present our experimental results. After an analysis of the datasets, aimed to investigate the space of legal moves, we present three different system configurations. Then, we describe the tasks on which such systems have been tested, and we discuss the achieved results. The source code for the replication of these experiments is available online, together with the trained neural network models.[7]

2.5.1 Datasets Analysis

In order to prove that the space of legal moves is very small in comparison with the space of possible moves, the two datasets have been analyzed, in particular measuring (1) the number of legal moves for each state, and (2) the number of states in which a move is legal.

For both datasets the highest number of legal moves allowed by a state is 58, while the lowest is 1. The mean number of legal moves per state is 18 for the Matches Dataset and 22 for the States Dataset. Since our representation allows 15,000 different moves ($24 \times 25 \times 25$), the space of legal moves is at most the 0.39% of the space of possible moves, and on average it is less than 0.15%.

Moreover, for each of the 15,000 moves, we counted the number of states in each dataset where such move is considered legal. On average, a move is legal in the 0.12% of the total number of states in the Matches dataset, and in the 0.15% of the States dataset. The number of moves which are always illegal in both datasets is 1,920, while only 32 moves are legal in more than 10% of both datasets. There is no move which is legal in more than 13% of either the Matches or the States datasets. These 32 moves that are more frequently legal are all characterized by a REMOVE value of 0 (thus they do not remove any checker) and FROM values that represent those positions of the board connected with most other positions. No special pattern is observed in the TO values.

This analysis suggests that the problem of learning to play by the rules in Nine Men's Morris, without any background knowledge of the problem, is particularly challenging.

2.5.2 Setup

The Matches Dataset has been used as the development dataset. It was partitioned into a training set, a validation set to monitor learning and to perform parameter tuning and early stopping, and a test set to evaluate the model at the end of the training phase. The test set consisted of 10% of the whole dataset (about 163,000 pairs), while the validation set consisted in 5% of the

[7] https://github.com/AGalassi/NNMM

dataset (about 81,000 pairs), leaving about 1.4 million pairs for the training set. Each network was trained independently, using the game state and, eventually, a partial part of the move in the dataset as inputs.[8] The desired partial move played by the symbolic trainer was used as target.

The loss function chosen as the objective of training was the negative log-likelihood of the target class, with an L1 regularization with weight 10^{-3}. Adam [130] was used as optimizer, with parameters $b_1 = 0.99$ and $b_2 = 0.999$. The initial learning rate $\alpha_0 = 2 \times 10^{-3}$, was progressively annealed through epochs with decay proportional to training epoch t, resulting in a learning rate $\alpha = \frac{\alpha_0}{1+k\times t}$, with $k = 0.01$ for TO and FROM networks, and $k = 0.02$ for REMOVE networks. Parameters were initialized with He initialization [106], specifically designed for ReLU activation. We employed mini-batch optimization, with batch size equal to 20,000. For TO and FROM networks, dropout was applied to each layer, with $p = 0.1$, while it was not used for the REMOVE network. For early stopped, we used a patience value of 50 epochs. The whole system was implemented with the Lasagne [55] and Theano [255] frameworks.

Three different NNMM system configurations were compared, designed with different orders in deciding move parts. We name them after the order in which the decisions are made. The configurations are: TO-FROM-REMOVE (TFR), FROM-TO-REMOVE (FTR) and REMOVE-FROM-TO (RFT). The first configuration (TFR) is the one which appears to be more logical for a human player: in each phase of the game the system has to place a checker somewhere – so that it is the first decision that has to be taken – then it decides where that checker has to come from (to know if a mill has been closed) and if an opponent's checker should be removed. The second one (FTR) is an alternative to the first one, where the checker to be moved is considered the most important decision. The last one (RFT) can be considered as an extreme case study, as it seems illogical for a player to decide which opponent's checker has to be removed before even deciding which of his/her own checker should be moved and where.

2.5.3 Tasks

The system was tested to evaluate three different aspects: (1) accuracy, that is the capability of reproducing the same complete move of its teacher; (2) legality, that is the capability to suggest, as the best complete move, one that respects all the game rules; (3) reliability, that is its capability to give legal decisions a higher probability than non-legal decisions – thus, considering not only the top-ranked decision, but also the subsequent ordering.

[8] Because the second and the third networks in each configuration need a partial move as input.

The accuracy test simply measures how many times the move suggested by the sub-symbolic system is equal to that provided by the symbolic trainer. Such a test does not give any hint about the quality of the system, because it does not evaluate whether the outputs of the sub-symbolic system are better or worse than the choices of the symbolic system. A high accuracy, in this sense, is not necessary for our purpose of learning legal moves. Yet, it is a useful metric to assess that the training phase reached a reasonable network configuration.

The legality test is the most important one for our goals, since it measures whether the system has been able to learn to play by the rules, by counting how many times the move suggested by the sub-symbolic system violates any of the game rules. It has been performed on both datasets.

Finally, the reliability test moves the legality test a step further. It is designed with the goal of assessing whether the system is able to correctly discriminate between legal and illegal decisions, and thus if it has learned a sort of *correct behaviour*. To this aim, we separately consider the ranking of the output – partial or complete – moves for each network, according to their probability. For a partial move, we hereby mean the outcome either of the first network, or of the first and the second networks together. If the system has correctly learned the concept of game rules, then all the legal decisions should ideally appear before the illegal ones in such ranking. It has been performed both on the good moves and the testing datasets. To give a quantitative measure of this property, we build a sort of recall-precision curve as follows. For each board configuration in the test set, we consider the output of each network in the pipeline, ranking decisions by their probabilities, and we compute the percentage of legal moves (precision) as the number of retrieved legal moves (recall) increases. Finally, we average over all the states in the test set for each of the three networks separately. Clearly, for the first two networks in the pipeline the considered decisions will be partial moves, while for the last network these will be complete moves: each network in fact has to know the prediction of the previous network(s) in the pipeline. In the ideal case, when all legal decisions precede the illegal decisions, all the precision values are equal to 100%. Otherwise, if some illegal move is ranked higher than some legal move, the precision will decrease accordingly. Table 2.2 shows an example of how such evaluation metric is computed. For the last network the recall is thus the number of legal complete moves retrieved upon the number of existing legal complete move, whereas its precision is the number of legal retrieved complete moves upon the total number of retrieved complete moves. The same principle is applied to the second and first networks, considering only the legality of the partial moves. While the definition of legal complete move is straightforward, this is not the case for legal partial moves, which could be subject to different interpretations. For the purpose of our tests, we have listed all the possible legal complete moves for each state and decomposed them into partial ones, so as to define

Table 2.2: An example of computation of the recall-precision curve. For a given board configuration, suppose there exist only five legal decisions, namely outputs 1,7,9,5,12. For each value of N from 1 up to 5, we thus consider the decision ranking that is necessary to retrieve N legal decisions: in this way, for each value of N we compute recall and precision. Bold numbers represent illegal decisions.

N	Decision Ranking	Recall	Precision
1	1	0.20 (1/5)	1.00 (1/1)
2	1,7	0.40 (2/5)	1.00 (2/2)
3	1,7,9	0.60 (3/5)	1.00 (3/3)
4	1,7,9,**0**,**4**,5	0.80 (4/5)	0.67 (4/6)
5	1,7,9,**0**,**4**,5,12	1.00 (5/5)	0.71 (5/7)

all the legal partial moves. In the case that one network takes an illegal decision, this makes every possible decision of the following networks illegal too. Therefore, those cases where there is no legal decision available to a network are discarded during the reliability evaluation of that network. Finally, note that for the FROM and REMOVE networks, we discarded the configurations for which no checker is moved/removed, respectively.[9]

2.5.4 Results and Discussion

Table 2.3 reports the accuracy on the Matches Dataset of each trained network in each configuration (TFR-FTR-RFT), thus considering the three partial moves separately. Results show that similar values of accuracy have been achieved on training, validation and test sets, thus suggesting that the networks have maintained a good generalization, and that the phenomenon of overfitting does not have a strong impact here. Figure 2.5 shows an example of learning curve, for the TO network within the TFR configuration.

In Table 2.4, we instead report the accuracy of the three configurations over the complete moves, highlighting the differences with respect to each phase of the game. It is interesting to note that the values of accuracy are very similar for the three network configurations. The game phase where the system achieves the best accuracy is phase 1.

The most impressive results have been obtained for the legality test. As depicted in Table 2.5, the system demonstrates to have learnt to respect all the rules of the game in more than 99% of the cases, independently from the configuration of the networks and from the dataset. To better investigate which are the rules that are more frequently broken by our system, the legality of partial moves has been tested too. For the TFR and FTR configurations, the mistakes mostly regard the constraints on the REMOVE part. The RFT configuration obtains a slightly higher

[9]Trivially, those cases do not add information with respect to the legality test, since there is only one legal move.

Table 2.3: Accuracy of each network on training, validation and test sets for each of the three considered configurations.

Configuration	Move part	Training	Validation	Test
TFR	TO	53.07%	51.77%	51.73%
	FROM	90.07%	89.43%	88.97%
	REMOVE	88.01%	86.54%	85.66%
FTR	TO	75.73%	73.72%	74.17%
	FROM	65.00%	63.90%	63.33%
	REMOVE	88.54%	86.73%	85.88%
RFT	TO	78.20%	76.40%	76.75%
	FROM	68.92%	68.38%	67.94%
	REMOVE	81.27%	80.25%	79.45%

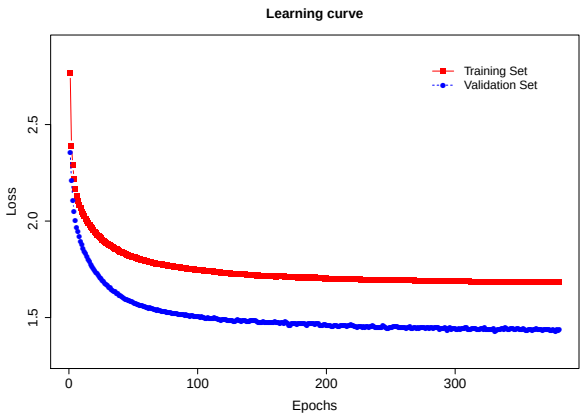


Figure 2.5: Learning curve of the TO network in the TFR configuration.

Table 2.4: NNMM accuracy test result.

Configuration	All phases	Phase 1	Phase 2	Phase 3
TFR	37.20%	47.91%	36.27%	29.19%
FTR	38.13%	49.28%	37.75%	27.58%
RFT	37.52%	48.70%	37.31%	26.33%

Table 2.5: NNMM legality test results.

Config.	Dataset	Whole move	TO	FROM	REMOVE
TFR	Matches	99.53%	99.94%	99.96%	99.62%
	States	99.53%	99.93%	99.96%	99.71%
FTR	Matches	99.53%	99.98%	100.00%	99.63%
	States	99.56%	99.98%	100.00%	99.73%
RFT	Matches	99.25%	99.91%	100.00%	99.86%
	States	99.19%	99.89%	100.00%	99.85%

legality percentage for the REMOVE network, which suggests that in that case the constraints which are more often broken regard the whole move.

The reliability test has confirmed that the system is able to differentiate between legal and illegal decision, holding very high average precision values for all the values of recall. As illustrated in Figure 2.6, the results on the two datasets are similar, which confirms that the system has learnt to generalize well on previously unseen data. The FTR configuration is the best setting: all the networks maintain a precision of about 99% for all recall percentages (note that the y-axis in the plots in Figure 2.6 starts at 0.9). The TFR configuration performs well for the second and third network, while the first one slightly loses accuracy as the recall increases. The RFT configuration not surprisingly results to be the worst (as it is difficult to first decide a checker to remove, before deciding which checker to move, and where), but still mantaining a 92% precision at 100% of recall on the States dataset.

2.6 Related Work

Gasser [84] solve the game of Nine Men's Morris exploiting brute-force approaches, demonstrating that its solution is a draw. The study of the game has been pushed forward by Gévay and Danner [89], who have found the "extended strong solution" and the "ultrastrong solution".

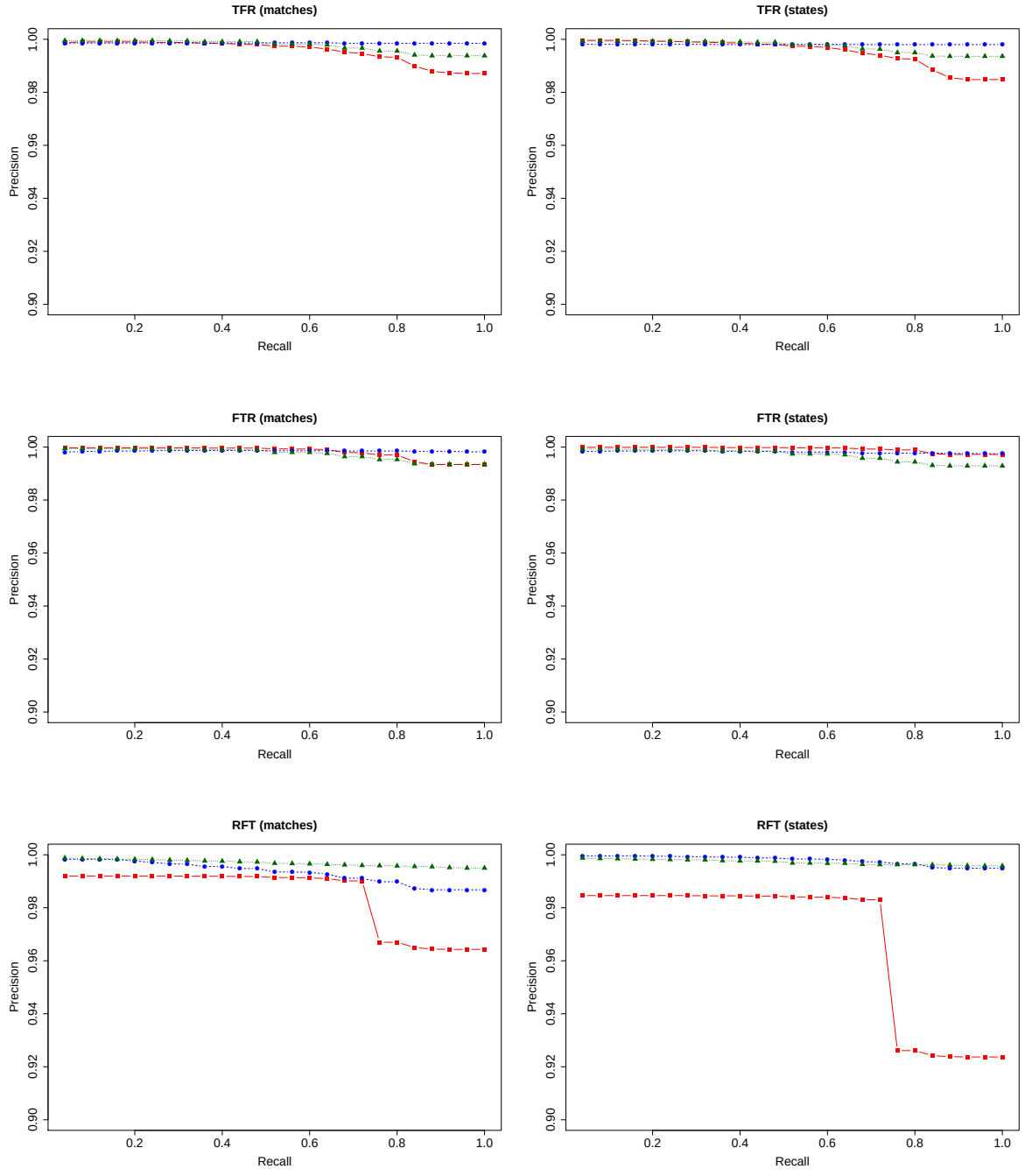


Figure 2.6: Reliability of NNMM in each configuration (top to bottom: TFR, FTR, RFT), reporting precision of retrieved legal moves as a function of the recall of retrieved legal moves. Left/right charts refer to the Matches/States Datasets, respectively. Note that y-axis starts at 0.9.

The former is the computation of the game-theoretic values of all the game states that could be reached in a match if the players have less than 9 checkers to place. The latter is the definition of a strategy that, against a fallible opponent, increases the chances to achieve a result which is better than the theoretic one. Even though these studies may have paved the way to our work, their purpose was very different from ours: their objective was to find optimal solutions to the game, whereas our focus has been on creating a system able to learn the game rules.

Game playing is a whole research field in AI, and a review of the many approaches available in the literature is out of the scope of this paper. The interested reader can refer to Yannakakis and Togelius [283] for a panorama of the main AI techniques. In such a context, our paper could be classified as an instance of behavioural learning of Non-Playing-Characters. Usually, reinforcement learning techniques or evolutionary computation are used to this end. This thematic is addressed by Muñoz-Avila et al. [192].

A main characteristic of our approach is that only sub-symbolic techniques are exploited. Systems which used combinations of sub-symbolic knowledge (acquired through learning) and symbolic knowledge (encoded into the system itself) have been proven extremely successful in playing many different games. Backgammon, Chess, Checkers and Go are only few notable examples of the games which have been addressed with combination of artificial neural networks and symbolic techniques, resulting in artificial players capable of playing a specific game achieving very good results. Among the many, we can cite the following works: Chellapilla and Fogel [33], David et al. [49], Lai [135], Silver et al. [235], Tesauro [254]. Yet, such approaches do not use neural networks directly to decide the move to be played, but rather to rank a list of moves that are typically generated by some symbolic approach (and, thus, which are certainly legal).

Recent works that instead employ deep networks for game playing, such as AlphaGo Fan/Lee [235], AlphaGo Zero [236], or the system developed by Clark and Storkey [39], consider only legal moves, by forcing to zero all the illegal options in the last network weight layer before softmax or by excluding illegal moves during a symbolic exploration phase. Among the many features pre-processed and given as input to AlphaGo Fan/Lee's neural networks, there are the characteristics of the status of each intersection of the Go board: liberties, legality, stone color, number of opponent and own stones that would be captured, and whether the move is a part of a ladder escape or capture. AlphaGo Zero, instead, uses only the last board configurations and players' choices as features. For the former, the game symmetries are handled by giving as input to the networks a mini-batch of all the symmetric states, so that they can be computed in parallel, while the latter is trained with a dataset which is augmented considering the symmetries, in a similar fashion to our work.

Another famous application of deep networks to game playing was given in Mnih et al. [188], where a reinforcement learning approach was undertaken to train a system to play Atari videogames, using only information of raw pixels in input, thus having no a priori knowledge of the game. Such a work, yet, does not have to deal with the concept of legal move.

An orthogonal approach with respect to our work has been investigated in the General Game Playing (GGP) competition [86]. Instead of creating a player specialized in a single game, the GGP problem consists in creating a system able to play any kind of game, given its rules. Within this context, rules are thus explicit knowledge that systems receive as input, represented in an appropriate language called Game Description Language. The competition is held once a year, since 2005, and it provides a benchmark for general approaches to AI and games. Even if many participants to this challenge rely only on symbolic techniques, some notable systems which combine symbolic and sub-symbolic techniques have been among the participants. For example, we cite the work of Reisinger et al. [217], where neuro-evolution is exploited to learn a game strategy. Although GGP considers the game rules as a known input, and asks participants to learn game strategies, it can be considered a very interesting point to further investigate the generality of our approach. For example, in the context of GGP, Björnsson [20] present a symbolic algorithm capable of learning the rules of a *simplified boardgame* from a dataset of matches. The dataset used for the training phase is made by game states and a non-exhaustive list of legal moves. Despite this similarity, Nine Men's Morris does not meet the definition of "simplified boardgame", thus making it unfeasible to apply such a solution to our case study.

In this work we used as training set a collection of (only) legal moves: in other words, we provided to the neural network only "positive" examples. Other approaches instead require both positive and negative examples, that is also a collection of illegal decisions. For example, in [191], variant chess rules are learned as extended logic programming theories from both positive and negative examples, background knowledge and by applying theory revision. Although being a very interesting approach, the need for both types of examples might be not feasible in a number of domains, were only observations of correct system dynamics are available. This is a common situation in fields like, for example, process mining, anomaly detection, human behaviour simulation and profiling.

Finally, it is relevant to underline that many other models of artificial neural networks exist, and thus different system architectures could be employed, possibly leading to better results. Stochastic depth networks [118] randomly drop layers during training, allowing to greatly increase the depth of the networks. Since we modeled the move as a sequence of decisions, Recurrent Neural Networks [276] could also be a useful alternative architecture, as they are usually applied for the classification of data sequences (e.g., in speech recognition tasks). Dense

networks [119] exploit the same intuition of residual networks, creating shortcuts between layers at different depth, and concatenating the outputs instead of summing them. A deeper investigation of different neural network architectures and training techniques applied to our context could be the subject for future works.

2.7 Discussion

Deep learning methods are widely employed in game playing. In the work presented so far the aim was to analyze whether such sub-symbolic systems are capable of learning to play a game by the rules just by observing a single player matches, without the need to explicitly model or encode any background knowledge of the game within the architecture of the network, nor providing any information about legality during the supervised training. Our analysis exploits residual networks, a particular type of deep networks specifically designed to learn models with many layers. Experimental results show that such systems are capable not only to suggest legal decision as best choices, but also of preferring legal decisions to illegal ones. Clearly, the chosen architecture and move encoding strongly affect the percentage of both possible and legal moves. Yet, it is worth remarking that, in the general case, it is not always possible to define an encoding that discards *a priori* illegal moves: in many board games, in fact, such as Nine Men's Morris but also chess or checkers, the legality of the move depends on the game status. In addition, looking forward beyond games, there are many applicative scenarios in the context of behavior compliance where it is just not possible to define in advance the concept of legality, and thus it certainly cannot be encoded within the move modeling. The proposed architecture is general enough to be employed with any board game where checkers are moved from a position to another, and opponent checkers are removed. Checkers and chess are other examples of such games. Thus, the impact of this kind of result goes beyond the application to game playing, opening the doors to the application of deep networks in many contexts where behavioural rules and decision policies could be learned directly from data, such as anomaly detection tasks.

Chapter 3

Learning to Solve Constraint Satisfaction Problems

The encouraging results obtained in the previous Chapter make us wonder whether it is possible to extend our method to other domains. We can interpret the problem of choosing a legal move in a board game as a combinatorial problem, where the rules of the game act as constraint. We have therefore decided to apply our method to the domain of Constraint Satisfaction Problems (CSPs). In this Chapter, we probe whether a DNN can learn how to construct solutions of a CSP, without any explicit symbolic information about the problem constraints. We train a DNN to extend a feasible solution by making a single, globally consistent, variable assignment. The training is done over intermediate steps of the construction of feasible solutions. From a scientific standpoint, we are interested in establishing whether a DNN can learn the structure of a combinatorial problem, even when trained on (arbitrarily chosen) construction sequences of feasible solutions. In practice, the network could also be used to guide a search process, e.g. to take into account (soft) constraints that are implicit in past solutions or hard to capture in a traditional declarative model. This research line is still at an early stage, and a number of complex issues remain open. Nevertheless, we already have intriguing results on the classical Partial Latin Square and N-Queen completion problems. The content of this Chapter is largely based on the work presented in Galassi et al. [78].

3.1 Introduction

Deep Neural Networks (DNNs) [142], are characterized by the ability to learn high-level concepts without the need of symbolic features. Crucially, previous use of Neural Networks to

solve CSPs rely on full knowledge of the problem constraints to craft both the structure and the weights of the networks. What we are trying to do is in fact radically different.

In this thesis, we investigate the idea that DNNs could be capable of learning how to solve combinatorial problems, with no explicit information about the problem constraints. This is partially motivated by the results achieved in a previous work regarding the application of DNNs to a board game [36]. In particular, we train a DNN to extend a feasible partial solution by making a single, globally consistent, variable assignment.

In principle, such a network could be used to guide a search process: this may be used to take into account constraints that are either implicit in the training solutions, or too difficult to capture in a declarative model. In this sense, the approach is complementary to Empirical Model Learning (EML) [163], where the goal is instead to learn a constraint. The method presented here is applicable even when only positive examples (i.e., feasible solutions) are available. Moreover, using the DNN to guide search may also provide a speed-up when solving multiple instances of the same problem. Practical applications are not our only driver, however: there is a strong scientific interest in assessing to what extent a sub-symbolic technique, trained on arbitrarily chosen solution construction sequences, can learn something of the problem structure.

This line of research is at an early stage, and there are many complex issues to be solved before reaching practical viability. So far, we have focused on two classical *Constraint Satisfaction Problems* (CSPs), namely N-queen completion and Partial Latin Square. For these benchmarks we have intriguing results, the most striking being an impressive discrepancy between the (low) DNN accuracy and its (very high) ability to generate feasible assignments: this suggest that the network is indeed learning something about the problem structure, even if it has been trained to "mimic" specific solution construction sequences.

This is not the first time that Neural Networks have been employed to solve CSPs. For example, Guarded Discrete Stochastic networks [2] can solve generic CSPs in an unsupervised way. They rely on a Hopfield network to find consistent variable assignment, and on a "guard" network to force the assignment of all the variables. The GENET [270] method can construct neural networks capable of solving binary CSPs, and was later extended in EGENET [143] to support non-binary CSPs. In [23], a CSP is first reformulated as a quadratic optimization problem. Then, a heuristic is used to guide the evolution of a Hopfield network from an initial state representing an infeasible solution to a final feasible state.

3.2 General Method and Grounding

3.2.1 General Approach.

We train a DNN to extend a partial solution of a combinatorial problem, by making a single additional assignment that is *globally consistent*, i.e. that can be extended to a full solution.

We use simple bit vectors for both the network input and output. We represent assignments using a one-hot encoding, i.e. for a variable with n values we reserve n bits; raising the i-th bit corresponds to assigning the i-th domain value. If no bit is raised, the variable is unassigned. Using such a simple format makes our input encoding *general* (any set of finite domain variables can be encoded), and truly *agnostic* to the problem constraints. As a major drawback, the method is currently restricted to problems of a pre-determined size.

Our training examples are obtained by deconstructing a comparatively small set of solutions. We considered two different strategies, referred to as *random* and *systematic deconstruction*, as described in Algorithm 1 and 2. Both methods operate by processing a partial solution s and populate a dataset T with pairs of partial solutions and assignments. In the pseudo code, s_i refers to the value of the i-th variable in s, and $s_i = \bot$ if the variable is unassigned. The random strategy generates in a backward fashion one arbitrary construction sequence for the solution. The systematic strategy generates all possible construction sequences. When all the original solutions have been deconstructed, we prune the dataset by considering all groups of examples sharing the same partial solution, and selecting a single representative at random.

Alg. 1 RandomDeconstruction(s)

Randomly choose a variable index i
$s' = s$ (copy the partial solution)
$s'_i = \bot$ (undo one assignment)
Insert (s', s_i) in T
RandomDeconstruction(s')

Alg. 2 SystematicDeconstruction(s)

for all variable indices i **do**
  $s' = s$ (copy the partial solution)
  $s'_i = \bot$ (undo one assignment)
  Insert (s', s_i) in T
  SystematicDeconstruction(s')

The DNN is trained for a classification task: for each example, the target vector (i.e. the class label) contains a single raised bit, corresponding to the assignment s_i in the dataset. The network yields a normalized score for each bit in the output vector, which can be interpreted as a probability distribution. The bit with the highest score corresponds to the suggested next assignment. We take no special care to prevent the network from trying to re-assign an already assigned variable. These choices have three important consequences: 1) the network is agnostic to the problem structure; 2) the network is technically trained to mimic specific construction sequences of arbitrarily chosen solutions; 3) assuming that the DNN is used to guide a search process, it is easy to take into account propagation by disregarding the scores for variable-value

pairs that have been pruned. As an adverse effect, we are forsaking possible performance advantages that could come by including information about the problem structure.

3.2.2 Grounding: Benchmark Problems.

So far, we have grounded our approach on two classical CSPs, namely the N-queen completion and Partial Latin Square (PLS, see [44]) problems. Classical problems let us work in a controlled setting with well known properties [87, 92, 93], and simplifies drawing scientific conclusions.

The N-queen completion problem consist in placing n queens pieces on a $n \times n$ chessboard, so that no queen threats another. The PLS problem consist in filling an $n \times n$ square with numbers from 1 to n so that the same number appears only once per row and column. In both cases, some variables may be pre-assigned. We focus on N-queen problems of size 8 and PLSs of size 10. In both cases, we model assignments using a one-hot encoding, leading to vector of size $8 \times 8 = 64$ for the n-queens and $10 \times 10 \times 10 = 1,000$ for the PLS.

For the 8-queen problem, we have used 1/4 of the 12 non-symmetric solution to seed the training set, and the remaining ones for the test set. Both the training and the test set are then obtained by generating all the symmetric equivalents, and then by applying systematic deconstruction to the resulting solutions.

For the PLS, we have used an unbiased random generation method to obtain two "raw" datasets, respectively containing 10,000 and 20,000 solutions. The numbers are considerably large in this case, but they are very small compared to the number of size 10 PLS ($\sim 10^{31}$). As comparison, it is a bit like making sense of the layout of Manhattan from ~ 0.75 square nanometers of surface scattered all over the place. Each of the raw datasets is split into a training and test set, containing respectively 1/4 and 3/4 of the solutions. The actual examples have then been obtained by random deconstruction.

3.2.3 Grounding: Networks and Training.

As in the previous Chapter, we have chosen to use pre-activated residual networks. We have trained the networks in a supervised fashion, using 10% of the examples (chosen at random) as a validation set. The loss function is the negative log-likelihood of the target class, with a 10^{-4} L1 regularization coefficient. The choice of the network and training hyper-parameters has been made after an informal tuning. We have eventually settled for using the Adam [130] optimizer, with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The initial learning rate α_0, was progressively annealed through epochs with decay proportional to training epoch t, resulting in a learning

rate $\alpha = \frac{\alpha_0}{1+k\times t}$ with $k = 10^{-3}$. Training was stopped after there was no improvement on the validation accuracy for e epochs.

For the 8-queens problem we have used an initial layer of 200 neurons, than 100 residual blocks, each one composed by two layers of 500 and 200 neurons, and finally an output layer of 64 neurons, for a total of more than 200 layers. Batch optimization has been employed, using a initial learning rate of $\alpha_0 = 0.1$ and a patience of $e = 200$ epochs. Dropout [245] has been applied to each input and hidden neuron with probability $p = 0.1$.

For the PLS problem, we have used a smaller network because of the bigger input/output vectors and the larger datasets would have required too much training time. Therefore we have used an initial layer of 200 neurons, then 10 residual blocks, each one composed by two layers of 300 and 200 neurons, and a final output layer of 1000 neurons, for a total of 22 layers. Mini-batch optimization has been employed, using shuffling in each epoch, using a initial learning rate of $\alpha_0 = 0.03$ and a patience of $e = 50$ epochs. Dropout has been applied to hidden neuron with probability $p = 0.1$. The size of the mini batch has been setted to 50,000 for the training on the 10k dataset and to 100,000 for the 20k dataset.

3.3 Experimentation

We designed our experiments to address four main questions. First, we want to assess how well the DNNs are actually learning their designated task, i.e. to guess the "correct" assignment according to the employed deconstruction method. Second, we are interested in whether the DNNs learn to generate feasible assignments, no matter whether those are "correct" according to datasets. Third, assuming that the networks are actually learning something about the problem constraints, it makes sense to check whether some constraint types are learned better than others. Finally, we want to investigate whether using the DNNs to guide an actual tree search process leads to a reduction in the number of fails.

3.3.1 Network Accuracy.

Here we are interested in assessing the performance of our DNNs in their natural task, i.e. learning "correct" variable-value assignment, as defined by our deconstruction procedure. Figure 3.1 shows the accuracy reached by our DNNs on both the training and the test sets, grouped by the number of pre-assigned variables in the example input. For comparison, random guessing would reach an accuracy of $1/64 \simeq 0.015$ for the 8-queens and $1/1,000$ for the PLS. There are three notable facts:

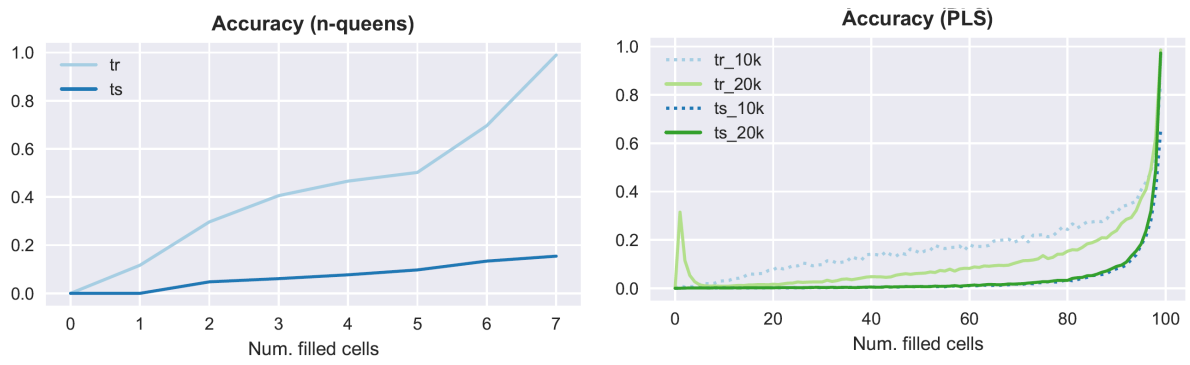


Figure 3.1: Accuracy on the training and test sets

1. The accuracy is at least one order or magnitude larger than random guessing, but still rather low, in particular for the PLS; this suggest that he networks are not doing particularly well at the task they are being trained for.

2. Second, the accuracy on the test set if considerably lower than on the training set; normally this is symptomatic of overfitting, but in this case there is also a structural reason. The pruning in the last phase of dataset generation introduces a degree of ambiguity in our training: as an extreme case, for the same partial assignment, the training and the test set may report different "correct" assignments that cannot be both predicted correctly.

3. Third, the accuracy tends to increase with the number of filled cells. Having many filled cells means having very few feasible completions, and therefore it is more unlikely for the same instance to appear both in the test and train set with a different target. In this situation it is intuitively easier for the network to label a specific assignment as the "correct" one.

The third observation leaves an open question: while the small number of feasible completions can explain why the accuracy raises, it fails to explain the magnitude of the increase. The result would be much easier to explain by assuming that the DNN has somehow learned something about the problem constraints.

3.3.2 Feasibility Ratio

It makes sense to evaluate the ability of the DNNs to yield globally consistent assignments, even if those are not chosen as "correct", since this is our primary goal. Figure 3.2 show the ratio of predictions of the DNNs (both on the training and test set) that could be expanded

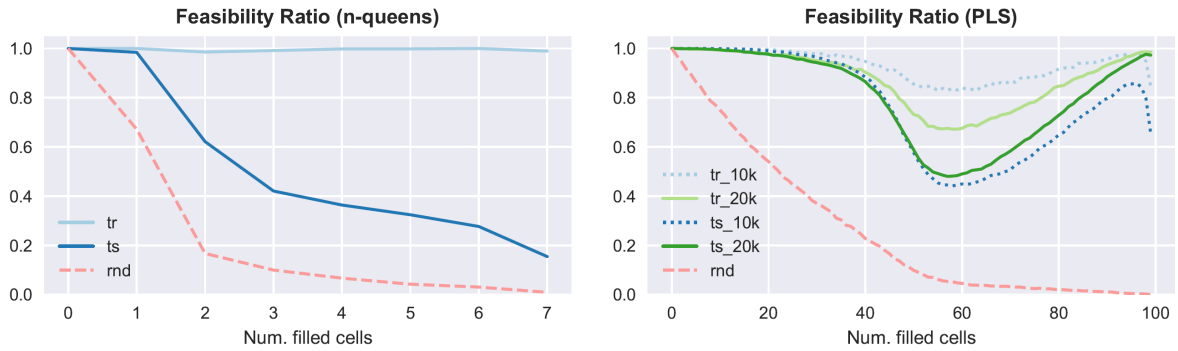


Figure 3.2: Feasibility ratios on the training and test sets

to full solutions. For comparison, the figures report also the results that can be obtained by guessing at random on the test sets. There are three very relevant observations to make:

1. There is a striking difference between the accuracy values from Figure 3.1 and the feasibility ratios.

2. Such discrepancy may be due to the fact that the more a partial solution is empty, the more are the feasible assignments that can be found even by guessing. However, the reported feasibility ratio are also significantly higher than the random baseline. This is hard to explain, unless we assume that the DNNs have somehow learned the semantic of the problem constraints.

3. The feasibility ratios for the PLS networks have a dip between 50 and 60 pre-assigned variables, and then tend to raise again. This is exactly the behavior that one would expect thank to constraint propagation: when many variables are bound many values are pruned and the number of available assignments is reduced. However, the DNNs at this stage do not rely on propagation at all. Even the higher accuracy from Figure 3.1 is not enough to justify how much the feasibility rations tend to increase for almost full solutions. Assuming that the DNN has learned the problem constraints can explain the increase, but not so easily the dip.

3.3.3 Constraints Preference

Next, we have designed an experiment to investigate whether some constraints are handled better than others. We start by generating a pool of (partial) solutions by using the DNN to guide a randomized constructive heuristic. Given a partial solution, we use the DNN to obtain a probability distribution over all possible assignments, one of which is chosen randomly and performed. Starting from an empty solution, the process is repeated as many times as there are

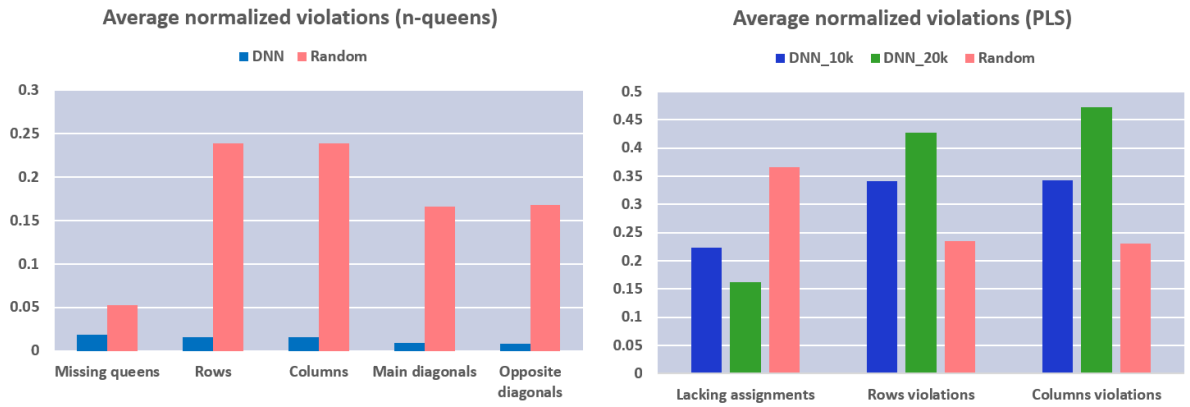


Figure 3.3: Average violations on the two problems

variables, and relies on our low-level, bit vector, representation of the partial solution. As a consequence, at the end of the process there may be variables that have been "assigned multiple times", and therefore also unassigned variables. We have used this approach to generate 10,000 solutions for each DNN, and for comparison we have done the same using a uniform distribution.

Once we have such a pool of partial solutions, we count the average degree of violation of each abstract problem constraints, e.g., the number of rows with multiple queens. Each quantity is then normalized over the corresponding maximum (i.e., the number of row/columns, or the number of variables). Looking at the average violations for the random baseline intuitively tells the natural difficulty of satisfying a constraint type. Comparing such values with those of the DNN allows to evaluate how well the DNN is faring.

As reported in Figure 3.3, for the N-queens problem the network gets much closer to feasibility than the random baseline and all problem constraints are handled equally well. For the PLS problem, the DNNs violates the row and column constraints significantly more than the random baseline, but they also tend to leave fewer variable unassigned. There is a logic correlation between these two values, since assigning more variables increases the probability to violate a row or a column constraint.

3.3.4 Guiding Tree Search

Finally, we have tried using our DNNs to guide a Depth First Search process for the Partial Latin Square[1]. In particular, we always make the assignment with the largest score, excluding bounded variables and values pruned by propagation.

[1]The 8-queens problem is too easy to provide meaningful measurements.

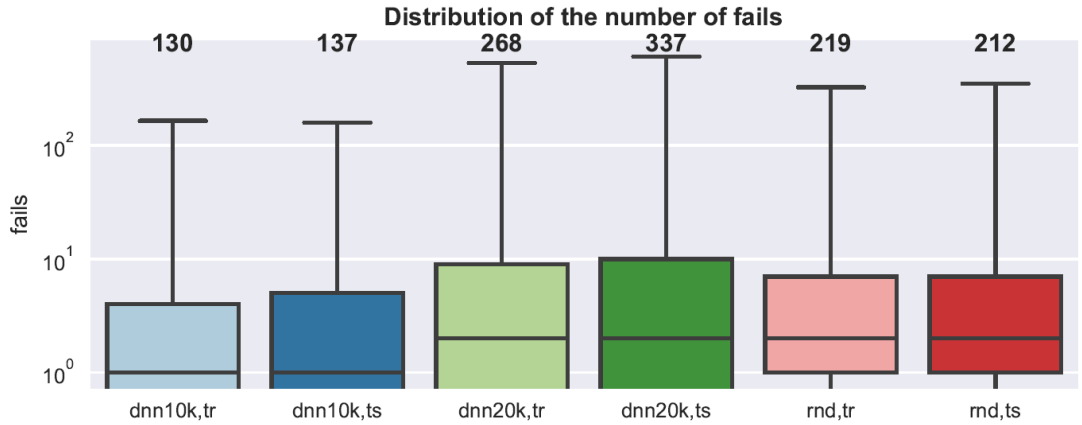


Figure 3.4: Distribution of the number of fails on the train and test sets. At the top of each box we report the number of times the fail cap was reached.

We employ a classic CP model for the PLS (one finite domain variable per cell), and use the GAC ALLDIFFERENT propagator for the row and column constraint. We compare the results of the two DNNs with those of heuristic that pick uniformly at random both the variable and the value to be assigned. Given that our research is at a early stage, we have opted for a simple (but inefficient) implementation relying on the Google or-tools python API: for this reason we focus our evaluation on the number of fails. As a benchmark, we have sampled 4,000 partial solutions from the 20k training and test set, at the complexity peak. All instances have been solved with a cap at 10,000 fails.

The results of this experimentation are reported in Figure 3.4, using box plots. Apparently, the DNN trained on the 10k dataset is more efficient than the random baseline, but the opposite holds for the one trained in the 20k dataset. This matches the results obtained in our analysis of violated constraints, but not those obtained for the feasibility ratio. We suspect however that explaining the performance (and obtaining practical speed-up) will require to take into account the complex trade-off making choice that are likely feasible, and recovering quickly from the inevitable mistakes. This is a well know open problem in Constraint Programming, that we plan to tackle as part of future work.

3.4 Discussion

We have performed a preliminary investigation to understand whether DNNs can learn how to solve combinatorial problems. We have adopted a general setup, totally agnostic to the problem structure, and we have trained the networks on arbitrarily chosen solution construction sequences.

Our experimentation has provided evidence that, despite having low accuracy at training time, a DNN can become capable of generating globally consistent variable-value assignments. This cannot be explained assuming that the networks only mimic the assignment sequences in the training set, but it is compatible with the hypothesis that the DNNs have learned something about the problem structure. The networks do not seem to favor any abstract constraint in particular, suggesting that what they are learning does not match our usual understanding of CSPs. When used for guiding a search process, our DNNs have provided mixed results, highlighting that achieving performance improvements may require to deal more explicitly with the peculiarities of a specific solution technique (e.g., constraint propagation).

This research line is still at an early stage: there are considerable overheads that make practical applications still far, and the described method is limited to problems of fixed size, a problem that maybe could be solved using only convolutional layers. However, we believe the approach to have enough potential to deserve further investigation.

Chapter 4

Considerations

In Part I of this dissertation, we have demonstrated that the information neural networks can learn about a generic problem may extend beyond the simple imitation of training instructions. Indeed, we have demonstrated that networks can learn to respect the rules of a game and, to a certain degree, to respect constraints in CSPs, without relying on any a priori information.

A few other works have recently pushed our investigation further. Hottung et al. [113] have used a similar approach for optimization problems, training a NN on near-optimal solutions to guide a heuristic search. Silvestri et al. [237] have extended our work via an approach inspired by Semantic Based Regularization [56], analyzing how injecting incomplete knowledge at training time may improve robustness, especially when the available history of solutions is small.

These experiments suggest that the networks' internal representation of such problems may not match our usual human representation and understanding of them, but we have not been able to very this hypothesis further. Our framework does not allow us to inspect what the network may have learned, nor to freely test properties of the acquired knowledge. Indeed, our hypotheses are based on observations of the networks' behavior, empirically tested using a tailored implementation, without any way to interpret the sub-symbolic knowledge stored inside them. From these observations, we conclude that there are two elements that our research should consider:

- A method to increase the interpretability of neural networks, without compromising their ability. The method on which we have decided to focus on is Neural Attention [76].
- A symbolic framework that allows us to query the models without implementing ad-hoc tests, but rather rely on a symbolic representation of properties and rules. After considering many different frameworks, we have chosen the LTN framework [59, 60, 228, 229].

Moreover, we have decided to change our domain of study. Indeed, game-playing and CSPs real-world scenarios have requirements regarding efficiency and optimization that would probably benefit more from specifically tailored solutions. With the purpose to conduct our research on a benchmark that is still challenging for both symbolic and subsymbolic approaches, and where a model-agnostic approach could bring positive contributes, in the rest of this thesis we will address the task of Argument Mining.

Part II

Neural Architectures for Argument Mining

Chapter 5

Argument Mining

Argument Mining (AM)[1] is a discipline that stems from Natural Language Processing (NLP) and Knowledge Representation and Reasoning (KRR) [27] with the goal to automatically extract arguments and their relations from a given natural language document [159]. It has been defined as "the general task of analyzing discourse on the pragmatics level and applying a certain argumentation theory to model and automatically analyze the data at hand" [103]. AM's purpose is therefore the extraction of structured information from raw textual sources, giving an understanding of the relations between single arguments and the complex network they create. Among the many useful and practical applications of such a discipline, it can be used to perform fact-checking and recognize deceptive content [42, 66], support decision making in healthcare application [181] and the legal domain, improve the peer review process [117], improve the understanding of the position of political candidates [182], support teachers in an educational context [168], and support debate technology [238].

AM can be generally divided into a series of subtasks [27, 140, 159] that are often addressed in a pipeline fashion. The *components detection* consists of extracting the arguments from the document, detecting their boundaries, and classifying them. Two examples of components type are *claims* and *evidences*. The latter are facts and objective information that are reported in the document, while the former may be opinions and hypotheses expressed by the author. The following step is the *relations prediction*, whose purpose is to establish which components are in an argumentative relationship (*link prediction*) and what type of relationship do they have (*relation classification*). An example of a relationship is *support*, for example when an evidence component does provide information based on which a claim can be made. Another example is *attack*, which can be drawn when two claims contradict each other and therefore can not both be true at the same time.

[1] Also referred to as Argumentation Mining.

While it is possible to address these subtasks independently, it is surely beneficial to address them together, since the information obtained by one of them can have a deep influence on the other. For these reasons, most of the approaches adopt a pipeline scheme, so as to exploit the knowledge gathered in the component detection to perform the more challenging task of relation prediction. Others have approached the problem so with systems that jointly learn to perform both tasks, often creating a high-level representation of the problem during the process.

The development of new advanced Deep Learning techniques for NLP has had a beneficial impact on this field, leading to great results in some AM areas. Nonetheless, more work is still required. Indeed, the argumentation model and the domain of the documents often provide information regarding constraints and rules that may regulate the type of the components and their relationships. While attempts have been made so as to integrate this knowledge into DL approaches, the proper formal integration of the two worlds in this domain has not been carried on yet.

This Chapter is structured as follows. In Section 5.1 we present the process of creation of corpora and describe the four corpora that will be used in our experimental settings. Section 5.2 presents an overview of AM approaches, with a specific focus on approaches that are the state of the art on our benchmarks. Section 5.3 concludes with a note regarding the difficulties of comparing different AM approaches and possible solutions to them.

5.1 Corpora and Resources for AM

5.1.1 Creation and Evaluation of Corpora

The creation of annotated linguistic corpora is often a challenging and expensive procedure in many NLP tasks, and this is especially true in AM. Indeed, argumentation is a complex topic, and there is no unique and universally accepted definition for its entities. On one hand, the complexity of the task requires experts to be involved. They are typically involved for the annotation job itself or for writing tailored guidelines, which may need to be improved through multiple rounds of annotation [253]. Unsupervised or self-supervised annotation approaches are still not widely used and the effective approaches rely on domain-specific heuristics [203], therefore the size of available AM corpora is modest compared to the ones available for other NLP tasks [140]. On the other hand, the existence of multiple theoretical frameworks, along with the many different ways they can be adapted to the various domains, leads to the creation of corpora based on different premises. These differences make it impossible to combine multiple resources in a single corpus, at least not without a great effort in pre-processing. The

outcome is the impossibility to create a wide dataset on which it would be possible to train general-purpose models.

The quality of manual annotations directly impacts machine learning methods [213], therefore it is important to verify their quality to better understand a corpus. Since there is no objective method to evaluate this aspect, inter-annotator agreement (IAA) is used: at least two annotators are required to annotate the same document(s) in parallel, then their work is compared and agreement metrics are computed. Many of these metrics exist [6], and their interpretation can sometimes be difficult [179]. To reduce the cost of annotation, sometimes this procedure is not applied to the whole corpus, but only on a limited amount of documents, while the rest of the corpus is annotated by a single person.

One of the metrics that is used most frequently is Cohen's Kappa [43], which measures the agreement between two annotators, considering also the chance of a random agreement. The definition of such coefficient is formulated in Eq 5.1, where p_o is the probability for the annotators to agree (the main diagonal of the matrix), while p_e is the probability for them to agree by chance. If p_x is the probability of the annotators to randomly agree on a class x, p_e is given as summation of p_x for each class x. Since this metric approaches F1-measure when the number of negative instances grows, another possibility is to use the average F-measure among pairs of experts [114].

$$\kappa = (p_o - p_e)/(1 - p_e) \tag{5.1}$$

Another popular metric is Fleiss' kappa [75], also called multi-π, which is applicable for any number of raters giving categorical ratings. It can be interpreted as the difference between the agreement observed between the annotators and what would be expected if all the annotations were done completely random [6]. Finally, Krippendorff's alpha [134] is a generalization of several known reliability indices, making it possible to compare different data using the same reliable standard. Among its characteristics, it allows specifying a weight for each type of disagreement. According to Krippendorff's guidelines, data should be considered reliable when $\alpha \geq 0.8$, and acceptable when $\alpha \geq 0.667$.

5.1.2 Cornell eRulemaking Corpus (CDCP)

The Cornell eRulemaking Corpus (CDCP) [194, 197] consist of user-generated documents in which specific regulations are discussed. The authors have collected user comments from an eRulemaking website[2] on the topic of Consumer Debt Collection Practices (CDCP) rule by the

[2] www.regulationroom.org

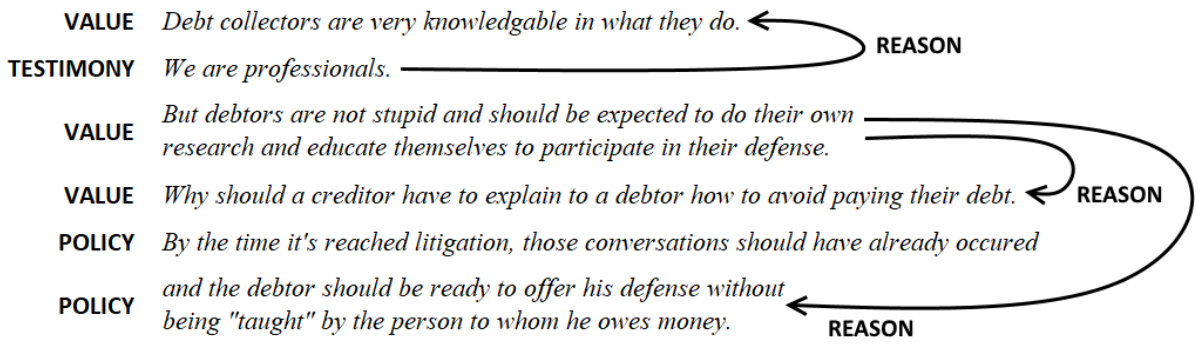


Figure 5.1: Argumentation structure in one of the documents of the CDCP corpus.

Consumer Financial Protection Bureau (CFPB). The corpus contains 731 user comments, for a total of about 4,700 components, all considered to be argumentative.

Since the comments are created by users, they are not structured, and more often than not, they present grammatical errors, typos, and do not follow usual writing conventions (such as the blank space after the period mark). This characteristic makes the corpus quite challenging to pre-process, since most of the off-the-shelf tools may fail even in simple tasks such as tokenization.

The annotation of the argument models follows the model proposed by Park et al. [198], where links are constrained to form directed graphs. The corpus is suitable for all the sub-tasks of argument mining since it presents 5 classes of propositions and two types of links. The original version of the corpus [197] does contain documents where there are nested propositions, and where the transitive closure is not guaranteed. These characteristics introduce considerable complexity to a corpus which is already challenging for its document, therefore for the rest of this work, we will instead consider the pre-processed version of CDCP [194], where these problems have been fixed.

The components are addressed as propositions, and they consist of a sentence or a clause. Propositions are divided into POLICY (17%), VALUE (45%), FACT (16%), TESTIMONY (21%) and REFERENCE (1%). Out of more than 43,000 possible proposition pairs, only 3% of them are linked; almost all of them are labeled as REASON (97%), while only a few are labeled as EVIDENCE (3%). Figure 5.1 shows an annotated document from the CDCP corpus.

The unstructured nature of documents, the strong unbalance between the classes, and the presence of noise make the corpus particularly challenging for all the subtasks of argument mining, especially the ones which involve the relationships between components.

Regarding the process of annotation, each document was annotated by two annotators, and a third one resolved conflicts. Inter-annotator agreement between the 2 annotators reached

Krippendorf's α of 64.8% for components and 44.1% for links.[3] Most of the disagreement regarding annotation of components occurred between VALUE vs TESTIMONY and VALUE vs FACT.

5.1.3 Dr. Inventor Argumentative Corpus

This resource is the result of an extension [137] of the original Dr. Inventor Corpus [74], adding an annotation layer containing argumentative components and relations. The corpus consists of 40 scientific publications from computer graphics, which contain about 12,000 argumentative components, and contains also annotations for tasks related to Discourse Role, Citations, Subjective Aspect, and Summarization.

The classes of argumentative components are DATA (4093), OWN CLAIM (5445), and BACKGROUND CLAIM (2751). The former two are related to the concepts of premises and claims, while the latter is something in between since it is a claim related to the background, for example, made by another author in a previous work. The relation classes are SUPPORTS (5790), CONTRADICTS (696), and SEMANTICALLY SAME (44), since it is common practice in scientific publications to re-iterate the same claim (or more rarely the same data) multiple times.

Since the corpus includes documents where the structure of the discourse is complex, and data are often presented along with claims, it makes argument mining more challenging: in more than 1,000 cases some components are split into multiple text sequences, located in non-contiguous parts of the documents. This phenomenon mostly concerns claims, but data are affected too, in fewer cases. This introduces the difficulty of recognizing different segments of the documents as part of a single component and makes link prediction more difficult to address through non-pipeline approaches.

The annotation was performed by 4 persons, one of whom was a computational linguistic expert. In a preliminary phase all annotators have worked on the same 5 documents, so as to train them. The final IAA amounted to a macro F1 score of 73 for components and 47 for relations considering relaxed criteria regarding the boundaries of the components, and about 10 percentage points less in both the score considering strict criteria. After this training phase, the remaining documents have been evenly split to be annotated by a single person.

The unbalanced distribution between the 3 classes and the presence of split components makes this corpus quite challenging for link prediction, a difficulty highlighted by the low IAA on the task.

[3]The IAA for link prediction is measured treating IDs of supported elementary units as labels for the supporting elementary units.

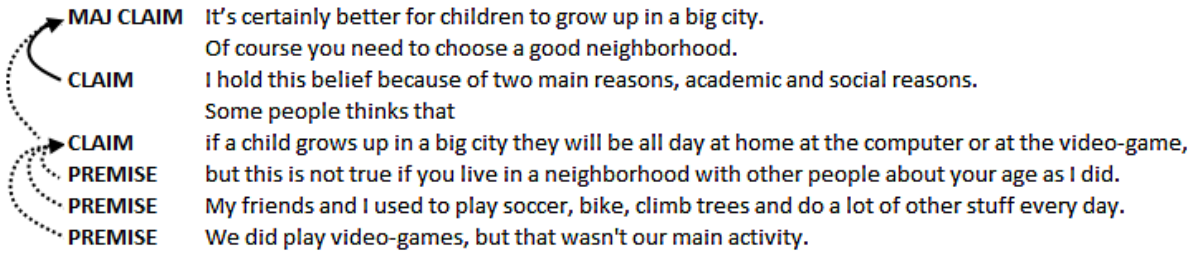


Figure 5.2: Argumentation structure in an excerpt from one of the documents of the UKP corpus. Support links are represented using continuous lines, while dashed lines represent attack relations.

5.1.4 Persuasive Essays Corpus (UKP)

The Persuasive Essays Corpus [247] consist of 402 documents coming from an online community[4] were users post essays and similar material, provide feedback, and advise each other. The dataset is divided into a test split of 80 essays and a training split with the remaining documents.

The argumentative components belong to one of three classes: MAJOR CLAIM (751), CLAIM (1,506), and PREMISE (3,832). Premises and claims can have a relation of SUPPORT (3,613) or ATTACK (219), while claims and major claims' relations are encoded in an attribute called stance.[5] The argumentation structure is modeled following a rigid scheme that imposes many constraints. The argumentation graph consists of various tree connected to a single common root, which is the major claim. The children of the major claim are claims, which can have only premises among their descendants. Also, relations can exist only between components that belong to the same paragraph and premises can have only one outgoing relation. Finally, the structure of the argumentation follows conventions that are very specific to the domain. For example, in most cases, the MAJOR CLAIM is present in the first or the last paragraph of the document, and in more than 50% of cases the paragraph does not contain any other argumentative component. An example of the argumentative structure of the UKP dataset is shown in Figure 5.2.

The annotation was performed by one expert annotator and two other persons who have annotated independently a random subset of 80 documents. On these documents, the measured IAA is quite good, with Krippendorf's α of 76.7% for boundaries detection and component classification, with a solid agreement regarding premises and major claims, while claims obtain lesser results. For what concerns argumentative relations, they reach a Fleiss' κ around 0.7. The remaining 322 essays were annotated by the expert.

[4] essayforum.com.

[5] In this work we will not address the task of stance prediction, therefore this aspect will not be considered further on.

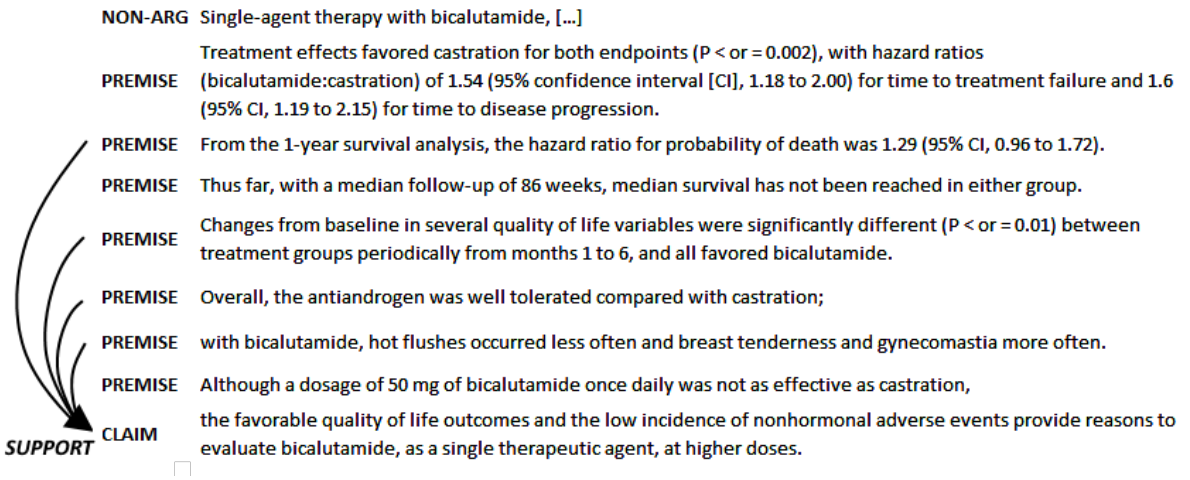


Figure 5.3: Argumentation structure in one of the documents of the AbstRCT corpus.

5.1.5 AbstRCT

The AbstRCT Corpus [181] extends a previous work [180], and consists of abstracts of scientific papers regarding randomized control trials for the treatment of specific diseases (i.e. neoplasm, glaucoma, hypertension, hepatitis b, diabetes). The final corpus contains 659 abstracts, for a total of about 4000 argumentative components. The dataset is divided into three parts: neoplasm, glaucoma, and mixed. The first one contains 500 abstracts about neoplasm, divided into train (350), test (100), and validation (50) splits. The remaining two are designed to be test sets. One contains 100 abstracts for glaucoma, while the other contains 20 abstracts for each disease[6].

Components are labeled as EVIDENCE (2808) and CLAIM (1390), while relations are labeled as SUPPORT (2259) and ATTACK (342)[7]. Out of 25,000 possible pairs of components, about 10% of them have a relationship. The argumentative model chosen for annotation enforces only one constraint regarding the structure of the resulting argumentation graph: claims can have an outgoing link only to other claims. An example of an annotated document from AbstRCT is shown in Figure5.3.

The annotation of the neoplasm abstracts was performed by two computational linguistics experts. IAA has been evaluated on 30 documents, resulting in a Fleiss' kappa of 0.72 for argumentative components and 0.68 for the more fine-grained distinction between claims and evidence, meaning substantial agreement for both tasks.

[6]Glaucoma and neoplasm documents of the mixed set are present also in the respective test set.

[7]The corpus allows also the distinction between CLAIM/MAJOR CLAIM and ATTACK/PARTIAL ATTACK. For the sake of consistency with previous works, this detail will not be considered.

5.2 State of the Art in AM

Due to the absence of large corpora and the complexity of the task at hand, the use of deep learning approaches on AM is relatively recent. Indeed, until a few years ago, researchers were more focused on the definition of specific features, often tailored to a specific corpus. The differences between corpora, both regarding the domain and the theoretical framework followed during the annotation process, force researchers to test a model on the same corpora on which it was trained, and to the best of our knowledge, transfer learning approaches have not seen wide experimentation. These two elements lead to the common practice to define a method or a model and validate it only on a single corpora [159].

In the domain of persuasive essays, Eger et al. [69] consider several sub-tasks of argument mining, making use of various neural architectures. These include neural parsers [67, 131], LSTMs for joint entity and relation extraction (LSTM-ER) [186], and Bidirectional LSTM coupled with Conditional Random Fields and Convolutional Neural Networks (BLCC) [171] in a multi-task learning framework [240]. Eger et al. conclude that neural networks can outperform feature-based techniques in argument mining tasks.

Schulz et al. [226] investigate multi-task learning (MTL) settings, addressing component detection on 5 datasets as 5 different tasks. Their architecture is composed of a CRF layer on top of a biLSTM, whose recurrent layers are shared across the tasks. They obtain positive results, and the MTL setting shows to be beneficial especially for small datasets, even if the auxiliary AM tasks involve different domains and even different component classes. Lauscher et al. [138] analyze an MTL setting where rhetorical classification tasks are performed along with components detection. They use a hierarchical attention-based model so as to perform both word-level and sentence-level tasks with the same neural architecture. The results show improvements in the rhetorical tasks, but not in AM.

Convolutional Neural Networks and LSTMs have been used by Guggilla et al. [98] to perform claim classification, whereas bidirectional LSTMs have been exploited by Cocarascu and Toni [41, 42] to classify relations and by Habernal and Gurevych [102] to assess the persuasiveness of arguments. More recently, neural networks have been applied to the task of topic-dependent evidence detection [234], improving the performance on a manually labeled corpus through the use of unsupervised data.

Among the AM works that use neural attention, Suhartono et al. [250] integrate hierarchical attention and biGRU for the analysis of the quality of the argument, Lin et al. [152] use attention to integrate sentiment lexicon, while in other works [101, 244, 248] attention module are stacked on top of recurrent layers. Potash et al. [207] tackle argument mining through Pointer Networks [265], an attention-based architecture. Finally, some researchers rely on

Transformer architecture, completely replacing recurrent approaches. These cases will be discussed in a dedicated following section.

5.2.1 Structured Learning

The best approach on the CDCP corpus so far is the work described by the corpus authors themselves [194]. They use a structured learning framework based on factor graphs to jointly classify all the propositions in a document and determine which ones are linked together. To perform the classification, the models heavily rely on a priori knowledge, encoded as factors and constraints. The unary factors represent the model's belief in each possible class for each proposition or link, without considering any other proposition or link. For each link between two propositions, the compatibility factors influence link classification according to the proposition classes, taking into account adjacency between propositions and precedence between source and target. The second-order factors influence the classification of pairs of links that share a common proposition, by modeling three local argumentation graph structures: grandparent, sibling, and co-parent. Constraints are introduced to enforce adherence to the desired argumentation structure, according to the argument model and domain characteristics.

The authors discuss experiments with 6 different models, which differ by complexity (the type of factors and constraints involved) and by how they model the factors (SVMs and RNNs). The RNN models compute sentence embeddings, by exploiting initialization with GloVe word vectors, while the SVMs models rely on many specific features. The first-order factors rely on the same features used by Stab and Gurevych [247], both for the propositions and the links. This feature set contains unigrams, dependency tuples, token statistics, proposition statistics, proposition location, indicators from hand-crafted lexicons and handcrafted ones, shared phrases, subclauses, depth of the parse tree, tense of the main verb, modal verbs, POS, production rules, type probability, discourse triplets [154], and average GloVe embeddings. The higher-order factors exploit the following features between all three propositions and between each pair: same sentence indicators, proposition order, Jaccard similarity, presence of any shared nouns, and shared noun ratios. The overall feature dimensionality is reportedly 7000 for propositions and 2100 for links, not counting 35 second-order features.

5.2.2 Transformer-based approaches

Reimers et al. [216] have been the first to make use of Transformer-based approaches in AM, using BERT [54] and ELMO [204] to create contextualized word embeddings. Specifically, they address the tasks of component classification and argument clustering, a related task where the aim is to identify similar arguments. Similarly, Lugini and Litman [167] use BERT embeddings

along other contextual information to perform component classification, and Wang et al. [271] use them to train a different model for each type of component. Trautmann et al. [259] use pre-trained BERT models to perform word-level classification of the stance of components regarding a given topic, while Poudyal et al. [208] use RoBERTa [162], an improved version of the original BERT, for component detection.

Mayer et al. [181] conduct extensive experimentation on AbstRCT, addressing all AM subtasks with a pipeline scheme. They analyze the impact of various BERT models, which are pre-trained on other corpora and then fine-tuned on the corpus at hand. Segmentation and component classification are performed as sequence tagging with BIO scheme. Link prediction and relation classification follow, taking into account all the pair of components obtained in the first step and classifying their relations as attack, support, or non-existing. Their architecture is based on bi-directional transformers followed by a softmax layer and various encoders. Another approach, consisting of predicting at most one related component for each component, and then classifying their relation, has been tested but yields worse results. The architectures that yield the best results are BioBERT [145], which is pre-trained on a large-scale biomedical corpus, SciBERT [13], which is pre-trained on scientific articles of various nature, and RoBERTa.

5.3 On the Difficulty of Comparing Approaches

Since many different approaches to Argument Mining are possible, sometimes it is difficult to compare results. For example, a common approach is to tackle jointly component segmentation and classification as a sequence labelling approach [69, 181, 226, 259] using BIO tagging. This means to label each word as either Begin or Inside a specific component class, or Outside any of them and therefore as non-argumentative. Another approach may be a pipeline scheme [247] where boundaries are found first and then the classification is performed on each component independently. The comparison between these two approaches on components classification is not straightforward. One possibility is to evaluate both of them on word-level, transferring the labels obtained in the second approach from the components to their words. Alternatively, it is possible to extract the components from the first approach and evaluate both on component-level: for each component of the gold standard, firstly it is checked if a predicted component that shares more than a threshold percentage of tokens does exist (50% is a common threshold), then if it does, the classes of the two are compared. Even when the component detection is performed at sentence level, it is tricky to compare approaches that discriminate between argumentative and non-argumentative sentences before classifying the argumentative ones [104, 208, 247], and approaches that consider non-argumentative as one of the possible classes [181, 259].

The problem becomes even more tricky when link prediction and relation identification are taken into account. First of all, some works do tackle these two tasks as a pipeline [247], other do it jointly as a multi-class classification where one of the options is no-relation [194]. But the complexity of the problem increases when the first subtasks are taken into account. Taking into account the previous example, the first approach may involve in the BIO tagging also a label that specifies the distance of the linked component [69], while the pipeline scheme may perform a classification of every possible pair of components [181].

Finally, it is difficult also to compare works that address multiple subtasks and works that do not address the early stages of the pipeline, such as the ones that do not tackle boundary detection [194]. Indeed, the former do carry over the errors related to the first steps, while the latter do not. Therefore it is debatable whether, for the sake of comparison on the same single subtask, it is fairer to evaluate them on the same exact test set, or the instances on which the former method has failed the previous steps should be not considered for it, or if those instances should be excluded from the test set for both the approaches.

For all these reasons, we shall remark that the adjustments and approximations used to make two methods comparable can greatly influence the results. They will hardly completely subvert the evaluation of a technique, but two methods with similar performances may end up yielding significantly different outcomes, depending on how the results have been processed for evaluation

Chapter 6

Neural Attention

Attention is an increasingly popular mechanism used in a wide range of neural architectures. The mechanism itself has been realized in a variety of formats. However, because of the fast-paced advances in this domain, a systematic overview of attention is still missing. In this chapter we lay the background about neural attention, which we have used in the experiment that will be described in the following chapter. We define a unified model for attention architectures in natural language processing, with a focus on those designed to work with vector representations of the textual data. We propose an original taxonomy of attention models according to four dimensions: the representation of the input, the compatibility function, the distribution function, and the multiplicity of the input and/or output. We present examples of how prior information can be exploited in attention models, and discuss ongoing research efforts and open challenges in the area, providing the first extensive categorization of the vast body of literature in this exciting domain. The content of this chapter is further expanded in our survey on the same topic [76].

6.1 Introduction to Neural Attention

In many problems that involve the processing of natural language, the elements composing the source text are characterized by having each a different relevance to the task at hand. For instance, in aspect-based sentiment analysis, cue words such as “good" or “bad" could be relevant to some aspects under consideration, but not to others. In machine translation, some words in the source text could be irrelevant in the translation of the next word. In a visual question-answering task, background pixels could be irrelevant in answering a question regarding an object in the foreground, but relevant to questions regarding the scenery.

Arguably, effective solutions to such problems should factor in a notion of relevance, so as to focus the computational resources on a restricted set of important elements. One

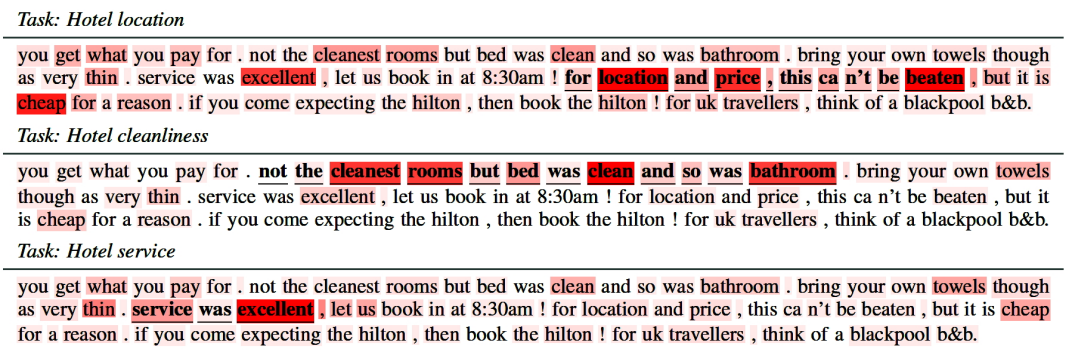


Figure 6.1: Example of attention visualization for an aspect-based sentiment analysis task, from [8, Figure 6]. Words are highlighted according to attention scores. Phrases in bold are the words considered relevant for the task, or human *rationales*.

possible approach would be to tailor solutions to the specific genre at hand, in order to better exploit known regularities of the input, by feature engineering. For example, in the argumentative analysis of persuasive essays, one could decide to give special emphasis to the final sentence. However, such an approach is not always viable, especially if the input is long or very information-rich, like in text summarization, where the output is the condensed version of a possibly lengthy text sequence. Another approach of increasing popularity amounts to machine-learning the relevance of input elements. In that way, neural architectures could automatically weigh the relevance of any region of the input, and take such a weight into account while performing the main task. The commonest solution to this problem is a mechanism known as *attention*.

Attention was first introduced in natural language processing (NLP) for machine translation tasks by [7]. However, the idea of *glimpses* had already been proposed in computer vision by Larochelle and Hinton [136], following the observation that biological retinas fixate on relevant parts of the optic array, while resolution falls off rapidly with eccentricity. The term *visual attention* became especially popular after Mnih et al. [187] significantly outperformed the state of the art in several image classification tasks as well as in dynamic visual control problems such as object tracking thanks to an architecture that could adaptively select and then process a sequence of regions or locations at high resolution, and use a progressively lower resolution for further pixels.

Besides offering a performance gain, the attention mechanism can also be used as a tool for interpreting the behaviour of neural architectures, which are notoriously difficult to understand. Indeed, neural networks are sub-symbolic architectures, therefore the knowledge they gather is stored in numeric elements that do not provide any means of interpretation by themselves. It then becomes hard if not impossible to pinpoint the reasons behind the wrong output of a

neural architecture. Interestingly, attention could provide a key to partially interpret and explain neural network behaviour [40, 99, 147, 262, 275], even if it cannot be considered a reliable means of explanation [124, 230]. For instance, the weights computed by attention could point us to relevant information discarded by the neural network or to irrelevant elements of the input source that have been factored in and could explain a surprising output of the neural network. Therefore, visual highlights of attention weights could be instrumental to analyzing the outcome of neural networks, and a number of specific tools have been devised for such a visualization [144, 161]. Figure 6.1 shows an example of attention visualization in the context of aspect-based sentiment analysis.

For all these reasons, attention has become an increasingly common ingredient of neural architectures for NLP [85, 285]. Besides NLP and computer vision [96, 278, 291], attentive models have been successfully adopted in many other different fields, such as speech recognition [30, 38, 243], recommendation [273, 284], time-series analysis [241, 258], games [37], and mathematical problems [71, 132, 265].

In NLP, after an initial exploration by a number of seminal papers [7, 251], a fast-paced development of new attention models and attentive architectures ensued, resulting in a highly diversified architectural landscape. Because of, and adding to, the overall complexity, it is not unheard of different authors who have been independently following similar intuitions leading to the development of almost identical attention models. For instance, the concepts of *inner attention* [269] and *word attention* [277] are arguably one and the same. Unsurprisingly, the same terms have been introduced by different authors to define different concepts, thus further adding to the ambiguity in the literature. For example, the term *context vector* is used with different meanings by Bahdanau et al. [7], Wang et al. [274], Yang et al. [282].

In this Chapter, we offer a systematic overview of attention models developed for NLP. To this end, we provide a general model of attention for NLP tasks, and use it to chart the major research activities in this area. We also introduce a taxonomy that describes the existing approaches along four dimensions: input representation, compatibility function, distribution function, and input/output multiplicity. To the best of our knowledge, this is the first taxonomy of attention models. Accordingly, we provide a succinct description of each attention model, compare models with one another, and offer insights on their use. Moreover, we present examples regarding the use of prior information in unison with attention, debate about the possible future uses of attention, and describe some interesting open challenges.

We restrict our analysis to attentive architectures designed to work with vector representation of data, as it typically is the case in NLP. Readers interested in attention models for tasks where data has a graphical representation may refer to Lee et al. [146].

What do not offer is a comprehensive account of all the neural architectures for NLP (for an excellent overview see Goldberg [91]), or of all the neural architectures for NLP that use an attention mechanism. That would be impossible and would rapidly become obsolete, because of the sheer volume of new articles featuring architectures that increasingly rely on such a mechanism. Moreover, our purpose is to produce a synthesis and a critical outlook rather than a flat listing of research activities. For the same reason, we do not offer a quantitative evaluation of different types of attention mechanisms, since such mechanisms are generally embedded in larger neural network architectures devised to address specific tasks, and it would be pointless in many cases to attempt comparisons using different standards. Even for a single specific NLP task, a fair evaluation of different attention models would require experimentation with multiple neural architectures, extensive hyper-parameter tuning, and validation over a variety of benchmarks. However, attention can be applied to a multiplicity of tasks, and there are no datasets that would meaningfully cover such a variety of tasks. An empirical evaluation is thus beyond the scope of this paper. There are, however, a number of experimental studies focused on particular NLP tasks, including machine translation [25, 58, 185, 252], argumentation mining [244], text summarization [193], and sentiment analysis [147]. It is worthwhile remarking that, in several occasions, attention-based approaches enabled a dramatic development of entire research lines. In some cases, such a development has produced an immediate performance boost. That was the case, for example, with the Transformer [263] for sequence-to-sequence annotation, as well as with BERT [54], currently among the most popular architectures for the creation of embeddings. In other cases, the impact of attention-based models was even greater, paving the way to radically new approaches for some tasks. Such was the influence of Bahdanau et al.'s work [7] to the field of machine translation. Likewise, the expressive power of memory networks [251] significantly contributed to the idea of using deep networks for reasoning tasks.

This Chapter is structured as follows. In Section 6.2 we define a general model of attention and we describe its components. In Section 6.3 we elaborate on the uses of attention in various NLP tasks. Section 6.4 presents our taxonomy of attention models. Section 6.5 discusses how attention can be combined with knowledge about the task or the data. Section 6.6 is devoted to open challenges, current trends and future directions. Section 6.7 concludes.

6.2 The Attention Function

The attention mechanism is a part of a neural architecture that enables to dynamically highlight relevant features of the input data, which in NLP is typically a sequence of textual elements. It can be applied directly on the raw input, or on its higher-level representation. The core idea

Table 6.1: Notation.

Symbol	Name	Definition
x	Input sequence	Sequence of textual elements constituting the raw input.
K	Keys	Matrix of d_k vectors (k_i) of size n_k, whereupon attention weights are computed: $K \in \mathbb{R}^{n_k \times d_k}$.
V	Values	Matrix of d_k vectors (v_i) of size n_v, whereupon attention is applied: $V \in \mathbb{R}^{n_v \times d_k}$. Each v_i and its corresponding k_i offer two, possibly different, interpretations of the same entity.
q	Query	Vector of size n_q, or sequence thereof, in which respect attention is computed: $q \in \mathbb{R}^{n_q}$.
kaf qaf vaf	Annotation functions	Functions that encode the input sequence and query, producing K, q and V respectively.
e	Energy scores	Vector of size d_k, whose scalar elements (energy "scores", e_i) represent the relevance of the corresponding k_i, according to the compatibility function: $e \in \mathbb{R}^{d_k}$.
a	Attention weights	Vector of size d_k, whose scalar elements (attention "weights", a_i) represent the relevance of the corresponding k_i according to the attention model: $a \in \mathbb{R}^{d_k}$.
f	Compatibility function	Function that evaluates the relevance of K with respect to q, returning a vector of energy scores: $e = f(K,q)$.
g	Distribution function	Function that computes the attention weights from the energy scores: $a = g(e)$.
Z	Weighted values	Matrix of d_k vectors (z_i) of size n_v, representing the application of a to V: $Z \in \mathbb{R}^{n_v \times d_k}$.
c	Context vector	Vector of size n_v, offering a compact representation of Z: $c \in \mathbb{R}^{n_v}$.

behind attention is to compute a weight distribution on the input sequence, assigning higher values to more relevant elements.

The characteristics of an attention model depend on the structure of the data whereupon they operate, and on the desired output structure. The unified model we propose is based on and extends the models proposed by Daniluk et al. [48], Vaswani et al. [263]. It comprises a *core* part shared by almost the totality of the models found in the surveyed literature, as well as some additional components that, although not universally present, are still found in most literature models.

Figure 6.2 illustrates the core attention model, which is part of the general model shown in Figure 6.3. Table 6.1 lists the key terms and symbols. The core of the attention mechanism maps a sequence K of d_k vectors k_i, the *keys*, to a distribution a of d_k weights a_i. K encodes

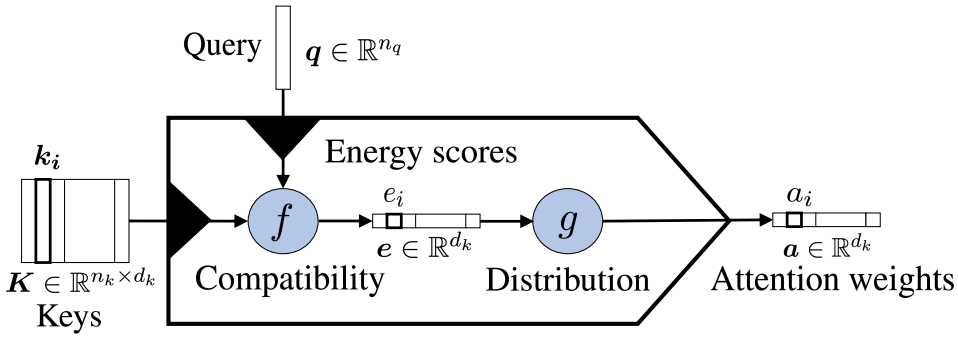


Figure 6.2: Core attention model.

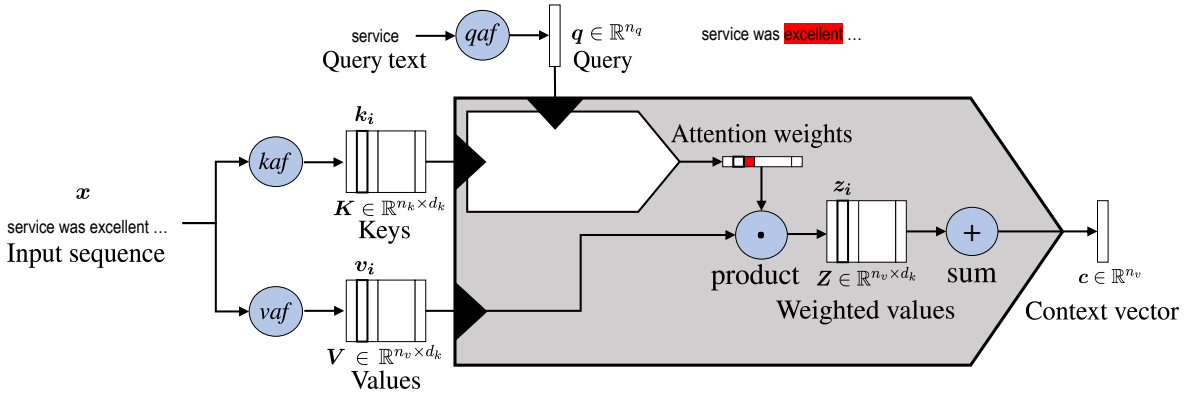


Figure 6.3: General attention model.

the data features whereupon attention is computed. For instance, K may be word or character embeddings of a document, or the internal states of a recurrent architecture. In some cases, K could include multiple features or representations of the same object (e.g., both one-hot encoding and embedding of a word), or even—if the task calls for it—representations of entire documents.

More often than not, another input element q, called *query*,[1] is used as a reference when computing the attention distribution. In that case, the attention mechanism will give emphasis to the input elements relevant to the task *according to* q. If no query is defined, attention will give emphasis to the elements *inherently* relevant to the task at hand. q may represent different entities: embeddings of actual textual queries, contextual information, background knowledge, hidden states of a bigger architecture and so on. It can also take the form of a matrix rather than a vector.

From the keys and query, a vector e of d_k energy scores e_i is computed through a *compatibility function* f (Eq. 6.1).

$$e = f(q, K) \tag{6.1}$$

Function f is sometimes called *alignment model* or *energy function* [298]. Energy scores are then transformed into attention weights using what we call a *distribution function*, g (Eq. 6.2).

$$a = g(e) \tag{6.2}$$

Such weights are the outcome of the core attention mechanism. The commonest distribution function is the softmax function which normalizes all the scores to a probability distribution. Weights represent the relevance of each element to the given task, with respect to q and K.

The computation of these weights may already be sufficient for some tasks such as the classification task addressed by Cui et al. [47]. Nevertheless, many tasks require the computation of new representation of the keys. In such cases, it is common to have another input element: a sequence V of d_k vectors v_i, the *values*, representing the data whereupon the attention computed from K and q is to be applied. Each element of V corresponds to one and only one element of K, and the two can be seen as different representations of the same data. Indeed, many architectures, do not distinguish between K and V. The distinction between keys and values was introduced by Daniluk et al. [48], who use different representations of the input for computing the attention distribution and the contextual information.

[1]The concept of "query" in attention models should not be confused with that used in tasks like question answering or information retrieval. In our model, the "query" is part of a general architecture and is task-independent.

Table 6.2: Possible uses of attention and examples of relevant task.

Use	Tasks
Feature selection	Multimodal tasks
Auxiliary task	Visual question answering Semantic role labelling
Contextual embedding creation	Machine Translation Sentiment Analysis Information Extraction
Sequence-to-sequence annotation	Machine Translation
Word selection	Dependency Parsing Cloze Question Answering
Multiple input processing	Question Answering

V and a are thus combined to obtain a new set Z of weighted representations of V (Eq. 6.3), which are then merged together so as to produce a compact representation of Z usually called the *context vector* c (Eq. 6.4).[2] The commonest way of obtaining c from Z is by summation. However, alternatives have been proposed, including gating functions [231]. Either way, c will be mainly determined by values associated with higher attention weights.

$$z_i = a_i v_i \tag{6.3}$$

$$c = \sum_{i=1}^{d_k} z_i \tag{6.4}$$

What we described so far was a synthesis of the most frequent architectural choices made in the design of attentive architectures. Other options will be explored in Section 6.4.4.

6.3 The Uses of Attention

Attention enables to estimate the relevance of the input elements as well as to combine said elements into a compact representation—the context vector—that condenses the characteristics of the most relevant elements. Because the context vector is smaller than the original input, it requires fewer computational resources to be processed at later stages, yielding a computational gain. We summarize possible uses of attention and the tasks in which they are relevant in Table 6.2.

[2] Although most authors use this terminology, we shall remark that Wang et al. [274], Yang et al. [282] and other authors use the term *context vector* to refer to other elements of the attention architecture.

For tasks such as document classification, where usually there is only K in input and no query, the attention mechanism can be seen as an instrument to encode the input into a compact form. The computation of such an embedding can be seen as a form of *feature selection*, and as such it can be applied to any set of features sharing the same representation. This applies to cases where features come from different domains, as in multi-modal tasks [288], or from different levels of a neural architecture [9], or where they simply represent different aspects of the input document [173]. Similarly, attention can also be exploited as an auxiliary task during training, so that specific features can be modeled via a multi-task setting. This holds for several scenarios, such as visual question answering [209], domain classification for natural language understanding [128], and semantic role labelling [249].

When the generation of a text sequence is required, as in machine translation, attention enables to make use of a dynamic representation of the input sequence, whereby the whole input does not have to be encoded into a single vector. At each time step, the encoding is tailored according to the task, and in particular q represents an embedding of the previous state of the decoder. More generally, the possibility to perform attention with respect to a query q allows us to create representations of the input that depend on the task *context*, creating specialized embeddings. This is particularly useful in tasks such as sentiment analysis and information extraction.

Since attention can create contextual representations of an element, it can also be used to build sequence-to-sequence annotators, without resorting to RNNs or CNNs, as suggested by Vaswani et al. [263], who rely on an attention mechanism to obtain a whole encoder/decoder architecture.

Attention can also be used as a tool for *selecting specific words*. This could be the case for example in dependency parsing [249], and in cloze question-answering tasks [47, 125]. In the former case, attention can be applied to a sentence in order to predict dependencies. In the latter, attention can be applied to a textual document or to a vocabulary to perform a classification among the words.

Finally, attention can come in handy when multiple *interacting input sequences* have to be considered in combination. In tasks such as question answering, where the input consists of two textual sequences—for instance, the question and the document, or the question and the possible answers—an input encoding can be obtained by taking into account the mutual interactions between the elements of such sequences, rather than by applying a more rigid a-priori defined model.

6.4 A Taxonomy for Attention Models

Attention models can be described on the basis of the following orthogonal dimensions: the nature of inputs (Section 6.4.1), the compatibility function (Section 6.4.2), the distribution function (Section 6.4.3), and the number of distinct inputs/outputs, which we refer to as "multiplicity" (Section 6.4.4). Moreover, attention modules can themselves be used inside larger attention models to obtain complex architectures like hierarchical-input models (Section 6.4.1), or in some multiple-input co-attention models (Section 6.4.4).

6.4.1 Input Representation

In NLP-related tasks, generally K and V are representations of parts of documents, such as sequences of characters, words, or sentences. These components are usually embedded into continuous vector representations and then processed through key/value annotation functions (called *kaf* and *vaf* in Figure 6.3), so as to obtain a hidden representation resulting in K and V. Typical annotation functions are recurrent neural (RNN) layers such as Gated Recurrent Units (GRUs) and Long Short-Term Memory networks (LSTMs), and Convolutional Neural Networks (CNNs). In this way, k_i and v_i represent an input element relative to its local context. If the layers in charge of annotation are trained together with the attention model, they can learn to encode information useful to the attention model.

Alternatively, k_i/v_i can be taken to represent each input element in isolation, rather than in context. For instance, they could be a one-hot encoding of words or characters, or a pre-trained word embedding. This results in an application of the attention mechanism directly to the raw inputs, which is a model known as *inner* attention [269]. Such a model has proven to be effective by several authors, who have exploited it in different fashions [151, 200, 263, 277]. The resulting architecture has a smaller number of layers and hyper-parameters, which reduces the computational resources needed for training.

We shall now explain in more detail two successful structures, which have become well-established building blocks of neural approaches for NLP, namely self-attention and hierarchical-input architectures.

Self-attention

We made a distinction between two input sources: the input sequence, represented by K and V, and the query, represented by q. However, some architectures compute attention only based on the input sequence. These architectures are known as *self-* or *intra-attentive* models. We shall remark, however, that these terms are used to indicate many different approaches. The

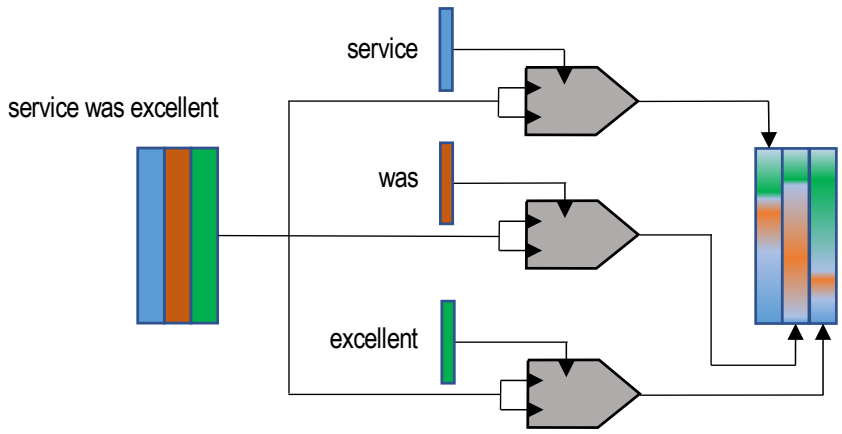


Figure 6.4: Example of use of attention in a sequence-to-sequence model.

commonest one amounts to the application of multiple steps of attention to a vector K, using the elements k_t of the same vector as query at each step [263, 291]. At each step, the weights a_i^t represent the relevance of k_i with respect to k_t, yielding d_K separate context embeddings, c^t, one per key. Attention could thus be used as a sequence-to-sequence model, as an alternative to CNNs or RNNs (see Figure 6.4). In this way, each element of the new sequence may be influenced by elements of the whole input, incorporating contextual information without any locality boundaries. This is especially interesting, since it could overcome a well-known shortcoming of RNNs: their limited ability of modeling long-range dependencies [15]. For each element k_t, the resulting distribution of the weights a^t should give more emphasis to words that strongly relate to k_t. The analysis of these distributions will therefore provide information regarding the relation between the elements inside the sequence. Modern text-sequence generation systems often rely on this approach. Another possibility is to construct a single query element q from the keys through a pooling operation. Furthermore, the same input sequence could be used both as keys K and query Q, applying a technique we will describe in Section 6.4.4, known as *co-attention*. Other self-attentive approaches, such as Lin et al. [153], Yang et al. [282], are characterized by the complete absence of any query term q, which results in simplified compatibility functions (see Section 6.4.2).

Hierarchical-Input Architectures

In some tasks, portions of input data can be meaningfully grouped together into higher-level structures. There, *hierarchical-input* attention models can be exploited to subsequently apply multiple attention modules at different levels of the composition, as illustrated in Figure 6.5.

Consider, for instance, data naturally associated with a two-level semantic structure, such as characters (the “micro”-elements) forming words (the “macro”-elements), or words forming

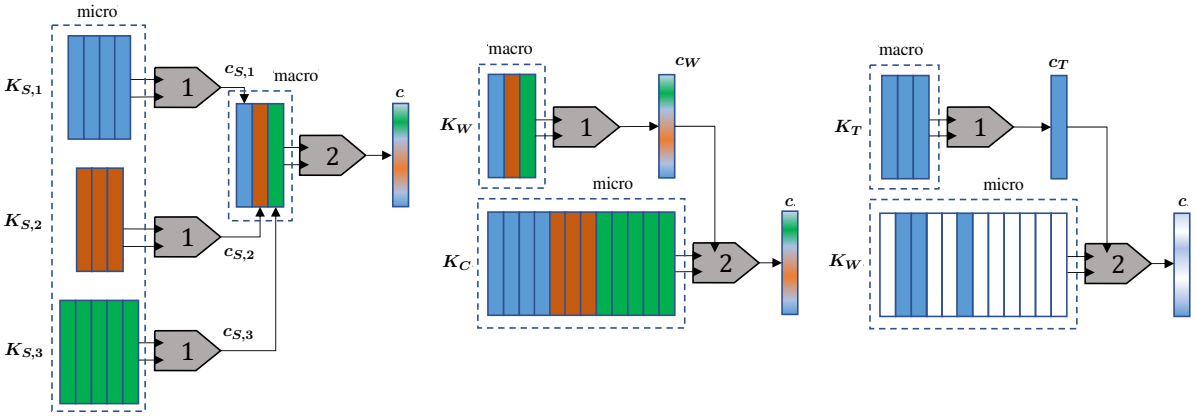


Figure 6.5: Hierarchical input attention models defined by Yang et al. [282] (left), Zhao and Zhang [298] (center), and Ma et al. [172] (right). The number inside the attention shapes indicates the order of application. Different colors highlight different parts of the inputs.

sentences. Attention can be first applied to the representations of micro elements k_i, so as to build aggregate representations k_j of the macro-elements, such as context vectors. Attention could then be applied again to the sequence of macro element embeddings, in order to compute an embedding for the whole document D. With this model, attention first highlights the most relevant micro-elements within each macro-element, and then the most relevant macro-elements in the document. For instance, Yang et al. [282] apply attention first at word level, for each sentence in turn, to compute sentence embeddings. Then, they apply attention again on the sentence embeddings to obtain a document representation.

The alternatives presented by Zhao and Zhang [298] and Ma et al. [172] refer to settings where, respectively, both micro-level and macro-level elements are available and where there is a "target" element that can be considered the macro-object.

6.4.2 Compatibility Functions

The compatibility function is a crucial part of the attention architecture, because it defines how keys and queries are matched or combined. In our presentation of compatibility functions, we will consider a data model where q and k_i are mono-dimensional vectors. For example, if K represents a document, each k_i may be the embedding of a sentence, a word or a character. In such a model, q and k_i may have the same structure, and thus the same size, although that is not always necessary. However, in some architectures q can consist of a sequence of vectors or a matrix, a possibility we explore in Section 6.4.4.

Some common compatibility functions are listed in Table 6.3. Two main approaches can be identified. A first one is to match and compare K and q. For instance, the idea behind the *similarity* attention proposed by Graves et al. [95] is that the most relevant keys are the most

Table 6.3: Summary of compatibility functions found in literature. W, W_0, W_1, ..., and b are learnable parameters. L is a fixed parameter.

Name	**Equation**	**Reference**
similarity	$f(q,K) = sim(q,K)$	Graves et al. [95]
multiplicative or *dot*	$f(q,K) = q^\intercal K$	Luong et al. [169]
scaled multiplicative	$f(q,K) = \frac{q^\intercal K}{\sqrt{n_k}}$	Vaswani et al. [263]
general or *bilinear*	$f(q,K) = q^\intercal W K$	Luong et al. [169]
biased general	$f(q,K) = K^\intercal(Wq+b)$	Sordoni et al. [242]
activated general	$f(q,K) = act(q^\intercal W K + b)$	Ma et al. [170]
concat	$f(q,K) = w_{imp}{}^\intercal act\big(W[K;q]+b\big)$	Luong et al. [169]
additive	$f(q,K) = w_{imp}{}^\intercal act(W_1 K + W_2 q + b)$	Bahdanau et al. [7]
deep	$f(q,K) = w_{imp}{}^\intercal E^{(L-1)} + b^L$ $E^{(l)} = act\Big(W_l E^{(l-1)} + b^l\Big)$ $E^{(1)} = act\Big(W_1 K + W_0 q + b^1\Big)$	Pavlopoulos et al. [200]
convolution-based	$f(q,K) = [e_0;\ldots;e_{d_k}]$ $e_j = \frac{1}{l}\sum_{i=j-l}^{j} e_{j,i}$ $e_{j,i} = act\Big(w_{imp}{}^\intercal[k_i;\ldots;k_{i+l}]+b\Big)$	Du et al. [62]
location-based	$f(q,K) = f(q)$	Luong et al. [169]

similar to the query. These approaches are particularly suitable in tasks where the concept of relevance of a key is known to be closely related to that of similarity to a query element. These include, for instance, tasks where specific keywords can be used as query, such as abusive speech recognition and sentiment analysis.

A different approach amounts to combining rather than comparing K and q, using them together to compute a joint representation, which is then multiplied by an *importance vector*[3] w_{imp}, which has to adhere to the same semantic of the new representation. Such a vector defines, in a way, relevance, and could be an additional query element, as offered by Ma et al. [172], or a learnable parameter. In that case, we speculate that the analysis of a machine-learned

[3]Our terminology. As previously noted, w_{imp} is termed *context vector* by Yang et al. [282] and other authors.

importance vector could provide additional information on the model. One of the simplest models that follow this approach is the *concat* attention by Luong et al. [169], where a joint representation is given by juxtaposing keys and query. These approaches are especially suitable when a representation of “relevant" elements is unavailable, or it is available but encoded in a significantly different way from the way keys are encoded. That may be the case, for instance, with tasks such as document classification and summarization.

6.4.3 Distribution functions

Attention distribution maps energy scores to attention weights. The choice of the distribution function depends on the properties the distribution is required to have—for instance, whether it is required to be a probability distribution, a set of probability scores, or a set of Boolean scores—on the need to enforce sparsity, and on the need to account for the keys' positions.

One possible distribution function g is the logistic sigmoid, as proposed by Kim et al. [129]. In this way, weights can thus be interpreted as probabilities that an element is relevant. The most common approach is to use a softmax function, in what we refer to as *soft attention*. Each attention weight can then be interpreted as the probability that the corresponding element is the most relevant.

It can be argued that, in some cases, some parts of the input are completely irrelevant, and if they were to be considered, they would likely introduce noise rather than contribute with useful information. In such cases, one could exploit attention distributions that altogether ignore some of the keys, thereby reducing the computational footprint. One option is the *sparsemax* distribution [177], which truncates to zero the scores under a certain threshold by exploiting the geometric properties of the probability simplex. This approach could be especially useful in those settings where a large number of elements is irrelevant, such as in document summarization or cloze question-answering tasks. In some tasks, such as machine translation, or image captioning, the relevant features are found in a neighborhood of a certain position. In those cases it could be helpful to focus the attention only on a specific portion of the input. If the position is known in advance, one can apply a positional mask [231]. Since the location may not be known in advance, other approaches [169, 278] considers the keys in a dynamically determined location. Finally, the concept of locality can also be defined according to semantic rules, rather than the temporal position. This possibility will be further discussed in Section 6.5.

6.4.4 Multiplicity

We shall now present variations of the general unified model where the attention mechanism is extended to accommodate multiple, possibly heterogeneous, inputs or outputs.

Multiple Outputs

Some applications suggest that the data could, and should, be interpreted in multiple ways. This can be the case when there is ambiguity in the data, stemming, for example, from words having multiple meanings, or when addressing a multi-task problem. For this reason, models have been defined that jointly compute not only one, but multiple attention distributions over the same data. One possibility presented by Lin et al. [153], and also exploited by Du et al. [63], is to use additive attention (seen in Section 6.4.2) with an *importance matrix* $W_{imp} \in \mathbb{R}^{n_k \times n_o}$, instead of a vector. In *multi-dimensional attention* [231], where the importance matrix is a square matrix, attention can computed feature-wise. Convolution-based attention [62] always produces multiple energy scores distributions according to the number of convolutional filters and the size of those filters. Another possibility explored by [263] is *multi-head* attention. There, multiple linear projections of all the inputs (K, V, q) are performed according to learnable parameters, and multiple attention functions are computed in parallel. Usually, the processed context vectors are then merged together into a single embedding. Finally, *label-wise* attention [10] computes a separate attention distribution for each class. This may improve the performance as well as lead to a better interpretation of the data, because it could help isolate data points that better describe each class. These techniques are not mutually exclusive. For example, multi-head and multi-dimensional attention can be combined with one another [35].

Multiple Inputs: Co-Attention

Some architectures consider the query to be a sequence of d_q multi-dimensional elements, represented by a matrix $Q \in \mathbb{R}^{n_q \times d_q}$, rather than by a plain vector. Examples of this set-up are common in architectures designed for tasks where the query is a sentence, as in question answering, or a set of keywords, as in abusive speech detection. In those cases, it could be useful to find the most relevant query elements according to the task and the keys. A straightforward way of doing that would be to apply the attention mechanism to the query elements, thus treating Q as keys and each k_i as query, yielding two independent representations for K and Q. However, in that way we would miss the information contained in the interactions between elements of K and Q. Alternatively, one could apply attention *jointly* on K and Q, which become the "inputs" of a *co-attention* architecture [165]. Co-attention models can be *coarse-grained* or *fine-grained* [72]. Coarse-grained models compute attention on each input,

using an embedding of the other input as a query. Fine-grained models consider each element of an input with respect to each element of the other input. Furthermore, co-attention can be performed *sequentially* or *in parallel*. In parallel models, the procedures to compute attention on K and on Q symmetric, thus the two inputs are treated identically.

Coarse-grained co-attention Coarse-grained models use a compact representation of one input to compute attention on the other. In such models, the role of the inputs as keys and queries is no longer focal, thus a compact representation of K may play as query in parts of the architecture and vice versa. A sequential coarse-grained model proposed by Lu et al. [165] is *alternating* co-attention, illustrated in Figure 6.6 (left), whereby attention is computed three times to obtain embeddings for K and Q. First, self-attention is computed on Q. The resulting context vector is then used as a query to perform attention on K. The result is another context vector C_K, which is further used as a query as attention is again applied to Q. This produces a final context vector, C_Q. The architecture proposed by Sordoni et al. [242] can also be described using this model with a few adaptations. In particular, Sordoni et al. omit the last step, and factor in an additional query element q in the first two attention steps. An almost identical sequential architecture is used by Zhang et al. [295], who use q only in the first attention step. A parallel coarse-grained model is illustrated in Figure 6.6 (right). In such a model, proposed by Ma et al. [170], an average (*avg*) is initially computed on each input, and then used as query in the application of attention to generate the embedding of the other input. Sequential co-attention offers a more elaborate computation of the final results, potentially allowing to discard all the irrelevant elements of Q and K, at a the cost of a greater computational footprint. Parallel co-attention can be optimized for better performance, at the expense of a "simpler" elaboration of the outputs. It is worthwhile noticing that the averaging step in Ma et al.'s model could be replaced by self-attention, in order to filter out irrelevant elements from Q and K at an early stage.

Fine-grained co-attention In fine-grained co-attention models, the relevance (energy scores) associated with each key/query element pair $\langle k_i/q_j \rangle$ is represented by the elements $E_{j,i}$ of a *co-attention matrix* $E \in \mathbb{R}^{d_q \times d_k}$ computed by a *co-compatibility* function. Co-compatibility functions can be straightforward adaptations of any of the compatibility functions listed in Table 6.3. Alternatively, new functions can be defined. Because $E_{j,i}$ represent energy scores associated with $\langle k_i/q_j \rangle$ pairs, computing the relevance of K with respect to specific query elements, or, similarly, the relevance of Q with respect to specific key elements, requires extracting information from E using what we call an *aggregation function*. The output of such

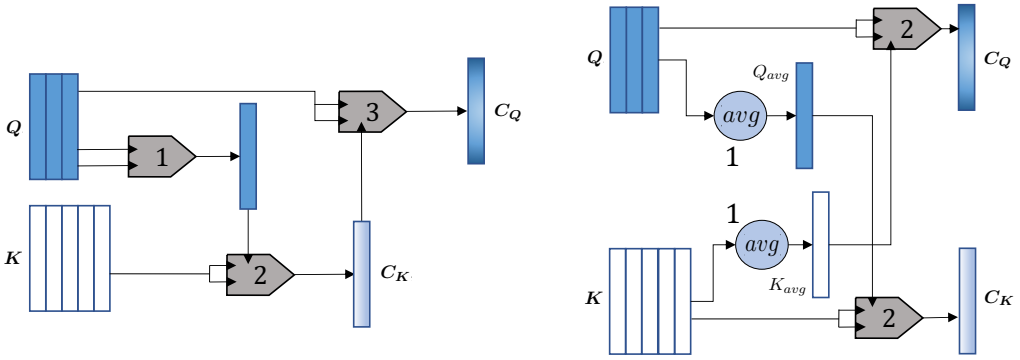


Figure 6.6: Coarse-grained co-attention by Lu et al. [165] (left) and Ma et al. [170] (right). The number inside the attention shapes (and besides the average operator) indicate the order in which they are applied. Different colors highlight different inputs.

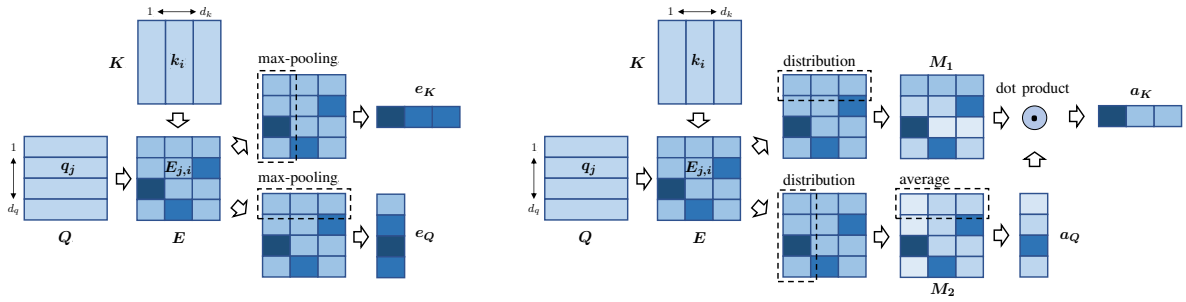


Figure 6.7: Fine-grained co-attention models presented by dos Santos et al. [61] (left) and by Cui et al. [47] (right). Dashed lines show how max pooling/distribution functions are applied (column-wise or row-wise).

a function is a pair a_K/a_Q of weight vectors. The commonest aggregation functions are listed in Table 6.4 and two example of fine-grained co-attention are illustrated in Figure 6.7.

Further improvements may be obtained by combining the results of multiple co-attention models. Fan et al. [72], for instance, compute coarse-grained and fine-grained attention in parallel, and combine the results into a single embedding.

6.5 Combining Attention and Knowledge

According to LeCun et al. [142], a major open challenge in AI is combining connectionist (or sub-symbolic) models, such as deep networks, with approaches based on symbolic knowledge representation, in order to perform complex reasoning tasks. Throughout the last decade, filling the gap between these two families of AI methodologies has represented a major research avenue. Popular approaches include statistical relational learning [88], neural-symbolic

Table 6.4: Aggregation functions. In most cases, a_K and a_Q are obtained by applying a distribution function, such as those seen in Section 6.4.3, to e_K and e_Q, and are thus omitted from this table in the interest of brevity. As customary, *act* is a placeholder for a generic non-linear activation function, whereas *dist* indicates a distribution function such as softmax.

Name	Equations	Reference
pooling	$e_{Ki} = \max_{1 \le j \le d_q} (E_{j,i})$ $e_{Qj} = \max_{1 \le i \le d_k} (E_{j,i})$	dos Santos et al. [61]
perceptron	$e_K = W_3{}^\intercal act(W_1 K + W_2 QE)$ $e_Q = W_4{}^\intercal act(W_2 K + W_1 QE^\intercal)$	Lu et al. [165]
linear transformation	$e_K = W_1 E$ $e_Q = W_2 E$	Li et al. [150]
attention over attention	$a_K = M_1 \cdot a_Q$ $a_Q = \underset{1 \le i \le d_k}{average}(M_{2,i})$ $M_{2,i} = \underset{1 \le j \le d_q}{dist} (E_{j,i})$ $M_{1,j} = \underset{1 \le i \le d_k}{dist} (E_{j,i})$	Cui et al. [47]
perceptron with nested attention	$e_K = W_3{}^\intercal act\big(W_1 K + (W_2 Q^\intercal) M_2\big)$ $e_Q = W_4{}^\intercal act\big(W_2 Q + (W_1 K^\intercal) M_1{}^\intercal\big)$ $M_{2,i} = \underset{1 \le j \le d_q}{dist} (E_{j,i})$ $M_{1,j} = \underset{1 \le i \le d_k}{dist} (E_{j,i})$	Nie et al. [195]

learning [82], and the application of various deep learning architectures [155] such as memory networks [251], neural Turing machines [95], and several others.

From this perspective, attention can be seen both as an attempt to improve the interpretability of neural networks, and as an opportunity to plug external knowledge into them. As a matter of fact, since the weights assigned by attention represent the relevance of the input with respect to the given task, in some contexts it could be possible to exploit this information to isolate the most significant features that allow the deep network to make its predictions. On the other hand, any prior knowledge regarding the data, the domain, or the specific task, whenever available, could be exploited to generate information about the desired attention distribution, which could be encoded within the neural architecture.

In this section, we overview different techniques that can be used to inject this kind of knowledge in a neural network. We leave to Section 6.6 further discussions on the open challenges regarding the combination of knowledge and attention.

6.5.1 Supervised Attention

In most of the works we surveyed, the attention model is trained with the rest of the neural architecture to perform a specific task. Although trained alongside a supervised procedure, the attention model *per se* is trained in an unsupervised fashion[4] to select useful information for the rest of the architecture. Nevertheless, in some cases knowledge about the desired weight distribution could be available. Whether it is present in the data as a label, or it is obtained as additional information through external tools, it can be exploited to perform a supervised training of the attention model.

Preliminary training

One possibility is to use an external classifier. The weights learned by such a classifier are subsequently plugged into the attention model of a different architecture. We name this procedure as preliminary training. For example, Zhang et al. [296] first train an attention model to represent the probability that a sentence contains relevant information. The relevance of a sentence is given by *rationales* [289], which are snippets of text that support the corresponding document categorizations. In work by Long et al. [164], a model is preliminarily trained on eye-tracking datasets to estimate the reading time of words. Then, the reading time predicted by the model is used as an energy score in a neural model for sentiment analysis.

Auxiliary training

Another possibility is to train the attention model without preliminary training, but by treating attention learning as an auxiliary task that is performed jointly with the main task. This procedure has led to good results in many scenarios, including machine translation [160, 184], visual question answering [209], and domain classification for natural language understanding [128].

In some cases, this mechanism can be exploited also to have attention model specific features. For example, since the linguistic information is useful for semantic role labelling, attention can be trained in a multi-task setting to represent the syntactic structure of a sentence. Indeed, in LISA [249], a multi-layer multi-headed architecture for semantic role labelling, one of the attention heads is trained to perform dependency parsing as an auxiliary task.

[4]Meaning that there is no target distribution for the attention model.

Transfer learning

Furthermore, it is possible to perform transfer learning across different domains [8] or tasks [286]. By performing a preliminary training of an attentive architecture on a source domain to perform a source task, a mapping between the inputs and the distribution of weights will be learned. Then, when another attentive architecture is trained on the target domain to perform the target task, the pre-trained model can be exploited. Indeed, the desired distribution can be obtained through the first architecture. Attention learning can therefore be treated as an auxiliary task as in the previously mentioned cases. The difference is that the distribution of the pre-trained model is used as ground truth, instead of using data labels.

6.5.2 Attention tracking

When attention is applied multiple times on the same data, as in sequence-to-sequence models, a useful piece of information could be how much relevance has been given to the input along different model iterations. Indeed, one may need to keep track of the weights that the attention model assigns to each input. For example, in machine translation it is desirable to ensure that all the words of the original phrase are taken into account. One possibility to maintain this information is to use a suitable structure and provide it as an additional input to the attention model. Tu et al. [260] exploit a piece of symbolic information called *coverage* to keep track of the weight associated to the inputs. Every time attention is computed, such information is fed to the attention model as a query element, and it is updated according to the output of the attention itself. In Mi et al. [183], the representation is enhanced by making use of a sub-symbolic representation for the coverage.

6.5.3 Modelling the distribution function by exploiting prior knowledge

Another component of the attention model where prior knowledge can be exploited is the distribution function. For example, constraints can be applied on the computation of the new weights to enforce the boundaries on the weights assigned to the inputs. In Malaviya et al. [175], Martins and Kreutzer [178], the coverage information is exploited by a *constrained* distribution function, regulating the amount of attention that the same word receives over time.

Prior knowledge could also be exploited also to define or to infer a distance between the elements in the domain. Such domain-specific distance could then be considered in any position-based distribution function, instead of the positional distance. An example of distance could be derived by the syntactical information. Chen et al. [34], He et al. [110] use distribution

functions that takes into account the distance between two words along the dependency graph of a sentence.

6.6 Challenges and Future Directions

In this section, we discuss open challenges and possible applications of the attention mechanism in the analysis of neural networks, as a support of the training process, and as an enabling tool for the integration of symbolic representations within neural architectures.

6.6.1 Attention for deep networks investigation

Whether attention may or may not be considered as a mean to explain neural networks is currently an open debate. Some recent studies [124, 230] suggest that attention cannot be considered a reliable mean to explain or even interpret neural networks. Nonetheless, other works [40, 147, 262, 275] advocate the use of attention weights as an analytic tool. Specifically, Jain and Wallace [124] have proved that attention is not *consistent* with other explainability metrics and that it is easy to create local adversarial distributions (distributions which are similar to the trained model but produce a different outcome). Wiegreffe and Pinter [275] have pushed the discussion further, providing experiments that demonstrate that creating an effective global adversarial attention models is much more difficult than creating a local one, and that attention weights may contain information regarding feature importance. Their conclusion is that attention may indeed provide an explanation of a model, if by explanation we mean a *plausible*, but not necessarily *faithful*, reconstruction of the decision-making process, as suggested by Riedl [218], Rudin [222].

In the context of a multi-layer neural architecture it is fair to assume that the deepest levels will compute the most abstract features [141, 142]. Therefore, the application of attention to deep networks could enable the selection of higher-level features, thus providing hints to understand which complex features are relevant for a given task. Following this line of inquiry in the computer vision domain, Zhang et al. [291] showed that the application of attention to middle-to-high level feature-sets leads to better performance in image generation. The visualization of the self-attention weights has revealed that higher weights are not attributed to proximate image regions, but rather to those regions whose color or texture is most similar to that of the query image point. Moreover, the spatial distribution does not follow a specific pattern, but instead it changes, modelling a region that corresponds to the object depicted in the image. Identifying abstract features in an input text might be less immediate than in an image, where the analysis process is greatly aided by visual intuition. Yet, it may be interesting

to test the effects of the application of attention at different levels, and to assess whether its weights correspond to specific high-level features. For example, Vaswani et al. [263] analyze the possible relation between attention weights and syntactic predictions, Voita et al. [266] do the same with anaphora resolutions, and Clark et al. [40] investigate the correlation with many linguistic features. Voita et al. [267] analyze the behaviour of the heads of a multi-head model, discovering that different heads develop different behaviours, which can be related to specific position or to specific syntactical element. Yang et al. [279] seem to confirm that the deeper levels of neural architectures capture non-local aspects of the textual input. They studied the application of locality at different depths of an attentive deep architecture, and showed that its introduction is especially beneficial when it is applied to the layers that are closer to the inputs. Moreover, when the application of locality is based on a variable-size window, higher layers tend to have a broader window.

A popular way of investigating whether an architecture has learned high-level features amounts to using the same architecture to perform other tasks, as it happens with transfer learning. This setting has been adopted outside the context of attention, for example by Shi et al. [233], who perform syntactic predictions by using the hidden representations learned with machine translation. In a similar way, attention weights could be used as input features in a different model, so as to assess whether they can select relevant information for a different learning task. This is what happens, for example, in *attention distillation*, where a student network is trained to penalize the most confusing features according to a teacher network, producing an efficient and robust model in the task of machine reading comprehension [115]. Similarly, in a transfer learning setting, *attentional heterogeneous transfer* [190] has been exploited in hetero-lingual text classification to selectively filter input features from heterogeneous sources.

6.6.2 Attention for outlier detection and sample weighing

Another possible use of attention may be for outlier detection. In tasks such as classification, or the creation of a representative embedding of a specific class, attention could be applied over all the samples belonging to that task. In doing so, the samples associated with small weights could be regarded as outliers with respect to their class. The same principle could be potentially applied to each data point in a training set, independently of its class. The computation of a weight for each sample could be interpreted as assessing the relevance of that specific data point for a specific task. In principle, assigning such samples a low weight and excluding them from the learning could improve a model's robustness to noisy input. Moreover, a dynamic computation of these weights during training would result in a dynamic selection of different training data in different training phases. While attention-less adaptive data selection strategies

have already proven to be useful for efficiently obtaining more effective models [73], to the best of our knowledge no attention-based approach has been experimented to date.

6.6.3 Attention analysis for model evaluation

The impact of attention is greatest when all the irrelevant elements are excluded from the input sequence, and the importance of the relevant elements is properly balanced. A seemingly uniform distribution of the attention weights could be interpreted as a sign that the attention model has been unable to identify the more useful elements. That in turn may be due to the data not contain useful information for the task at hand, or it may be ascribed to the poor ability of the model to discriminate information. Either way, the attention model would be unable to find relevant information in the specific input sequence, which may lead to errors. The analysis of the distribution of the attention weights may therefore be a tool for measuring an architecture's confidence in performing a task on a given input. We speculate that a high entropy in the distribution or the presence of weights above a certain threshold may be correlated to a higher probability of success of the neural model. These may therefore be used as indicators, to assess the uncertainty of the architecture, as well as to improve its interpretability. Clearly, this information would be useful to the user, to better understand the model and the data, but it may also be exploited by more complex systems. For example, Heo et al. [111] propose to exploit the uncertainty of their stochastic predictive model to avoid making risky predictions in healthcare tasks.

In the context of an architecture that relies on multiple strategies to perform its task, such as a hybrid model that relies on both symbolic and sub-symbolic information, the uncertainty of the neural model can be used as parameter in the merging strategy. Other contexts in which this information may be relevant are multi-task learning and reinforcement learning. Examples of exploitation of the uncertainty of the model, although in contexts other than attention and NLP, can be found in works by Blundell et al. [21], Kendall et al. [127], Poggi and Mattoccia [205].

6.6.4 Unsupervised learning with attention

To properly exploit unsupervised learning is widely recognized as one of the most important long-term challenges of AI [142]. As already mentioned in Section 6.5, attention is typically trained in a supervised architecture, although without a direct supervision on the attention weights. Nevertheless, a few works have recently attempted to exploit attention within purely unsupervised models. We believe this to be a promising research direction, as the learning process of humans is indeed largely unsupervised.

For example, in work by He et al. [109], attention is exploited in a model for aspect extraction in sentiment analysis, with the aim to remove words that are irrelevant for the sentiment, and to ensure more coherence of the predicted aspects. In work by Zhang and Wu [294], attention is used within autoencoders in a question-retrieval task. The main idea is to generate semantic representations of questions, and self-attention is exploited during the encoding and decoding phase, with the objective to reconstruct the input sequences, as in traditional autoencoders. Following a similar idea, Zhang et al. [290] exploit bidimensional attention-based recursive autoencoders for bilingual phrase embeddings, whereas Tian and Fang [256] exploit attentive autoencoders to build sentence representations and perform topic modeling on short texts. Yang et al. [281] instead adopt an attention-driven approach to unsupervised sentiment modification in order to explicitly separate sentiment words from content words.

In computer vision, *attention alignment* has been proposed for unsupervised domain adaptation, with the aim to align the attention patterns of networks trained on the source and target domain, respectively [126]. We believe this could be an interesting scenario also for NLP.

6.6.5 Neural-symbolic learning and reasoning

Recently, attention mechanisms started to be integrated within some neural-symbolic models, whose application to NLP scenarios is still at an early stage. For instance, in the context of *neural logic programming* [280], they have been exploited for reasoning over knowledge graphs, in order to combine parameter and structure learning of first-order logic rules. They have also been used in logic attention networks [272] to aggregate information coming from graph neighbors with both rules- and network-based attention weights. Moreover, prior knowledge has also been exploited by Shen et al. [232] to enable the attention mechanism to learn the knowledge representation of entities for ranking question-answer pairs.

Neural architectures exploiting attention performed well also in reasoning tasks that are also addressed with symbolic approaches, such as textual entailment [221]. For instance, Hudson and Manning [121] recently proposed a new architecture for complex reasoning problems, with NLP usually being one of the target sub-tasks, as in the case of visual question answering. In such an architecture, attention is used within several parts of the model, for example over question words, or to capture long-range dependencies with self-attention.

An attempt to introduce constraints in the form of logical statements within neural networks has been proposed in Li and Srikumar [149] where rules governing attention are used to enforce word alignment in tasks such as machine comprehension and natural language inference.

6.7 Conclusion

Attention models have become ubiquitous in NLP applications. Attention can be applied to different input parts, different representations of the same data, or different features, to obtain a compact representation of the data as well as to highlight relevant information. The selection is performed through a distribution function, which may take into account locality in different dimensions, such as space, time, or even semantics. Attention can be used to compare the input data with a query element based on measures of similarity or significance. It can also autonomously learn what is to be considered relevant, by creating a representation encoding what the important data should be similar to. Integrating attention in neural architectures may thus yield a significant performance gain. Moreover, attention can be used as a tool for investigating the behaviour of the network.

In this Chapter, we have introduced a taxonomy of attention models, which enabled us to systematically chart a vast portion of the approaches in literature and compare them one another. To the best to our knowledge, this is the first systematic, comprehensive taxonomy of attention models for NLP.

We have also discussed the possible role of attention in addressing fundamental AI challenges. In particular, we have shown how attention can be instrumental in injecting knowledge into the neural model, so as to represent specific features, or to exploit previously acquired knowledge, as in transfer learning settings. We speculate that this could pave the way to new challenging research avenues, where attention could be exploited to enforce the combination of sub-symbolic models with symbolic knowledge representations to perform reasoning tasks, or to address natural language understanding. Recent results also suggest that attention could be a key ingredient of unsupervised learning architectures, by guiding and focusing the training process where no supervision is given in advance.

In this dissertation, we will use neural attention in order to improve neural models for the task of AM. Its full integration in neural-symbolic frameworks for AM is left for future work, but it is a line of research we are eager to pursue.

Chapter 7

Feature-Agnostic Attention-based Deep ResNets for AM

We explore the use of residual networks and neural attention for argument mining mining, with an emphasis on link prediction. The method we propose makes no assumptions on document or argument structure. We define and experiment two different residual architectures (with and without attention), making use of ensemble learning. We evaluate them firstly on a challenging dataset consisting of user-generated comments collected from an online platform, and then on two other corpora made out of scientific publications. Results on the first corpus show that our models outperforms state-of-the-art methods that rely on domain knowledge. The experimentation on the other corpora confirm the validity of our method, which is comparable to BERT-based approaches. This Chapter builds on and extends what has been published in Galassi et al. [77], by proposing a new attention-based architecture and discussing new experimental results on 4 datasets. Part of this new content has been reported in Galassi et al. [80]. All the code used in our experiments is publicly available.[1]

7.1 Introduction

In spite of significant results achieved in component identification tasks, such as claim/evidence detection [157, 196, 199, 220, 246], classification [68, 194] and boundary detection [103, 148, 158, 224], comparatively less progress has been made in the arguably more challenging argument structure prediction task [26, 246], which is known to be more complex and nuanced [140].

[1] https://github.com/AGalassi/StructurePrediction18.

Again due to the challenging nature of the general argument mining problem, solutions have typically addressed a specific genre or application domain, such as legal texts [189], persuasive essays [247], or Wikipedia articles [148, 220] and have heavily relied on domain knowledge. One particular aspect of the domain is the argument model. While argumentation as a discipline has developed rather sophisticated argument models, such as Toulmin's [257], the majority of the available argument mining data sets refer to ad-hoc, usually simpler argument models, often in an effort to obtain a reasonable inter-annotator agreement. Another crucial aspect is the document structure. For instance, in some domains, certain argument components occupy a specific position in the document.

Moreover, until recently, approaches have mostly used traditional methods such as support vector machines, logistic regression, and naive Bayes classifiers. Only in the last years, the field has started to look more systematically into neural network-based architectures, such as long short-memory networks and convolutional neural networks, and structured output classifiers.

The aim of this study is to investigate the application of residual networks and neural attention to a challenging structure prediction task, namely link prediction. Similarly to Cocarascu and Toni [41], our ambition is to define a model that does not exploit domain-specific, highly engineered features or information on the underlying argument model, and could thus be, at least in principle, of general applicability. To evaluate the limits of applicability of our model, we test it on four different corpora, obtaining encouraging results.

The next section presents our models, Section 7.3 our methods, and Section 7.4 the results. Section 7.5 sums up our findings.

7.2 Residual Networks for AM

The architecture we propose makes use of the dense residual network model, along with an LSTM [112], and an attention module [76] to jointly perform link prediction and argument component classification. More specifically, our approach works at a local level on pairs of sentences, without any document-level global optimization, and without imposing model constraints induced, e.g., by domain-specific or genre-specific hypotheses. For that reason, it lends itself to the integration with more complex systems.

7.2.1 Model description

One of our aims is to propose a method that abstracts away from a specific argument model or domain. We thus reason in terms of abstract entities, such as argumentative propositions and the links among them. Such abstract entities are instantiated into concrete categories, such

as claims and premises, supports and attacks, as soon as we apply the method to a domain described by a specific dataset whose annotations follow a concrete argument model. According to the corpus on which we are working, we instantiate our model with the categories used as annotations.

In general, a document D is a sequence of *tokens*, i.e., words and punctuation marks. An argumentative proposition a is a sequence of contiguous tokens within D, which represents an argument, or part thereof. We do not perform components detection on the document, instead we consider argumentative propositions as already available and perfectly bounded. Labeling of propositions is induced by the chosen argument model. Such a labeling associates each proposition with the corresponding category P of the argument component it contains. Since we do not make a distinction between the concept of argumentative component and proposition containing one, we will use them as equivalent.

Given two argumentative propositions a and b belonging to the same document, a directed relation from the former (*source*) to the latter (*target*) is represented as $a \rightarrow b$. Reflexive relations ($a \rightarrow a$) are not allowed.[2]

Each relation $a \rightarrow b$ is characterized by two labels: a (Boolean) *link label*, $L_{a \rightarrow b}$, and a *relation label*, $R_{a \rightarrow b}$. The link label indicates the presence of a link, and is therefore *true* if there exists a directed link from a to b, and *false* otherwise. The relation label instead contains information on the nature of the link connecting a and b. In particular, it represents the direct or inverse relationship between the two propositions, according to the links that connect a to b or b to a. In other words, its domain is composed, according to the underlying argument model, not only by all the possible link types (e.g., *attack* and *support*) but also by their opposite types (e.g., *attackedBy* and *supportedBy*) as well as by a category, *no-rel*, meaning the absence of link in either direction. The choice to include opposite types of relations is motivated by the minute amount of instances that do have a relation. We speculate that the reduction of instances belonging to the no-rel class, introducing additional labels, may contribute positively to the optimization process. During the evaluation process, for the sake of simplicity and consistency with other works, these labels will be considered as one and only with the *no-rel* label.

One objective is to establish the value of the link label $L_{a \rightarrow b}$ for each possible input pair of propositions (a, b) belonging to the same document D. Such a *link prediction* task can be considered as a sub-task of argument structure prediction. Another objective is the *classification* of propositions and relations, i.e., the prediction of labels P_a, P_b, $R_{a \rightarrow b}$. All these 4 taks are performed jointly, in a multi-objective learning setting.[3]

[2]We will partially consider reflexive relations for the UKP dataset for a specific reason explained in Section 7.3.

[3]Since we examine only argumentative propositions, we do not consider the non-argumentative class for component classification.

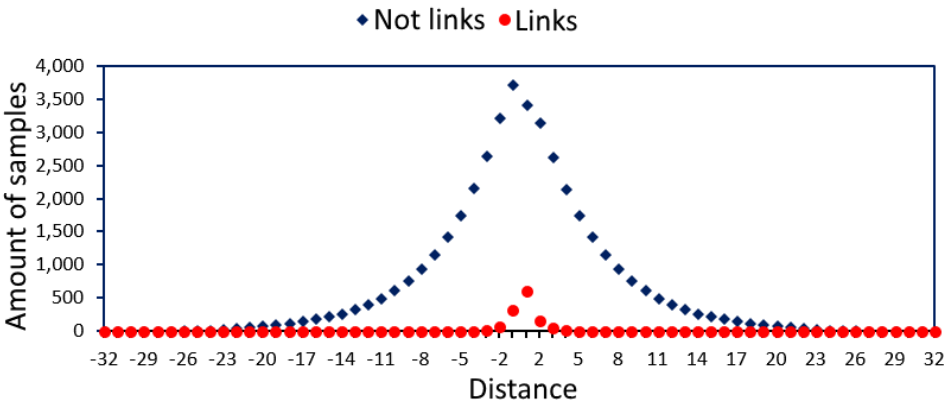


Figure 7.1: Link distribution in the CDCP dataset with respect to distance. The distance is considered positive when the source precedes the target, negative otherwise.

7.2.2 Embeddings and features

Since the purpose of this work is to evaluate deep residual networks and the attention module as instruments for AM, without resorting to any domain- or genre-specific information, the system relies on a minimal set of features that do not require elaborate processing.

Any input token is transformed into a 300-dimensional embedding by exploiting the GloVe pre-trained vocabulary [201]. Input sequences are zero-padded to the length of the longest sequence in the dataset, which we will address as T.

A preliminary analysis of the CDCP corpus suggests that the number of argumentative propositions separating source and target (which will be referred to as *argumentative distance*) could be a relevant feature since most linked propositions are not far from each other. Indeed, as Figure 7.1 shows, around 70% of links are between adjacent propositions. Such characteristic is not exclusive of this corpus and has been recently exploited also in other corpora [208]. We thus employed the argumentative distance as an additional feature. The same analysis applied to the other corpora has corroborated this hypothesis. Following the approach we have used in Chapters 2 and 3, we represented distance using as a 10-bit array, where the first 5 bits are used in the case that the source precedes the target, and the last 5 bits are used in the opposite case. In both cases, the number of consecutive "1" values encodes the value of the distance (distances are capped by 5).[4] In this way, the Hamming distance [105] between the encodings of two argumentative distances results equal to the absolute difference between their values.

[4]For example, if the target precedes the source by two sentences, the distance is -2, which produces encoding 0001100000; if the source precedes the target by three sentences, the distance is 3, with encoding 0000011100.

7.2.3 RESARG: Residual Network Architecture

We refer to the first network architecture we have designed as RESARG and we illustrate it in Figure 7.2a. It is composed of the following macro blocks:

- two deep embedders, one for sources and one for targets, that manipulate token embeddings;
- a dense encoding layer for feature dimensionality reduction;
- an LSTM to process the input sequences;
- a residual network;
- the final-stage classifiers.

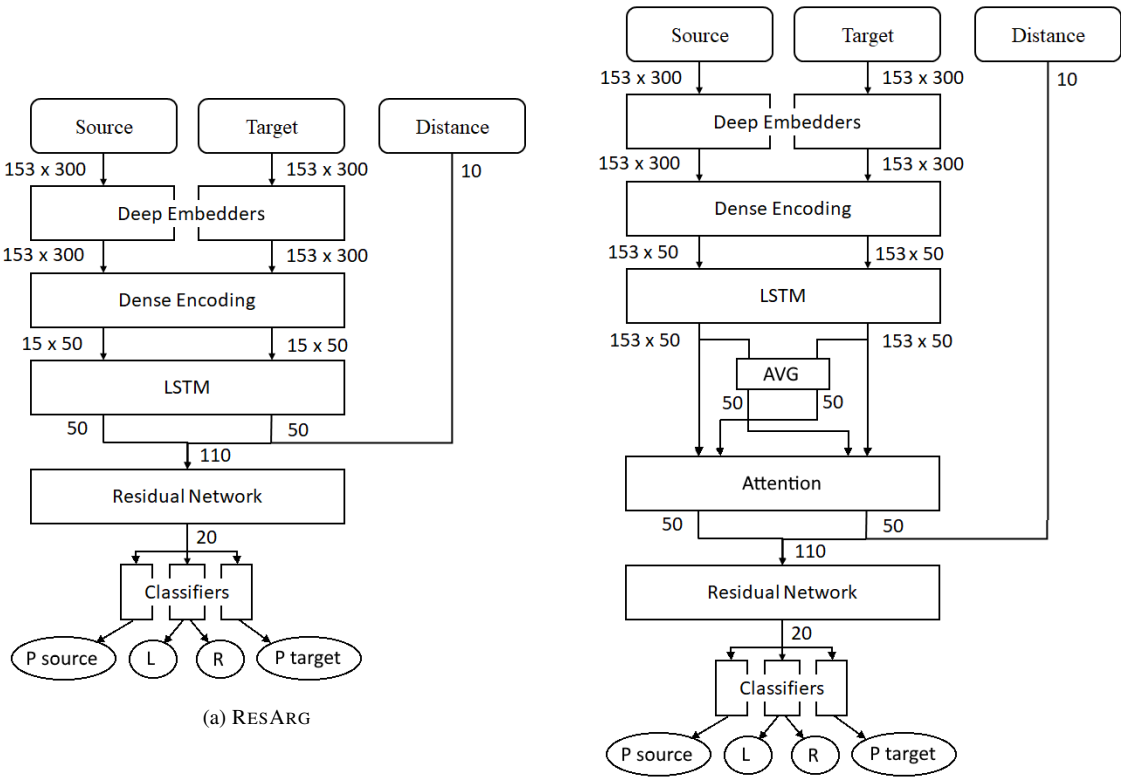
Source and target propositions are encoded separately by the first three blocks, then they are concatenated together, along with the distance, and given as input to the residual network.

The deep embedders refine the token embeddings, thus creating new, more data-specific embeddings. Relying on deep embedders instead of on pre-trained autoencoders, it aims to achieve a better generality, at least in principle, and avoid excessive specialization, thus limiting overfitting. The dimensionality reduction operated by the dense encoding layer allows the use of an LSTM with fewer parameters, which has two positive effects: it reduces the time needed for training, and again it limits overfitting.

The deep embedders are residual networks composed of a single residual block, composed of 4 pre-activated time-distributed dense layers. Accordingly, each layer applies the same transformation to each embedding, regardless of their position inside the sentence. All the layers have 50 neurons, except for the last one, which has 300 neurons.

The dense encoding layer reduces the size of the embedding sequences by applying a time-distributed dense layer, which reduces the embedding size to 50, and a time average-pooling layer [45], which reduces the sequence size to $1/10$ of the original. The resulting sequences are then given as input to the same bidirectional LSTM, producing a single representation of each proposition of size 50. Thus, for each proposition, T embeddings of size 300 are transformed first into T embeddings of size 50, then into $T/10$ embeddings of size 50, and finally in a single feature of size 50.

Source and target features, computed this way, alongside with the distance encoding, are then concatenated together and given as input to the residual network. The first level of the final residual network is a dense encoding layer with 20 neurons, while the residual block is composed of a layer with 5 neurons and one with 20 neurons. The sums of the results of the first and the last layers of the residual networks are provided as input to the classifiers.



(a) RESARG

(b) RESATTARG

Figure 7.2: A block diagram of the proposed architectures. The figure shows, next to each arrow, the dimensionality of the data involved (using the CDCP temporal size), so as to clarify the size of the inputs and the outputs of each block.

The final layers of the system are three independent softmax classifiers used to predict the source, the target, and the relation labels. The output of each classifier is a probability distribution among all the possible classes of that label. The predicted class is the one with the highest score. All these three classifiers, which predict labels for two different tasks, contribute simultaneously to our learning model. The link classifier is obtained by summing the relevant scores produced by the relation classifier.[5]

All the dense layers use the rectifier activation function [90], and they randomly initialize weights with He initialization [106]. The application of all non-linearity functions is preceded by batch-normalization layers [123] and by dropout layers [245], with probability $p = 0.1$.

[5]For instance, if our model considers *attack* and *support* relations as the only possible links, and the relation classifier scores are $attack = 0.15$, $support = 0.2$, $attackedBy = 0.1$, $supportedBy = 0.05$, $none = 0.5$, then the link classifier scores are: $true = 0.35$, $false = 0.65$.

7.2.4 RESATTARG: Attention-based Residual Network Architecture

Moved by the considerable results obtained by attention-based architectures in NLP tasks, we have improved the architecture previously described including also a neural attention block positioned after the bi-LSTM module, creating an architecture we have named RESATTARG. With the introduction of attention, we have decided to make minor changes to the rest of the architecture so as to provide more information to the new module. We have removed the time pooling layer from the dense encoding block, so as to avoid loss of information along the temporal axis, and maintained the whole output sequence from the LSTM. Therefore, in this new model, the input and the output of the LSTM module are of size $(T, 50)$. The resulting architecture is depicted in Figure 7.2b.

After a few preliminary experiments, we have decided to implement the attention module as coarse-grained co-attention, so as to consider both propositions at the same time while computing attention on each of them. Our method consists of exploiting the average embedding of one proposition as a query element while computing attention on the other, similarly to what has been done by Ma et al. [170] and we have described in Section 6.4.4. Specifically, calling K_s and K_t the outputs of the bi-LSTM obtained from, respectively, the processing of the source and the target propositions, we compute the average of K_t, obtaining a single embedding avg_t of size 50. This embedding is used as query element to compute attention on K_s (that act both as keys and values) obtaining a single source context vector c_s of size 50. An equivalent symmetric procedure is used to compute attention on K_t so as to obtain c_t. Since the final output of this module are two embeddings of size 50, as in our original architecture, the rest of the network is maintained the same.

7.3 Method

We evaluated our original model against CDCP, then we have extended the evaluation also to the other datasets. We tokenized documents using a hand-crafted parser based on the progressive splitting of the tokens and search within the GloVe vocabulary. We preferred not to use existing tools because of the nature of the data, to tackle the fact that user-generated content present in CDCP (and in other potential datasets) may not be well-formed. Out-of-vocabulary words are mapped into random embeddings.

Regarding comparability with other methods, it is important to underline that in this work we consider only argumentative units, therefore we perform component classification only between argumentative classes and we do not consider the "non-argumentative" class as a possibility. Since we perform component classification on propositions or sentences, to make our results comparable with architectures that perform it token-wise, we split each classified

component into tokens that share the same label, and compute the evaluation of token-wise classification. Since the tokenization method may not be the same one used by other approaches, the final results may not be perfectly comparable, but we believe that this minor difference will not introduce appreciable errors.

Because of the fact that each proposition is involved in many pairs, both as a source and as a target, its classification is performed multiple times by the same network. To classify it uniquely for evaluation purposes, we considered the average probability score assigned to each class and we label it as the most probable class. That is of course not the only option. Another possibility could be to assign the class that results to be the most probable in most of the cases, thus relying on a majority vote. A further option could be to simply consider the label with the highest confidence. However, this procedure might be more sensitive to outliers, because the misclassification of a sentence in just one pair would lead to the misclassification of the sentence, regardless of all the other pairs. A deeper analysis of different techniques to address this issues is left to future research.

7.3.1 Ensemble Learning

Since the training of neural models is non-deterministic, the results of a single training are influenced by the random seed that is used, thus they may not be reliable or reproducible [94, 214]. We have therefore decided to extend our initial analysis, repeating the training procedure 10 times, with different seeds, obtaining for each configuration 10 trained neural networks. We evaluate our models in two different ways. At first, we consider for each metric the average of the scores obtained by every single network. Then, we evaluate the predictions obtained using all the 10 models in ensemble voting. In our ensemble setting the class of each entity is assigned as the class voted by the majority of the networks. This technique is similar to the concept of bootstrap aggregating, also known as bagging [24]. While in proper bagging each model is trained on a stochastically selected sample of the training set, we train all the models on the same training set. The motivation is that stochastic elements are already present in the training procedure itself. We have chosen this ensemble method for sake of simplicity, but more advanced techniques do exist and may yield better results.

7.3.2 Approaches to other Corpora

UKP

Following previous works [194, 247], we consider exclusively pairs of components that belong to the same paragraph. This introduces a problem: many major claims (about 400) are the only

argumentative components of their paragraph, therefore it is not possible to include them in the classification. Therefore, we have decided to include reflexive pairs into our dataset, which are instances where the same component acts both as source and target. This greatly increases the number of pairs in the dataset (from 22,000 to 28,000). To improve optimization and to remain comparable with previous approaches, we do not consider these pairs for link prediction and relation classification during validation steps and in the final evaluation.

AbstRCT

Our approach is directly compatible with the AbstRCT corpus.

DrInventor

For what concerns DrInventor, we need to pre-process its data with a specific method since we face two problems: the length of the documents at hand and the split components.

First of all, due to the length of the documents, it becomes computationally impossible for our resources to consider all the possible couples of argumentative units. Moreover, this would lead to having an extremely unbalanced dataset for link prediction (less than 1% of pairs will have a link). We have therefore decided to include only the pairs that appear in the same section of the document and whose argumentative distance is included between -10 and +10.

To address the problem of components that are split into multiple parts, our approach has been the following: if two sequences of text $s1$ and $s3$ belong to the same component but are separated by another sequence $s2$, we consider both of them to be independent components. They will share the same label and share the same links so that they will have the same argumentative relation with any other components, and no relationship will exist between the two.

The dataset thus created consists of about 8,700 links out of 240,000 possible pairs, which amount roughly to 3.6%. Among these links, SUPPORTS relations amount to 89%, CONTRADICTS to 10%, and the remaining 1% are SEMANTICALLY SAME relations.

7.4 Results

We perform 4 stages of experimentation. Initially, we compare our first model against the structured learning setting of [194] on CDCP. In the second stage, we repeat the experiment training multiple models and evaluate their average performance and their behavior as ensemble. In the third stage, we introduce the attention-based model. In the last stage and we extend our evaluation to other datasets, testing our best model also on AbstRCT and Dr. Inventor.

Table 7.1: CDCP dataset composition.

Split	Train	Valid.	Test	Total
Documents	513	68	150	731
Propositions	3,338	468	973	4,779
Values	1,438	231	491	2,160
Policies	585	77	153	815
Testimonies	738	84	204	1,026
Facts	549	73	124	746
References	28	3	1	32
Couples	30,056	3,844	9,484	43,384
Links	923	143	272	1,338
Reasons	888	139	265	1,292
Evidences	35	4	7	46

We defined the learning problem as a multi-objective optimization problem, whose loss function is given by the weighted sum of four different components: the categorical cross-entropy on three labels (source and target categories, link relation category) and an L_2 regularization on the network parameters. The weights of these components were, respectively, 1, 1, 10, 10^{-4}.

We performed mini-batch optimization using Adam [130] with parameters $b_1 = 0.9$ and $b_2 = 0.9999$, and by applying proportional decay of the initial learning rate $\alpha_0 = 5 \times 10^{-3}$. The training was early-stopped after 200 epochs with no improvements on the validation data. For Dr. Inventor, due to the size of the dataset, we perform early-stopping using fewer epochs of patience. We chose the numerous hyper-parameters of the architecture and of the learning model after an initial experimental setup phase, based on the performance on the validation set for the link prediction task. Results obtained in this phase confirmed that the presence of the deep embedder block and of the distance feature leads to better results.

7.4.1 Residual Networks vs Structured Learning

We created a validation set by randomly selecting documents from the original training split with 10% probability. We used the remaining documents as training data and the original test split as is. Table 7.1 reports the statistics related to the three splits.

We compared the results of the RESARG model against an equivalent deep network with the same number of layers and the same hyper-parameters, but without the shortcut that characterizes the residual network block. We applied two different training procedures for both this deep network baseline and the residual network. In particular, as the criterion for early stopping, we used once the error on link prediction and once the error on proposition

Table 7.2: Results of the use of baselines, RESARG, and the structured approaches on CDCP. (Macro) F_1 percentage scores are reported. For each class, the number of instances is reported in parenthesis. For the comparison with structured learning, the best scores obtained by any of the structured configurations are reported.

Approach	Baseline		RESARG		Structured	
	LG	PG	LG	PG	SVM	RNN
Average (Link and Proposition)	33.18	42.88	47.28	46.37	**50.0**	43.5
Link (272)	22.56	22.45	**29.29**	20.76	26.7	14.6
Proposition (973)	43.79	63.31	65.28	71.99	**73.5**	72.7
VALUE (491)	73.77	74.45	72.19	73.24	76.4	73.7
POLICY (153)	73.85	76.09	74.36	76.43	77.3	76.8
TESTIMONY (204)	71.36	65.98	72.86	68.63	71.7	75.8
FACT (124)	0	0	40.31	41.64	42.5	42.2
REFERENCE (1)	0	100	66.67	100	100	100
Relation (272)	11.68	11.52	**15.01**	10.31		
REASON (265)	23.35	23.04	30.02	20.62		
EVIDENCE (7)	0	0	0	0		

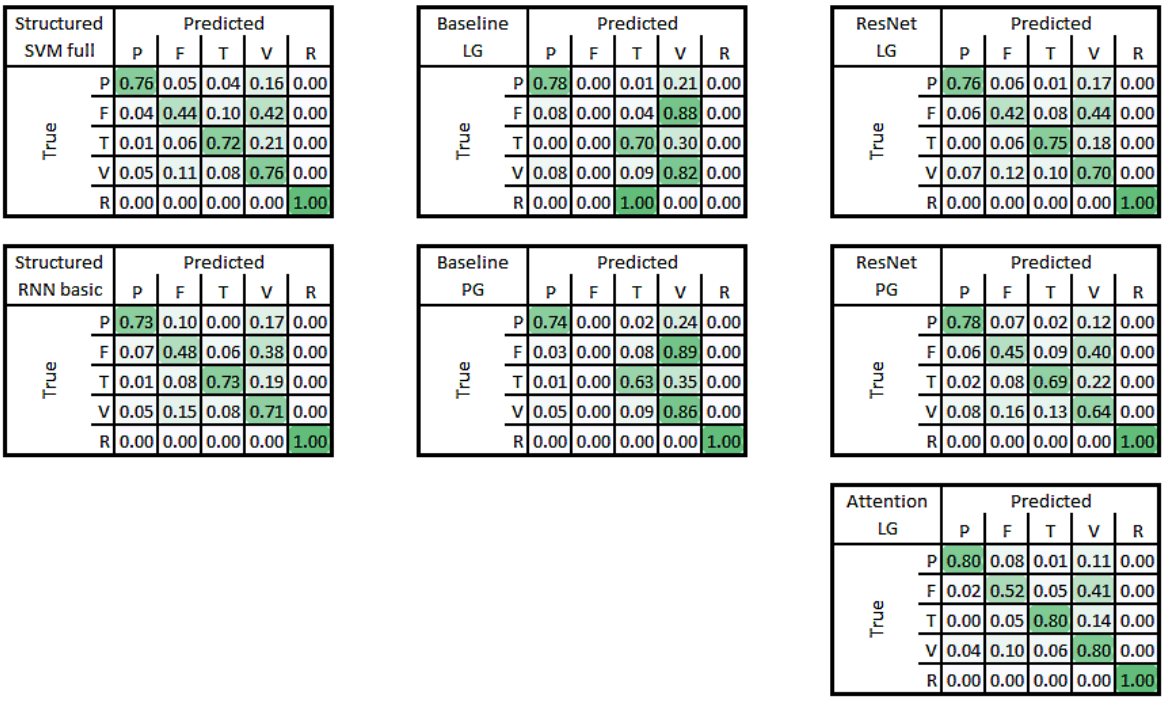
classification. In the presentation of our results, we will refer to these two models as link-guided (LG) and proposition-guided (PG).

Following Niculae et al. [194], we measured the performance of the models by computing the F_1 score for links, propositions, and the average between the two, in order to provide a summary evaluation. More specifically, for the links, we measured the F_1 of the positive classes (as the harmonic mean between precision and recall), whereas for the propositions we used the score of each class and then we computed the macro-average. We also reported the F_1 score for each direct relation class, alongside their macro-average. The "non-relation" class is considered in the classification, but it is excluded from the computation of the F_1 macro score.

Table 7.2 summarizes the evaluation of baselines and residual networks, also showing the best scores obtained by the structured learning configurations presented in [194].

Results highlight how the proposed approach based on residual networks outperforms the state of the art for what concerns link prediction. In addition, residual link-guided network training consistently performs better than both deep networks baselines in all three tasks.

As for proposition label prediction, the results obtained through structured approaches still maintain a slight advantage over residual networks. This could be partially explained by the fact that hyper-parameter tuning was done with the aim to select the best model for link prediction. It should also be considered that we perform proposition classification relying on the merging of labels obtained through local optimization, while the structured learning approach exploits a



Structured SVM full	Predicted P	F	T	V	R
True P	0.76	0.05	0.04	0.16	0.00
True F	0.04	0.44	0.10	0.42	0.00
True T	0.01	0.06	0.72	0.21	0.00
True V	0.05	0.11	0.08	0.76	0.00
True R	0.00	0.00	0.00	0.00	1.00

Baseline LG	Predicted P	F	T	V	R
True P	0.78	0.00	0.01	0.21	0.00
True F	0.08	0.00	0.04	0.88	0.00
True T	0.00	0.00	0.70	0.30	0.00
True V	0.08	0.00	0.09	0.82	0.00
True R	0.00	0.00	1.00	0.00	0.00

ResNet LG	Predicted P	F	T	V	R
True P	0.76	0.06	0.01	0.17	0.00
True F	0.06	0.42	0.08	0.44	0.00
True T	0.00	0.06	0.75	0.18	0.00
True V	0.07	0.12	0.10	0.70	0.00
True R	0.00	0.00	0.00	0.00	1.00

Structured RNN basic	Predicted P	F	T	V	R
True P	0.73	0.10	0.00	0.17	0.00
True F	0.07	0.48	0.06	0.38	0.00
True T	0.01	0.08	0.73	0.19	0.00
True V	0.05	0.15	0.08	0.71	0.00
True R	0.00	0.00	0.00	0.00	1.00

Baseline PG	Predicted P	F	T	V	R
True P	0.74	0.00	0.02	0.24	0.00
True F	0.03	0.00	0.08	0.89	0.00
True T	0.01	0.00	0.63	0.35	0.00
True V	0.05	0.00	0.09	0.86	0.00
True R	0.00	0.00	0.00	0.00	1.00

ResNet PG	Predicted P	F	T	V	R
True P	0.78	0.07	0.02	0.12	0.00
True F	0.06	0.45	0.09	0.40	0.00
True T	0.02	0.08	0.69	0.22	0.00
True V	0.08	0.16	0.13	0.64	0.00
True R	0.00	0.00	0.00	0.00	1.00

Attention LG	Predicted P	F	T	V	R
True P	0.80	0.08	0.01	0.11	0.00
True F	0.02	0.52	0.05	0.41	0.00
True T	0.00	0.05	0.80	0.14	0.00
True V	0.04	0.10	0.06	0.80	0.00
True R	0.00	0.00	0.00	0.00	1.00

Figure 7.3: Confusion matrices for component classification on CDCP. From left to right: Structured Learning approach, deep networks baselines, and our architectures.

global optimization. Nonetheless, the average score of residual networks is better than that of structured RNNs, thus proving the generality of the approach.

We shall also remark that our approach can achieve such results without exploiting any specific hypothesis or a priori knowledge of the genre or domain. This could be an added value in contexts where arguments may be laid out freely, without following a pre-determined argument model, yet it would be interesting to uncover the underlying argumentation's structure.

Results also indicate that the most common mistake regards the prediction of facts as values (see Figure 7.3). That should come as no surprise, since VALUE is by far the largest class in the corpus, and it is therefore also affected by many false positives. One of the most common errors is the misclassification of facts as values, which reflects an ambiguity between the two classes that has been reported also during the annotation process. Interestingly, the confusion table of the structured approach and of our method are very similar, so it is possible to speculate that our networks may have learned a similar behavior despite not having received any constraint or information regarding the argumentative structure.

As far as relation label prediction is concerned, this model apparently fails to predict the EVIDENCE relation. That negative result was also to be expected since such a class is scarcely present in the whole dataset (less than 1%).

Table 7.3: Results of the experiments involving multiple trainings of the same models and use of attention on CDCP. From left to right, the average scores of the RESARG architecture, the scores of the ensemble learning setting of the same model, the average and the ensemble scores of the RESATTARG architecture, and the best results of structured approaches based on SVM and RNN. F_1 and macro F_1 percentage scores are reported.

Approach	RESARG		RESATTARG		Structured	
	Average	Ensemble	Average	Ensemble	SVM	RNN
Avg (Link and Proposition)	47.75	52.14	51.57	**54.22**	50	43.5
Link (272)	24.99	28.76	27.40	**29.73**	26.7	14.6
No-Link (9212)			98.03	98.32		
Proposition (973)	70.51	75.53	75.75	**78.71**	73.50	72.7
VALUE (491)	72.30	75.37	77.84	80.37	76.4	73.70
POLICY (153)	75.39	79.60	80.09	82.55	77.3	76.8
TESTIMONY (204)	73.46	76.33	76.42	81.19	71.7	75.8
FACT (124)	41.39	46.37	44.39	49.42	42.5	42.2
REFERENCE (1)	90	100	100	100	100	100
Relation (272)	13.8	14.19	15.05	**15.28**		
REASON (265)	25.1	28.39	27.88	30.56		
EVIDENCE (7)	2.5	0	2.22	0		
Relations and No-Rel			42.69	42.95		

7.4.2 Multiple Trainings and Ensemble Learning

Then we wanted to investigate whether our results with a single model were solid or they were a lucky consequence of the non-deterministic nature of network training. We have therefore repeated the LG training procedure so as to obtain 10 models. For each metric, we have computed the average score obtained by the 10 models. To improve our results and obtain a more stable method, we have also decided to use the networks in an ensemble fashion, using majority voting to establish the class of each element. We show the results in Table 7.3.

The average of the 10 networks leads to a worse outcome with respect to the state of the art in both Link Prediction and Component Classification. This raises important questions regarding the reliability of our first stage of experiments. Compared to our previous results, it is worth noticing that we lose about 5 percentage points in Link Prediction, but improve by a similar amount our score for Component Classification.

On the contrary, the ensemble approach is proven to be valuable, substantially improving the results, and overcoming the structured learning approach on both the tasks at hand. The result on link prediction still does not reach the value obtained in the first experiment, but the gap is less than 1 percent.

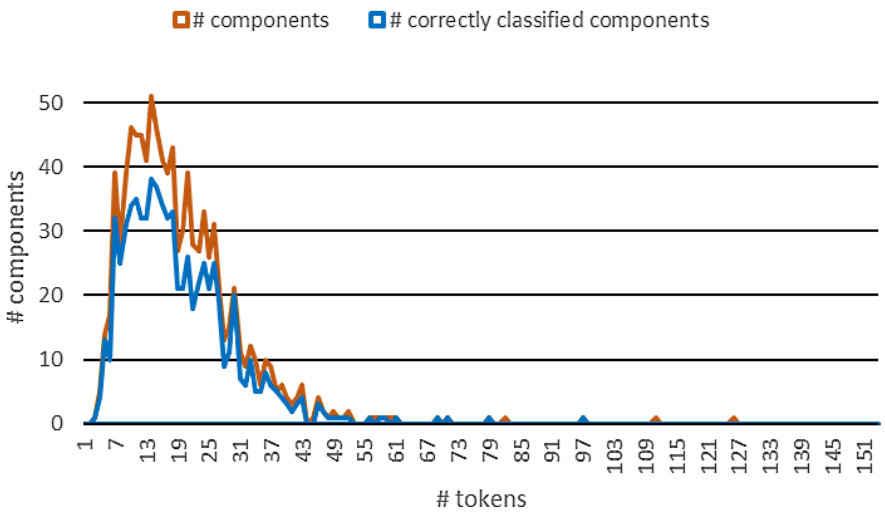


Figure 7.4: Total number of components and number of correctly classified components in relation with their length.

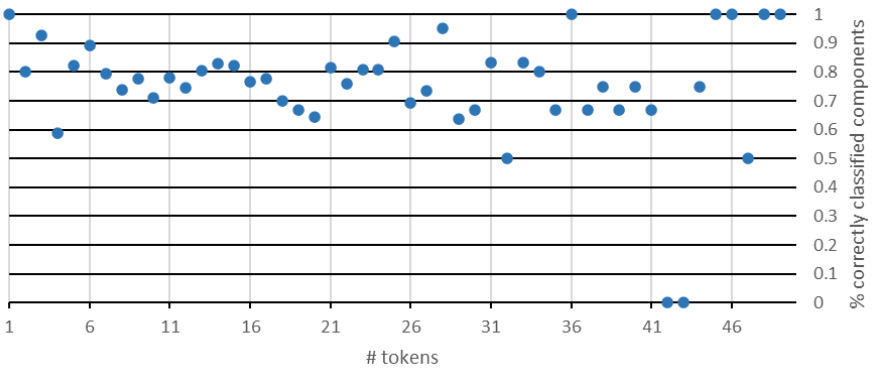


Figure 7.5: Percentage of correctly classified components in relation with their length. Only lengths between 1 and 50 tokens are considered.

7.4.3 Attention-based architecture

Introducing the attention module in the architecture has lead to appreciable improvements as shown in Table 7.3. The RESATTARG architecture improved both the average and the ensemble approaches, with the latter yielding better results. The new model, used in ensemble fashion, established itself as state-of-the-art on this dataset, outperforming all the previous methods in each task, even in the relation classification task. To estimate the agreement among the predictions of the networks in ensemble, we have computed Krippendorff's alpha for the three tasks, obtaining $\alpha = 0.70$ for component classification, and $\alpha = 0.45$ for both link prediction and relation classification. These values are similar to the IAA obtained by the authors of the dataset and confirm the difficulty of the link prediction task.

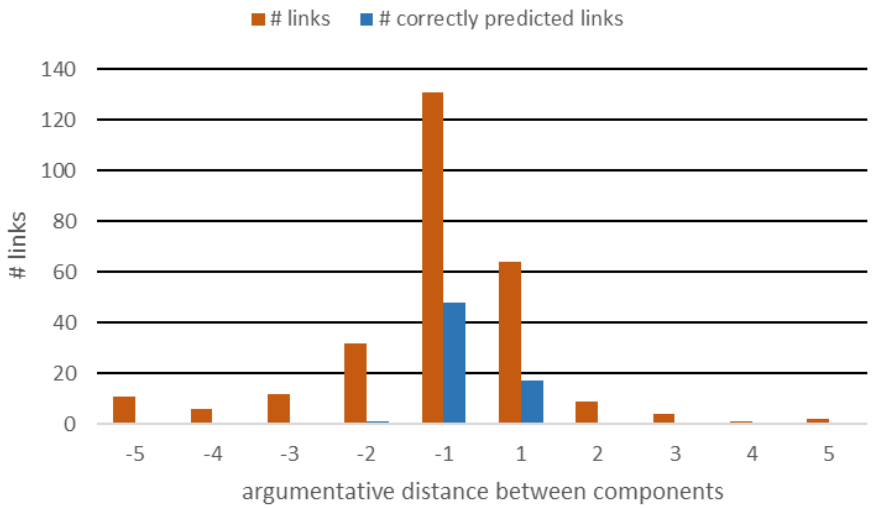


Figure 7.6: Number of linked pairs and correctly predicted links in relation with the argumentative distance between components

To better understand if there are factors that may influence the classification, we have analyzed the impact of the length of the components on their classification, and the impact of the distance between components on link prediction. As it can be seen in Figures 7.4 and 7.5, the number of tokens in a component does not seem to affect the ability of the networks to classify it. Figure 7.6 clearly shows that link prediction is highly affected by the argumentative distance between components, failing to predict almost any link between non adjacent components.

Encouraged by these results, we have decided to validate our attention-based model with ensemble also on three other corpora, which present very different characteristics.

7.4.4 Other Corpora

UKP

We tested our method on the UKP dataset, comparing it to the ILP joint model of its authors [247] and to Niculae et al.'s structured learning approach. We conducted the comparison on the test split of the dataset and use about 10% of the documents of the training split as validation split. The composition of the dataset is reported in Table 7.4.

As reported in Table 7.5, our approach is much worse than previous works, obtaining between 20 and 30 percentage points less in both the macro F_1 scores. The agreement between the networks is low as well, with $\alpha = 0.57$ for component classification and $\alpha = 0.38$ for link prediction, assessing them as nearly acceptable for the first task but unreliable for the others.

Table 7.4: UKP dataset composition.

Split	Train	Valid.	Test	Total
Paragraphs	1,229	176	359	1,764
Propositions	4,224	599	1,266	6,089
Premises	2,649	374	809	3832
Claims	1,051	151	304	1,506
Major Claims	524	74	153	751
Couples	19,338	2,136	4,922	26,396
Links	2,649	374	809	3,832
Support	2,493	353	767	3,613
Attack	156	21	42	219

These results are not surprising, since the dataset is well structured, and previous approaches rely on features and constraints tailored for this specific corpus and its argumentation model. For example, as reported by Stab and Gurevych, structural features that capture the position of the component in the document are valuable for the identification of major claims since they appear mostly in introductions or conclusions.

AbstRCT

For what concerns AbstRCT, we compared our RESATTARG architecture with ensemble against the best methods presented by its authors [181], which are reported in Table 7.7 and Table 7.8. Following the original paper, we trained and validated our model on the respective splits of the Neoplasm dataset, using the remaining part of the corpus as test. The composition of each split is reported in Table 7.6.

It is important to underline that Mayer et al. approach components classification as 3-class sequence tagging which includes the "non-argumentative" class, while we perform it as a 2-class classification. Moreover, since they employ a pipeline scheme, errors obtained during their first step may introduce noise in the link prediction/relation classification task. Our approach instead takes the argumentative components as already selected and perfectly bounded. Unfortunately, they do not report relation classification on the golden standard components, therefore this comparison must be considered qualitative since our approach is advantaged. Finally, their approach is completely distance independent, while our approach relies on an indication of distance as a feature, but it is not limited in its application. However, both approaches compare every possible pair of components, therefore the size of the dataset grows quadratically with the number of components in the document, which makes both of them not scalable to large documents.

Table 7.5: Results on UKP in terms of F_1 and macro F_1 scores. For the comparison with structured learning, the best scores obtained by any of the structured configurations are reported.

Approach	RESATTARG Ensemble	ILP joint model	Structured SVM	Structured RNN
Average (Link and Components)	38.68	**70.55**	68.85	64.85
Link (809)	36.3	58.5	**60.1**	50.4
No-Link (4,113)	88.61	91.8		
Components (1,266)	52.51	**82.6**	77.6	79.3
PREMISE (809)	81.59	**90.3**	**90.3**	87.6
CLAIM (304)	42.09	68.2	64.5	62
MAJOR CLAIM (153)	33.86	89.1	80	88.3
Relation (809)	18.06			
SUPPORT (767)	36.11			
ATTACK (42)	0			
Relations and No-Rel	88.7			

For what concerns components classification, our attention-based approach is comparable with the state-of-the-art. We yield better results on all datasets for what concerns the micro f_1 score. For what concerns macro F_1, we improve the previous approaches on Neoplasm, while BioBERT achieves a few points more on Glaucoma and Mixed. In relation classification, our model is better than all the others on Neoplasm and Glaucoma, while on the Mixed dataset it scores 1 point less than SciBERT. It is important to point out that BioBERT in this task is greatly overcome by our approach.

The agreement between the networks is very high for token-wise component classification in each dataset (α between 0.81 and 0.83), and lower but still acceptable for the other two tasks ($\alpha = 0.67$ on neoplasm and $\alpha = 0.62$ for the other two).

The success of our method on this dataset proves that it may be valuable also for well-structured corpora. Moreover, it is important to underline that the previous approaches have made use of contextual embeddings, pre-trained using domain-related corpora. On the contrary, we rely on rather simple embeddings (non contextualized, general-purpose). The combination of our architecture with sophisticated embeddings is an interesting future path of research.

In Table 7.9 and 7.10 we report more details regarding our results with the attention-based architecture, for sake of comparability with future approaches.

Table 7.6: AbstRCT dataset composition.

Dataset	Neoplasm			Glaucoma	Mixed
Split	Train	Valid.	Test	Test	Test
Documents	350	50	100	100	100
Components	2,267	326	686	594	600
Evidence	1,537	218	438	404	338
Claim	730	108	248	190	212
Couples	14,286	2,030	4,380	3,332	3,332
Links	1,418	219	424	367	329
Support	1,213	186	364	334	305
Attack	205	33	60	33	24

Table 7.7: Results of token-wise component classification on AbsRCT. We report the F_1 related to the micro and macro averaged score obtained on the 3 test sets.

	Neoplasm		Glaucoma		Mixed	
Approach	micro f_1	macro F_1	micro f_1	macro F_1	micro f_1	macro F_1
Mayer et al.						
BioBERT+GRU+CRF	90	84	92	**91**	92	**91**
SciBERT+GRU+CRF	90	87	91	89	91	88
RESATTARG	**92.12**	**90.14**	**92.92**	89.35	**92.79**	90.26

Dr. Inventor

Following previous work [138], we reserve 30% of the documents of the corpus as test set, and 20% of the remaining part as validation set. The composition of the resulting dataset is reported in Table 7.11.

To the best of our knowledge, no approaches have been tested on the DrInventor corpus besides a simple baseline, which performs token-wise component classification making use of GloVe embeddings and a Bi-LSTM followed by a feed-forward neural network with a single hidden layer as classifier. We outperform this baseline obtaining a macro F_1 score of 0.66 against a previous result of 0.45. Tables 7.12 and 7.13 includes a detailed report of our performance on the dataset. It is worthy to remember that for the tasks of link prediction and relation classification we are considering a limited number of pairs and that we have made each part of the same component equal in their relations.

As it was foreseeable, our model is not capable to classify the relation SEMANTICALLY SAME and has difficulties also with CONTRADICTS, which are the two less represented classes in the dataset. It is less straightforward to understand why the model is better at

Table 7.8: Results of relation classification between 3 classes (support, attack, no-link) on AbsRCT. We report the macro F_1 score obtained on the 3 test sets.

Approach	Neoplasm	Glaucoma	Mixed
BioBERT	64	58	61
SciBERT	68	62	**69**
RoBERTa	67	66	67
RESATTARG	**70.92**	**68.40**	67.66

Table 7.9: Results of Link Prediction and Relation Classification on AbsRCT. The “Relation" column reports the result of the macro F_1 score for the support and attack classes in Relation Classification. The remaining columns report the F_1 score of the respective classes.

Test set	**Link**	No-Link	**Relation**	Supp	Att
Neoplasm	54.43	94.54	59.08	52.77	65.38
Glaucoma	55.23	94.36	55.37	54.73	56
Mixed	51.2	94.21	54.35	49.62	59.09

classifying BACKGROUND CLAIM rather than DATA, even if the latter are more represented than the former. We speculate it may be related to the fact that data may be other than proper sentences. Figure 7.7 represent the confusion table regarding the components classification task, which confirms the model's predilection for predicting claims, while it rarely misclassifies claims as data.

Differently from previous experiments, the agreement between the networks is similar for all the tasks, with only $\alpha = 0.56$ for component classification and $\alpha = 0.60$ for the remaining tasks.

7.5 Discussion

We presented the first application of residual networks in the argument mining domain. We proposed a model, RESARG, that outperforms an equivalent deep network and competes with state-of-the-art techniques on a challenging dataset. We have seen that the introduction of an attention module and ensemble learning give a positive contribution. Our RESATTARG model performed poorly on a second corpus, which is characterized by being well structured and following an argumentation model that imposes many constraints. On two additional corpora, we have instead obtained satisfying results.

Table 7.10: Results of Component Classification on AbsRCT. “Average" is the macro F_1 score, the remaining columns report the F_1 score of the respective classes.

	Token-wise			Component-wise		
Test set	**Average**	Evidence	Claim	**Average**	Evidence	Claim
Neoplasm	90.14	94.56	85.72	87.87	91.44	84.30
Glaucoma	89.35	95.52	83.19	87.71	92.69	86.54
Mixed	90.26	95.23	85.3	89.70	92.86	82.72

Table 7.11: DrInventor dataset composition.

Split	Train	Valid.	Test	Total
Documents	142	38	83	263
Components	7,693	2,019	3,879	13,591
Backg. Claim	1,853	434	1,004	3,291
Own Claim	3,395	958	1,651	6,004
Data	2,445	627	1,224	4,296
Couples	138,356	36,230	68,680	243,266
Links	4,875	1,392	2,438	8,705
Support	4,311	1,284	2,187	7,782
Contradicts	510	106	217	833
Semantically Same	54	2	34	90

Considering that the model makes use of only one simple feature – the argumentative distance between two propositions – a natural extension of this study would be its integration in a more structured and constrained argumentation framework.

Since in argumentation it is often the case that single propositions cannot contain all the relevant information to predict argument components and relations, it could be useful to provide also the context of argumentation as an input. Hence, another interesting direction of investigation could be the integration of the whole document text in the model.

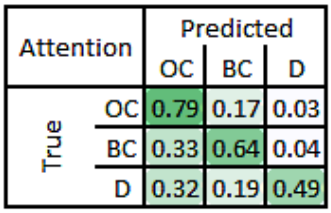


Figure 7.7: Confusion matrix for component classification on DrInventor.

Table 7.12: Results of token-wise component classification on DrInventor. “Average" is the macro F_1 score, the remaining columns report the F_1 score of the respective classes.

Token-wise				Component-wise			
Average	Own Claim	Background Claim	Data	**Average**	Own Claim	Background Claim	Data
65.71	78.03	61.58	57.53	69.64	73.13	59.63	76.15

Table 7.13: Results of Link Prediction, and Relation Classification on DrInventor. The “Relation" column reports the result of the macro F_1 score for the Support, Contradicts, and Semantically Same classes in Relation Classification. The remaining columns report the F_1 score of the respective classes. The last column includes also the “No-Rel" class in the average.

Link	No-Link	**Relation**	Supp.	Contrad.	Sem. Same	Relations and No-Rel
43.66	98.36	17.50	45.90	6.61	0	37.72

Part III

Neural-Symbolic Argument Mining

Chapter 8

Towards Neural-symbolic Argument Mining

Deep learning is bringing remarkable contributions to the field of argument mining, but the existing approaches still need to fill the gap towards performing advanced reasoning tasks. In this Chapter, we posit that neural-symbolic and statistical relational learning could play a crucial role in the integration of symbolic and sub-symbolic methods to achieve this goal. The discussion presented in this Chapter is based on the work of Galassi et al. [79].

8.1 Introduction

The majority of AM systems follows a pipeline scheme, starting with simpler tasks such as argument component detection, down to more complex tasks such as argumentation structure prediction. Recent years have seen the development of a large number of techniques in this area, on the wake of the advancements produced by deep learning on the whole research field of natural language processing (NLP). Yet, it is widely recognized that the existing AM systems still have a large margin of improvement, as good results have been obtained with some genres where prior knowledge on the structure of the text eases some AM tasks, but other genres such as legal cases and social media documents still require more work [27]. Performing and understanding argumentation requires advanced reasoning capabilities, which are natural human skills, but are difficult to learn for a machine. Understanding whether a given piece of evidence supports a given claim, or whether two claims attack each other, are complex problems that humans can address thanks to their ability to exploit commonsense knowledge, and to perform reasoning and inference. Despite the remarkable impact of deep neural networks in NLP, we argue that these techniques alone will not suffice to address such complex issues.

We envisage that a significant advancement in AM could come from the combination of symbolic and sub-symbolic approaches, such as those developed in the Neural Symbolic (NeSy) [50] or Statistical Relational Learning (SRL) [53, 88, 133] communities. This issue is also widely recognized as one of the major challenges for the whole field of artificial intelligence in the coming years [142].

In computational argumentation, structured arguments have been studied and formalized for decades using models that can be expressed in a logic framework [14]. At the same time, AM has rapidly evolved by exploiting state-of-the-art neural architectures coming from deep learning. So far, these two worlds have progressed largely independently of each other. Only recently, a few works have taken some steps towards the integration of such methods, by applying techniques combining sub-symbolic classifiers with knowledge expressed in the form of rules and constraints to AM. For instance, Niculae et al. [194] adopted structured support vector machines and recurrent neural networks to collectively classify argument components and their relations in short documents, by hard-coding contextual dependencies and constraints of the argument model in a factor graph. A joint inference approach for argument component classification and relation identification was instead proposed by Persing and Ng [202], following a pipeline scheme where integer linear programming is used to enforce mathematical constraints on the outcomes of a first-stage set of classifiers. More recently, Cocarascu and Toni [42] combined a deep network for relation extraction with an argumentative reasoning system that computes the dialectical strength of arguments, for the task of determining whether a review is truthful or deceptive.

We propose to exploit the potential of both symbolic and sub-symbolic approaches for AM, by combining both results in systems that are capable of modeling knowledge and constraints with a logic formalism, while maintaining the computational power of deep networks. Differently from existing approaches, we advocate the use of a logic-based language for the definition of contextual dependencies and constraints, independently of the structure of the underlying classifiers. Most importantly, the approaches we outline do not exploit a pipeline scheme, but rather perform joint detection of argument components and relations through a single learning process.

8.2 Modeling Argumentation with Probabilistic Logic

Computational argumentation is concerned with modelling and analyzing argumentation in the computational settings of artificial intelligence [14, 212]. The formalization of arguments is usually addressed at two levels. At the argument level, the definition of formal languages for representing knowledge and specifying how arguments and counterarguments can be

constructed from that knowledge is the domain of *structured argumentation* [18]. In structured argumentation, the premises and claim of the argument are made explicit, and their relationships are formally defined. However, when the discourse consists of multiple arguments, such arguments may conflict with one another and result in logical inconsistencies. A typical way of dealing with such inconsistencies is to identify sets of arguments that are mutually consistent and that collectively defeat their "attackers". One way to do that is to abstract away from the internal structure of arguments and focus on the higher-level relations among arguments: a conceptual framework known as *abstract argumentation* [64].

Similarly to structured argumentation, AM too builds on the definition of an argument model, and aims to identify parts of the input text that can be interpreted as argument components [159]. For example, if we take a basic claim/evidence argument model, possible tasks could be claim detection [3, 157], evidence detection [220], and the prediction of links between claim and evidence [77, 194]. However, in structured argumentation the formalization of the model is the basis for an inferential process, whereby conclusions can be obtained starting from premises. In AM, instead, an argument model is usually defined in order to identify the target classes, and in some isolated cases to express relations, for instance among argument components [194], but not for producing inferences that could help the AM tasks.

The languages of structured argumentation are logic-based. An influential structured argumentation system is *deductive argumentation* [17], where premises are logical formulae, which entail a claim, and entailment may be specified from a range of base logics, such as classical logic or modal logic. In *assumption-based argumentation* [65] instead arguments correspond to assumptions, which like in deductive systems prove a claim, and attacks are obtained via a notion of contrary assumptions. Another powerful framework is *defeasible logic programming* (DeLP) [83], where claims can be supported using strict or defeasible rules, and an argument supporting a claim is warranted if it defeats all its counterarguments. For example, that a cephalopod is a mollusc could be expressed by a strict rule such as:

$$mollusc(X) \leftarrow cephalopod(X)$$

as these notions belong to an artificially defined, incontrovertible taxonomy. However, since in nature not all molluscs have a shell, and actually cephalopods are molluscs without a shell, rules used to conclude that a given specimen has or does not have a shell are best defined as defeasible. For instance, one could say:

$$\begin{aligned} has\_shell(X) &\prec mollusc(X) \\ \sim has\_shell(X) &\prec cephalopod(X) \end{aligned}$$

where $\prec$ denotes defeasible inference.

The choice of logic notwithstanding, rules offer a convenient way to describe argumentative inference. Moreover, depending on the application domain, the document genre, and the employed argument model, different constraints and rules can be enforced on the structure of the underlying network of arguments. For example, if we adopt a DeLP-like approach, strict rules can be used to define the relations among argument components, and defeasible rules to define context knowledge. For example, in a hypothetical claim-premise model, support relations may be defined exclusively between a premise and a claim. Such structural properties could be expressed by the following strict rules:

$$\begin{aligned} claim(Y) &\leftarrow supports(X,Y) \\ premise(X) &\leftarrow supports(X,Y) \end{aligned}$$

whereby if X supports Y, then X is a claim and Y is a premise. As another abstract example, two claims based on the same premise may not attack each other:

$$\sim attacks(Y1,Y2) \leftarrow supports(X,Y1) \wedge supports(X,Y2)$$

As an example of defeasible rules, consider instead the context information about a political debate, where a republican candidate, R, faces a democrat candidate, D. Then one may want to use the knowledge that R's claims and D's claims are likely to attack each other:

$$attacks(Y1,Y2) \prec auth(Y1,R) \wedge rep(R) \wedge auth(Y2,D) \wedge dem(D)$$

where predicate $auth(A,B)$ denotes that claim A was made by B. There exist many non-monotonic reasoning systems that integrate defeasible and strict inference. However, an alternative approach that may successfully reconcile the computational argumentation view and the AM view is offered by probabilistic logic programming (PLP). PLP combines the capability of logic to represent complex relations among entities with the capability of probability to model uncertainty over attributes and relations [219]. In a PLP framework such as PRISM [225], LPAD [264] or ProbLog [210], defeasible rules may be expressed by rules with a probability label. For instance, in LPAD syntax, one could write:

$$attacks(Y1,Y2) : 0.8 \leftarrow auth(Y1,R) \wedge rep(R) \wedge auth(Y2,D) \wedge dem(D)$$

to express that the above rule holds in 80% of cases. In this example, 0.8 could be interpreted as a weight or score suggesting how likely the given inference rule is to hold. In more recent approaches, such weights could be learned directly from a collection of examples, for example

by exploiting likelihood maximization in a *learning from interpretations* setting [100] or by using a generalization of expectation maximization applied to logic programs [12].

From a higher-level perspective, rules could be exploited also to model more complex relations between arguments or even to encode argument schemes [268], for example to assess whether an argument is defeated by another, on the basis of the strength of its premises and claims. This is an even more sophisticated reasoning task, which yet could be addressed within the same kind of framework described so far.

8.3 Combining Symbolic and Sub-Symbolic Approaches

The usefulness of deep networks has been tested and proven in many NLP tasks, such as machine translation [285], sentiment analysis [293], text classification [46, 297], relations extraction [120], as well as in AM [41, 42, 52, 77, 138, 166, 226]. While a straightforward approach to exploit domain knowledge in AM is to apply a set of hand-crafted rules on the output of some first stage classifier (such as a neural network), NeSy or SRL approaches can directly enforce (hard or soft) constraints *during training*, so that a solution that does not satisfy them is penalized, or even ruled out. Therefore, if a neural network is trained to classify argument components, and another one[1] is trained to detect links between them, additional global constraints can be enforced to adjust the weights of the networks towards admissible solutions, as the learning process advances. Systems like DeepProbLog [176], Logic Tensor Networks [229], or Grounding-Specific Markov Logic Networks (GS-MLN) [156], to mention a few, enable such a scheme.

By way of illustration, we report how to implement one of the cases mentioned in Section 8.2 with DeepProbLog and with GS-MLNs. By extending the ProbLog framework, DeepProbLog allows to introduce the following kind of construct:

$$nn(m,\vec{t},\vec{u}) :: q(\vec{t},u_1);\ldots;q(\vec{t},u_n).$$

The effect of the construct is the creation of a set of ground probabilistic facts, whose probability is assigned by a neural network. This mechanism allows us to delegate to a neural network m the classification of a set of predicates q defined by some input features $\vec{t}$. The possible classes are given by $\vec{u}$. Therefore, in the AM scenario, it would be possible, for example, to exploit two networks *m_t* and *m_r* to classify, respectively, the type of a potential argumentative component

[1] In a multi-task setting, instead of two networks, the same network could be used to perform both component classification and link detection at the same time, as we have illustrated in the previous Chapter.

```
nn(m_t,H,[claim,prem,non_arg]) ::
    type(H,claim);
    type(H,premise);
    type(H,non_arg).

nn(m_r,H1,H2,[att,supp,none]) ::
    rel(H1,H2,att);
    rel(H1,H2,supp);
    rel(H1,H2,none).
```

(a) Definition of neural predicates

```
type(Y,claim) :- rel(X,Y,supp).
type(X,premise) :- rel(X,Y,supp).

\+rel(Y1,Y2,att) :-
    rel(X,Y1,supp),
    rel(X,Y2,supp).

0.8::rel(Y1,Y2,att) :-
    made_by(Y1,R), rep(R),
    made_by(Y2,D), dem(D).
```

(b) Definition of (probabilistic) rules

Figure 8.1: An excerpt of a DeepProbLog program for AM.

and the potential relation between two components. Figure 8.1a shows the corresponding DeepProbLog code. These predicates could be easily integrated within a probabilistic logic program designed for argumentation, so as to model (possibly weighted) constraints, rules, and preferences, such as those described in Section 8.2. Figure 8.1b illustrates one such possibility.

The same scenario can be modeled using GS-MLNs. In the Markov logic framework, first-order logic rules are associated with a real-valued weight. The higher the weight, the higher the probability that the clause is satisfied, other things being equal. The weight could possibly be infinite, to model hard constraints. In the GS-MLN extension, different weights can be associated to different groundings of the same formula, and such weights can be computed by neural networks. Joint training and inference can be performed, as a straightforward extension of the classic Markov logic framework [156]. Figure 8.2 shows an example of a GS-MLN used to model the AM scenario that we consider. Here, the first three rules model grounding-specific clauses (the dollar symbol indicating a reference to a generic vector of features describing the example) whose weights depend on the specific groundings (variables `x` and `y`); the three subsequent rules are hard constraints (ending with a period); the final rule is a classic Markov logic rule, with a weight attached to the first-order clause.

The kind of approach hereby described strongly differs from the existing approaches in AM. Whereas Persing and Ng [202] exploit a pipeline scheme to apply the constraints to the predictions made by deep networks at a first stage of computation, the framework we propose can perform a joint training, which includes the constraints within the learning phase. This can be viewed as an instance of *Constraint Driven Learning* [32] and its continuous counterpart, posterior regularization [81], where multiple signals contribute to a global decision, by being pushed to satisfy expectations on the global decision. Differently from the work by Niculae et al. [194], who use factor graphs to encode inter-dependencies between random variables, our

```
w1(x)    Sentence(x,$fx) => Claim(x)
w2(x)    Sentence(x,$fx) => Premise(x)
w3(x,y)  Sentence(x,$fx) ^ Sentence(y,$fy) => Supports(x,y)
         Supports(x,y) => Claim(y).
         Supports(x,y) => Premise(x).
         Supports(x,y1) ^ Supports(x,y2) => !Attacks(y1,y2).
w4       Rep(r) ^ Dem(d) ^ MadeBy(y1,r) ^ MadeBy(y2,d) =>
             Attacks(y1,y2)
```

Figure 8.2: An excerpt of a GS-MLN for the definition of neural, hard and weighted rules for AM.

approach enables to exploit the interpretable formalism of logic to represent rules. Moreover, the models of NeSy and SRL are typically able to *learn* the weights or the probabilities of the rules, or even to learn the rules themselves, thus addressing a *structure learning* task.

8.4 Discussion

After many years of growing interest and remarkable results, time is ripe for AM to move forward in its ability to support complex arguments. To this end, we argue that research in this area should aim at combining sub-symbolic and symbolic approaches, and that several state-of-the-art ML frameworks already provide the necessary ground for such a leap forward.

The combination of such approaches will leverage different forms of abstractions that we consider essential for AM. On the one hand, (probabilistic) logical representations enable to specify AM systems in terms of data, world knowledge and other constraints, and to express uncertainties at a logical and conceptual level rather than at the level of individual random variables. This would make AM systems easier to interpret — a feature that is now becoming a need for AI in general [99] — since they could help explaining the logic and the reasons that lead them to produce their arguments, while still dealing with the uncertainties stemming from the data and the (incomplete) background knowledge. On the other hand, AM is too complex to fully specify the distributions of random variables and their global (in)dependency structure a priori. Sub-symbolic models can harness such a complexity by finding the right, general outline, in the form of computational graphs, and processing data.

In order to fully exploit the potential of this joint approach, clearly many challenges have to be faced. First of all, several languages and frameworks for NeSy and SRL exist, each with its own characteristics in terms of both expressive power and efficiency. In this sense, AM would represent an ideal test-bed for such frameworks, by presenting a challenging, large-scale application domain where the exploitation of a background knowledge could play a crucial role to boost performance. Inference in this kind of models is clearly an issue, thus AM would

provide additional benchmarks for the development of efficient algorithms, both in terms of memory consumption and running time. Finally, although there are already several NeSy and SRL frameworks available, being these research areas still relatively young and in rapid development, their tools are not yet mainstream. Here, an effort is needed in integrating such tools with state-of-the-art neural architectures for NLP.

Chapter 9

Logic Tensor Networks for Neural-Symbolic AM

Typically the argumentative components of a document are interconnected with each other, so as to form an argumentative graph. The task of classifying a single component (or relationship) would probably benefit from considering not only the attributes of the component itself but also the attributes of the components and of the relationship that belong to the same neighborhood in the argumentative graph. The task of AM becomes therefore a task of collective classification [227], which is the combined classification of a set of interlinked objects according to the attributes of those objects and the objects in their neighborhood.

Most of the existing neural-symbolic frameworks may present difficulties of use for AM in their current implementation if they do not support collective classification. Indeed, the training procedures consider only a limited and well-defined amount of entities at the same time, whereas AM requires to consider a variable number of entities at the same time (the components of the document) and all the potential relationships between them. After an initial investigation of several neural-symbolic implementations, we decided to use the Logic Tensor Networks framework for argument mining and conducted what is, to the best of our knowledge, the first experiment of neural-symbolic AM.

While our experimental setting cannot be compared to the state of the art, our findings confirm that the introduction of logic rules may improve neural networks for link prediction under the perspective of accuracy, robustness, and respect of the domain properties.

9.1 Logic Tensor Networks

Logic Tensor Networks (LTN) [59, 60, 228, 229] is a framework that integrates first-order many-valued logical reasoning [16] with tensor networks [239], implemented in TensorFlow [1]. LTN belongs to the "tensorization" approaches, a class of undirect neuro-symbolic approaches [211] which embed First Order Logic entities, such as constants and facts, into real-valued tensors. The framework allows combining data-driven machine learning with background knowledge expressed through first order fuzzy logic representation. Therefore, it is possible to use FOL to impose soft constraints at training time and to verify and investigate properties at test time.

LTN variables are an abstract representation of data, which must be linked to a set of real-valued vectors. A single data point of this set can be represented using LTN constants. LTN functions represent operations that can be done over variables, and produce real-valued vectors as result. The evaluation is done by a set of TensorFlow operations, e.g., a neural network, which are specified when the function is defined. LTN predicates are defined as functions whose output is a single real value between 0 and 1, which represent the degree of truth of the predicate. They can be used to represent classes of objects as well as properties that may exist between multiple objects. The learning setting is defined in terms of LTN *axioms*, which specify which logic conditions must hold, and therefore allows to assign labels to data and to specify soft constraints. Axioms can be expressed by making use of logical connectives (and, or, not, implication) and quantifiers ($\forall$, $\exists$).

LTN, similarly to DeepProbLog, allows creating *vertical-hybrid* learning systems, where high-level logic is placed on top of deep networks, as opposed to *horizontal-hybrid* learning (e.g., the work of [116]), where the symbolic knowledge is encoded into the networks themselves [51]. The idea behind the design of these systems is that the symbolic part must influence the behavior of the neural part and provide means to interpret their results. Reasoning is performed in the form of *approximate satisfiability*, which means that the optimization process aims to maximize the level of satisfiability of a grounded theory, by minimizing the loss function [228]. Inference follows a model-theoretic perspective, which means that learning is done through learning the shared parameters over the ground model and inference is based on possible groundings of the model [211].

Once the models have been trained, it is possible to evaluate results doing queries expressed in FOL, which can be used to assess the performance of the neural networks but also to verify the degree of truth of a property, expressed as a logic rule.

9.2 LTN for AM

We define two neural networks *nnComp* and *nnLink*, dedicated respectively to component classification and link prediction. The first network takes a component as input and produces a probability distribution over the possible component classes as output. The second one receives two components and outputs a single value between 0 and 1 which represents the probability of the existence of an argumentative link between two components.

For link prediction we define three variables: *varP*, *varL*, *varNL*. The first is linked to all the possible pairs of components of our training set. The second one to all the training couples where there is an argumentative link, while the third one to all the others. Then, we define a predicate *LINK*, which we associate to the output of *nnLink*.

For component classification we define one variable, *varC*, that is linked to all the components of our training set, and one additional variable *varClassX* for each possible class, that are linked to the related data. For each class, we define a predicate *CLASSX*, with the semantic "belonging to that specific class", that we associate to the respective output of *nnComp*.

Finally, we define the learning setting specifying 2 axioms for the task of link prediction (Eq. 9.1 and 9.2), and one axiom for each component class for the task of component classification (Eq. 9.3).

$$\forall varL : LINK(varL) \tag{9.1}$$

$$\forall varNL : \sim LINK(varNL) \tag{9.2}$$

$$\forall varClX : CLASSX(varClassX) \tag{9.3}$$

Additionally, we can use axioms to specify additional soft constraints, creating new variables when needed. For example, we can define the variables *varC*2 and *varC*3, which are both linked to the components of the training set. Then, it is possible to specify the following axioms to impose the transitivity (Eq. 9.4) and non-symmetry (Eq. 9.5) properties of argumentative links.

$$\forall varC, varC2, varC3 : \\ LINK(varC, varC2) \ \& \ LINK(varC2, varC3) \Rightarrow LINK(varC, varC3) \tag{9.4}$$

$$\forall varC, varC2 : LINK(varC, varC2) \Rightarrow \sim LINK(varC2, varC) \tag{9.5}$$

9.3 Architecture and Method

The current implementation of LTN does not expose APIs to easily configure some aspects of the training procedure. Indeed, to guarantee the consistency of the tensor network, the training procedure employed in our experiments uses a single data batch, which unfortunately has repercussions on the computational resources required.

Due to our limited resources, in this investigation, we need to reduce the size of the data and the complexity of the networks' architecture. We have therefore chosen the AbstRCT corpus [181], described in Chapter 5, as the benchmark, performing AM at the sentence level. We will consequently make use of four predicates, corresponding to the classes of the dataset: *LINK*, *NOLINK*, *EVIDENCE*, and *CLAIM*. Each sentence will be represented using a single sentence embedding, so as to minimize the use of memory. Training, validation, and test process will be conducted on the respective splits.

9.3.1 Architecture

For sentence embeddings, we have decided to use GloVe [201] embeddings of size 25, averaging over the words of the sentences. We have chosen this method for sake of simplicity and because it allows us to obtain low-dimensional embeddings without the need of training new embeddings or relying on dimensionality-reduction techniques that may invalidate the expressiveness of the embeddings. We are well aware that such a technique is naive, and we intend to investigate more advanced techniques in future works, such as Sentence-BERT [215] and Universal Sentence Encodings [29].

For what concerns the networks' architecture, we rely on a simple architecture made of three stacked fully-connected layers and a softmax classifier. We use ReLU as activation function, and employ dropout with probability $p = 0.4$ after each layer. Similar to what we have done in Chapter 7, we train an ensemble of 20 networks both for *nnComp* and *nnLink*, and evaluate the aggregated output. In this context, using majority voting may not be the best way to compute the output of the ensemble. Indeed, it has the drawback of providing a categorical output, not preserving the probabilistic semantic of the prediction. An alternative would be to use the average of the output of the networks, with the inherent vulnerability to outliers. We have therefore decided to pursue both approaches, referring to the first one as MAJ and the second one as AVG, so as to evaluate the results from multiple perspectives.[1] Another possibility that may be explored in future work is to represent the probability score assigned

[1]The MAJ and AVG approaches are just two different methods to aggregate the outputs of the single networks into the output of the ensemble. They do not differ in terms of which networks are involved.

to a class as the percentage of networks that have assigned it the highest probability. Such a method should guarantee robustness while fitting the fuzzy logic semantic of the framework.

9.3.2 Method

To properly evaluate if the use of symbolic rules yields positive results, we perform a baseline training relying only on the data. As a comparison, we train a model including two LTN axioms based on characteristic properties of the dataset: no symmetric link can exist and claims can be linked only to other claims. Such rules can be written as in Eq. 9.5 and Eq. 9.6. Exploiting collective classification, the rules will be applied dataset-wise and the optimizer will have the objective to maximize their satisfiability for any possible pair of components of the training set, not only between components that belong to the same document. To avoid overfitting, we rely early-stopping on the F1 score of the link prediction, using patience of 1000 epochs.

$$\forall varC, varC2 : LINK(varC, varC2) \ \& \ CLAIM(varC) \Rightarrow CLAIM(varC2) \tag{9.6}$$

The evaluation of the two approaches will be based on several aspects. We will measure the F1 metrics regarding link prediction and component classification, to assess if the rules improve the performance of the models. Through LTN queries performed on the AVG ensemble, we will be able to test whether the models respect the two desired properties. Finally, we will compute the degree of agreement between the networks related to the predictions of the MAJ ensemble.

9.4 Experimental Results

Table 9.1 summarizes the results of our experiments. For the classification tasks, we report the F1 score for each class and the macro-F1 score for component classification. The agreement is measured as Krippendorff's alpha, while the degree of truth of the properties is evaluated through LTN queries. For what concerns the AM tasks, the difference between the MAJ and AVG approaches is negligible in the rule-based setting, while it is more accentuated in the no-rules setting for link prediction, where the majority voting improves w.r.t. the alternative by at least 2 percentage points.

The presence of rules seems to be beneficial especially for the task of link prediction, where the networks perform consistently better than the ones trained without rules. Conversely, the latter have performed slightly better on component classification, but such a difference is minimal and not present in all the datasets. The agreement between the networks is at least

Table 9.1: Results of neural-symbolic AM on AbstRCT. The first two column presents the baseline approach, the following two the approach involving rules during the training. Scores are reported as percentage values.

			Only Data		Data+Rules	
Test set	Aspect	Criterion	MAJ	AVG	MAJ	AVG
Neoplasm	Classification	**Components**	79	**80**	79	78
		Evidence	84	85	84	83
		Claim	74	75	74	74
	Agreement	**Link**	34	31	**35**	**35**
		No-Link	81	81	85	85
		Component class.	77		**79**	
		Link prediction	64		**70**	
	Properties	Non-symmetry		98		**100**
		Equation 9.6		**100**		**100**
Glaucoma	Classification	**Components**	**82**	**82**	81	**82**
		Evidence	89	89	88	88
		Claim	75	75	75	75
	Agreement	**Link**	45	43	**47**	45
		No-Link	87	87	90	89
		Component class	**75**		**75**	
		Link prediction	66		**71**	
	Properties	Non-symmetry		99		**100**
		Equation 9.6		**100**		**100**
Mixed	Classification	**Components**	**81**	**81**	**81**	80
		Evidence	86	86	86	85
		Claim	75	75	76	75
	Agreement	**Link**	38	34	39	**40**
		No-Link	83	83	87	87
		Component class.	75		**76**	
		Link prediction	64		**69**	
	Properties	Non-symmetry		98		**100**
		Equation 9.6		**100**		**100**

acceptable in all the settings, with higher values for component classification. The use of rules clearly benefits robustness,[2] boosting the agreement of at least 5 percentage points for link prediction and a few points for component classification. This benefit is confirmed also by the

[2]We mean robustness against the randomness present in the training process [214]. We have not evaluated robustness against other elements, such as noise in the data.

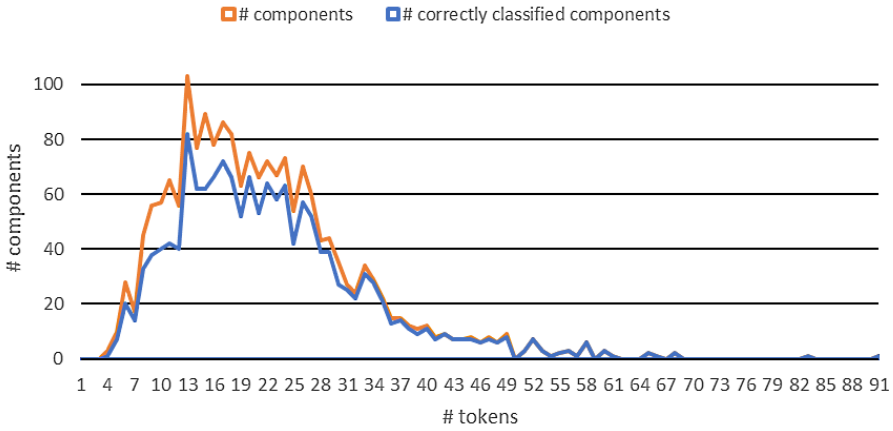


Figure 9.1: Total number of components and number of correctly classified components in relation with their length.

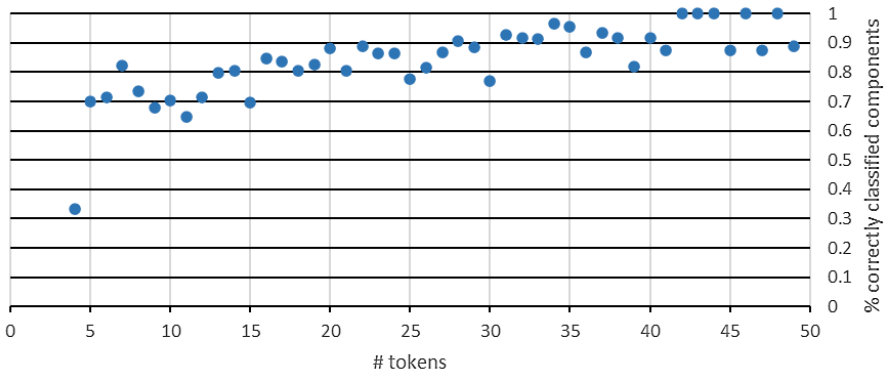


Figure 9.2: Percentage of correctly classified components in relation with their length. Only lengths between 1 and 50 tokens are considered.

smaller difference between the AVG and the MAJ approaches. Finally, even if the data-based approach already satisfies a property completely and the other property almost completely, the introduction of rules during the training phase makes it possible to guarantee their satisfaction to an approximate degree of 100%. The results obtained on the three test sets present minor differences between them, nonetheless, the contribution of the use of rules is clearly visible in all of them.

We have analyzed whether the length of the components influence the performance of the networks on component classification task, testing the ensemble using the MAJ approach on the whole test dataset. Figures 9.1 and 9.2 indicate that the networks are better at classifying longer sentences. One possible explanation to this behaviour is linked to the process of creation of sentence embeddings. Since they are computed as average of word embeddings, in longer sentences the number of words that are significant for the task may surpasse the number of non-

significant words, resulting in a more expressive sentence embedding. We have not observed any significant difference between the behaviour of the networks trained only on the data and those trained with the help of rules.

9.5 Discussion

To the best of our knowledge, our approach has been the first neural-symbolic approach to AM. In our method, logic rules play a role both during the very training process of the neural networks and at inference time as means to investigate the behavior of the models. The definition of training rules and queries requires only to know first order logic, without the need to have any expertise regarding machine learning, neural-symbolic systems, or deep networks. The decoupling between the symbolic and the neural part allows changing either of them without any direct impact on the other, except for the definition of basic concepts such as the predicates/labels of the problem. Such modularity would be highly beneficial in the context of AM, allowing to easily use the same neural architecture in different contexts, since differences across corpora can be expressed through different symbolic rules.

Due to the characteristic of the framework, we have designed an experimental setting that is simpler than the one used in previous chapters. Nonetheless, our results show how the introduction of two symbolic rules has given a positive contribution to the task, increasing all the three aspects we have evaluated: accuracy, robustness, and respect of the properties. Such results are far from the state of the art and our previous results, probably due to the architectures we have used, but we speculate that the impact of rules may hold even for more advanced models.

The most obvious future directions of investigation regard the manipulation of the "softness" of the rules and the use of mini-batches. The former aspect would be beneficial in contexts where some rules express preferences (or theories) while others express constraints. The latter would allow using more complex architectures and a sophisticated experimental setting. Indeed, in our benchmark, the argumentation graphs link only entities that belong to the same document, hence the collective classification may be performed document-wise, rather than dataset-wise. It should be possible to exploit an experimental setting where each mini-batch is associated with a single document, with the consequence that rules will be applied only between elements of the same document. Such a consequence may be a desired property or an unwanted drawback, according to the specific context. A collective classification on the whole corpus would be beneficial in applications where the argumentation spans across multiple domains and the aim is to find relations between components that belong to different documents. This approach suits

contexts such as mining argumentation in social networks [22] or retrieving arguments related to a specific topic [70].

Finally, another direction regards the possibility of training neural networks apt to recognize properties that are not explicit in the training data but can be defined through logical rules. In the context of our setting, it would be possible to define a predicate that represents a property without the need to provide any grounding example for it. This could be achieved by creating axioms that involve such a predicate and other grounded predicates, so as to train the neural network associated with the new predicate along with the ones for which the grounding is provided. This could allow the network to infer information regarding components or relations that are not available in the training data. Two examples in our setting are finding which claim is the major claim of a document, or which components agree with each other, but other more abstract properties may be inferred as well, insofar as they can be defined through rules.

Part IV

Conclusion

Chapter 10

Concluding Remarks and Future Work

The purpose of this thesis has been the investigation of deep networks' behavior in contexts and tasks where symbolic information may play a major role.

In the first part, we have focused on the application of neural models to problems related to the learning of rules and constraints. We have found that deep networks can actually learn complex behavior related to symbolic rules, such as the rules of a game and the constraints of a CSP.

To pursue a deeper analysis of these behaviors, we have decided to focus on the task of Argumentation Mining, a research field that both symbolic and sub-symbolic approaches find challenging. We have seen how the mechanism of Neural Attention can be beneficial, enhancing the performance of the models and making them more interpretable. We have validated Attention for Argument Mining empirically: we have designed a deep neural architecture based on residual networks and attention, obtaining satisfactory results on 3 of the datasets on which we have tested it.

Finally, we have focused on the possibilities of neural-symbolic approaches to AM. We endeavored to do this by firstly discussing the importance of this direction of research and analyzing the possible frameworks, and then concluding by using the Logic Tensor Networks framework to realize what it is, to the best of our knowledge, the first experiment regarding neural-symbolic argument mining. Our results have shown that using a neural-symbolic framework to introduce logical rules during the training process improves the models under all the dimensions we have evaluated: accuracy, robustness, and respect of specific properties of the domain.

While neural-symbolic learning is a trending topic in AI, the technology is not yet ripe for a broad uptake. Many of the existing frameworks are still not developed enough to be used with ease in multiple contexts, and their application is limited to the domain and the case studies proposed by their authors. Among their weak points, there is surely the limited scalability

of most of these methods, which is one of the main open challenges [211]. Moreover, we think that the lack of well-established standards and conventions, and a generally insufficient documentation makes these tools hard to approach for non-expert of the domain. Since the extension of their use to new purposes usually requires a deep knowledge of their functioning, it is difficult to disseminate their use outside of the Neural-Symbolic and Statistical Relational Learning communities, hence their spread across other research fields often weights on the developers of these very frameworks. In this work we have studied an application of such frameworks to a new complex domain, and we have highlighted the requirements we had for the task at hand, the limitations we faced, and some challenges that are still open. We think that future research should focus on these aspects, in order to make these instruments more approachable by scientists from other disciplines and so make their use broader.

Besides the integration of rules given by the domain or by the argumentative model, an interesting line of research would be to integrate formal logic rules regarding rhetorical devices [4] and properties of the arguments (such as cogency, effectiveness, and reasonableness). This may hopefully improve the performance of the models and provide a new perspective of analysis. Furthermore, it would be interesting to use such rules to extend the use of LTN to the task of argument quality assessment and further investigate the relationship between theory-based [139] and practical approaches [97].

Although we have experimentally assessed its importance, we have not used neural attention in our neural-symbolic approach to AM, due to the limitations given by the computational resources available to us. However, as future work, we aim to integrate it into our architectures. Additionally, we would like to map its outputs or parameters into logic predicates, so as to exploit it in logic queries that may be used to interpret the behavior of the networks.

It is undeniable that bridging the gap between computational argumentation and neural attention is still a highly challenging task. We recognize that the work covered in this thesis on neural-symbolic approach has been limited to the integration of a few logic rules and lacked the inclusion of proper argumentation frameworks. However, we still believe our research can be considered an important first step towards this, and we eagerly look forward to our future work in this direction.

Acknowledgments

Alla terza tesi comincia a essere difficile riuscire a trovare qualcosa di nuovo da dire, ma cercherò di fare del mio meglio. Se non siete soddisfatti, potete sempre andare a rileggere le precedenti e lamentarvi di quanto gli originali siano sempre meglio dei sequel.

Prima di tutto, grazie alla mia famiglia. Babbo non dovrebbe avvicinarsi a meno di 5 metri da un computer. Mamma è in grado di complicare anche le cose più semplici. Anna...oddio lei in realtà sembra essere diventata un essere umano piuttosto decente in effetti. Nonostante questo, suonerà banale, se sono qui è anche perché avete avuto fiducia in quello che volevo fare e mi avete sempre supportato. Quindi grazie, vi voglio bene.

Grazie a tutte le altre persone con cui ho lavorato questi anni, mostrandomi un posto di lavoro fatto di rispetto, competenza e non troppa seriosità, cosa che so non essere scontata, tantopiù in ambito accademico. A Paola e Federico (quello importante), che da subito mi hanno accolto con gentilezza, disponibilità e allegria. A Daniela, per la sua pazienza infinita e aver a cercato di insegnarmi come si lavora con altri esseri umano. Ad Allegra e tutte le volte che mi ha scritto cavolate e ho dovuto sopportarla (e per quell'una o due volte che forse è possibile sia successo che sia stato io ad ammorbare lei) . A Michela, Michele, Andrea e tutte le altre persone che in un modo o nell'altro mi hanno sopportato e supportato in questi anni.

Grazie a Paolo e Marco. Perché ho bisogno della loro firma per dottorarmi e se non li ringrazio poi si offendono. Ma anche perché non penso avrei potuto chiedere supervisori migliori. Grazie per la vostra pazienza, la vostra tranquillità e la vostra disponibilità. Un po' meno grazie a Paolo e alle sue battute atroci. Grazie a Federico (quello un po' meno importante), compagno di decine di scleri spesi su codici, grafi, modelli e formule. Grazie per essere stato il mio contatto di emergenza ogni volta che un codice brutto e cattivo mi maltrattava. Ora che sto per finire il dottorato...non penso questa cosa cambierà in alcun modo!

Thanks to Ana. You have been one of the most important persons in my life. And you still are. Thanks for all the support you have given me, the time spent together, the rants, the silly jokes, the amazing memes, and also those few times you have scolded me. I really don't think I can add anything more to what you have already said me in your thanks, so I will just answer "Likewise".

Thanks to Demet. My dear friend on the other side of the world, whose name I still struggle to pronounce correctly. Thanks for that brief but wonderful surreal period. And thanks for these years of conversations at improbable times of the day. Whether it is space, politics, philosophy, or face creams, it is, and I think it will always be, a real pleasure to spend time with you.

Grazie a Marta. Per avermi rifiutato 10 anni fa, perché se no come cacchio ci sopportavamo tutto questo tempo? Grazie per i pipponi e le pesatone, grazie per le gag e le cose estremamente stupide. Non penso di avere molto da dire qui, direi che sai già tutto. Abbiamo ancora tempo, ma prima o poi ci toccherà studiare come si gioca a Bridge.

Grazie a Tommy. Per tutte le ore spese assieme, muovendo pedine fisiche o digitali. Per ricordarmi ogni giorno che sono povero, nella vita reale quanto nei giochi da tavolo. Per avermi adottato a Londra, e non avermi cacciato quando ho distrutto Città del Messico. Due volte. E anche perché gli toccherà studiere il manuale di Bridge e spiegarlo.

Grazie a Pier. Che ormai non elemosina più i miei appunti e devo ammetterlo, un po' mi manca. Adesso che i ruoli sono invertiti, che tu mi dici le cose burocratiche, mentre io chiedo consigli di vita e sopravvivenza, grazie per aver fatto la mia stessa forse non troppo furba scelta di carriera, e quindi grazie avermi fatto sentire decisamente meno solo in questi anni.

Grazie a Nic, Zano e Piwa. Per le ore spese a cercare di riconoscere personaggi con gli occhi a mandorla e per i minuti spesi a rovinare personaggi bellissimi con nomi orribili. Grazie per le battute, le cattiverie, le gag improbabili e impossibili da spiegare.

Grazie a Ilaria, che se non la ringraziassi mi terrebbero il muso a vita. E avrebbe un po' ragione. Grazie a per non avermi rotto qualcosa quando probabilmente me lo sarei meritato, e sopportarmi ancora nonostante tutto. Grazie per le cose buffe e le cose strane che fai entrare nella mia vita, che di solito sono spesso anche cose molto belle.

Grazie all'NWO-CB, che in questi anni ha esteso la sua presa in ogni parte dell'Italia, ma senza mai distaccarsi dalla sede centrale, continuando a ricordare e onorare le sue origini. E grazie agli sbattitori e alle sbattitrici e alle sbattitricesse, probilmente i miei amici di più vecchia data. Grazie ai miei compagni di bevute, per tutte le birre condivise, le discussioni sterili a orari improbabili, i tormentoni che non stancano mai e, più in generale, tutto il tempo speso assieme.

Grazie a Mandi, che tecnicamente rientra sia negli AST che nei compagni di bevute, ma visto che l'ultima volta mi sono dimenticato di invitarla alla laurea mi devo fare perdonare. Leggi sopra, leggi sotto, aggiungi il grazie per le ore spese a sentire le lamentele e gli sbuffi reciproci, ma anche il tempo passato a ridere e giocare.

Grazie a Migliucci e a tutti AST, presenti e passati, più vicini e più lontani. Per tutti questi anni di risate e stupidate, dentro e fuori dal palco. Per avermi mostrato quanto poteva essere bello il teatro e per essere stati capaci di farmi rinnovare questa passione di anno in anno,

riempiendoci a vicenda la vita di personaggi e scene che a volte sembra di ricordare meglio di quelli che sono realmente esistiti.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. *CoRR*, abs/1603.04467, 2016. URL http://arxiv.org/abs/1603.04467.

[2] H. M. Adorf and M. D. Johnston. A discrete stochastic neural network algorithm for constraint satisfaction problems. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 917–924 vol.3, June 1990. doi:10.1109/IJCNN.1990.137951.

[3] Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68. Association for Computational Linguistics, 2014. doi:10.3115/v1/W14-2109. URL http://aclweb.org/anthology/W14-2109.

[4] Khalid Al Khatib, Viorel Morari, and Benno Stein. Style analysis of argumentative texts by mining rhetorical devices. In *Proceedings of the 7th Workshop on Argument Mining*, pages 106–116, Online, December 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.argmining-1.12.

[5] Louis Victor Allis et al. *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen, 1994.

[6] Ron Artstein and Massimo Poesio. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008. doi:10.1162/coli.07-034-R2. URL https://www.aclweb.org/anthology/J08-4004.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[8] Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. Deriving machine attention from human rationales. In *EMNLP*, pages 1903–1913. ACL, 2018. doi:10.18653/v1/D18-1216.

[9] Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. Training deeper neural machine translation models with transparent attention. In *EMNLP*, pages 3028–3033. ACL, 2018. doi:10.18653/v1/D18-1338.

[10] Francesco Barbieri, Luis Espinosa Anke, Jose Camacho-Collados, Steven Schockaert, and Horacio Saggion. Interpretable emoji prediction via label-wise attention lstms. In *EMNLP*, pages 4766–4771. ACL, 2018. doi:10.18653/v1/D18-1508.

[11] Robert Charles Bell. *Board and table games from many civilizations*, volume 1. Courier Corporation, 1979.

[12] Elena Bellodi and Fabrizio Riguzzi. Expectation maximization over binary decision diagrams for probabilistic logic programs. *Intelligent Data Analysis*, 17(2):343–363, 2013.

[13] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *EMNLP/IJCNLP (1)*, pages 3613–3618. Association for Computational Linguistics, 2019. doi:10.18653/v1/D19-1371. URL https://doi.org/10.18653/v1/D19-1371.

[14] T.J.M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artificial intelligence*, 171(10-15):619–641, 2007. ISSN 0004-3702. doi:10.1016/j.artint.2007.05.001. URL http://www.sciencedirect.com/science/article/pii/S0004370207000793.

[15] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227. doi:10.1109/72.279181.

[16] Merrie Bergmann. *An introduction to many-valued and fuzzy logic: semantics, algebras, and derivation systems*. Cambridge University Press, 2008.

[17] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, May 2001. ISSN 0004-3702. doi:10.1016/S0004-3702(01)00071-6. URL http://dx.doi.org/10.1016/S0004-3702(01)00071-6.

[18] Philippe Besnard, Alejandro Javier García, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Ricardo Simari, and Francesca Toni. Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4, 2014. doi:10.1080/19462166.2013.869764. URL http://dx.doi.org/10.1080/19462166.2013.869764.

[19] Johannes Bjerva, Barbara Plank, and Johan Bos. Semantic tagging with deep residual networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3531–3541. The COLING 2016 Organizing Committee, 2016. URL http://aclweb.org/anthology/C16-1333.

[20] Yngvi Björnsson. Learning rules of simplified boardgames by observing. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 175–180. IOS Press, 2012.

[21] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *ICML*, pages 1613–1622. JMLR.org, 2015.

[22] Tom Bosc, Elena Cabrio, and Serena Villata. Tweeties squabbling: Positive and negative results in applying argument mining on social media. In Pietro Baroni, Thomas F. Gordon, Tatjana Scheffler, and Manfred Stede, editors, *COMMA 2016*, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 21–32. IOS Press, 2016. doi:10.3233/978-1-61499-686-6-21. URL https://doi.org/10.3233/978-1-61499-686-6-21.

[23] A. Bouhouch, L. Chakir, and A. El Qadi. Scheduling meeting solved by neural network and min-conflict heuristic. In *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, pages 773–778, Oct 2016. doi:10.1109/CIST.2016.7804991.

[24] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996. doi:10.1007/BF00058655. URL https://doi.org/10.1007/BF00058655.

[25] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. In *EMNLP*, pages 1442–1451. ACL, 2017. doi:10.18653/v1/D17-1151.

[26] Elena Cabrio and Serena Villata. Combining textual entailment and argumentation theory for supporting online debates interactions. In *ACL (2)*, ACL '12, pages 208–212, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P12-2041/.

[27] Elena Cabrio and Serena Villata. Five years of argument mining: a data-driven analysis. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5427–5433. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/766. URL https://doi.org/10.24963/ijcai.2018/766.

[28] T. Cazenave. Residual networks for computer go. *IEEE Transactions on Games*, 10(1): 107–110, March 2018. ISSN 2475-1502. doi:10.1109/TCIAIG.2017.2681042.

[29] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-2029. URL https://www.aclweb.org/anthology/D18-2029.

[30] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *IEEE ICASSP*, pages 4960–4964, 2016. doi:10.1109/ICASSP.2016.7472621.

[31] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009. doi:10.1145/1541880.1541882. URL http://doi.acm.org/10.1145/1541880.1541882.

[32] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, Sep 2012. ISSN 1573-0565. doi:10.1007/s10994-012-5296-5. URL https://doi.org/10.1007/s10994-012-5296-5.

[33] K. Chellapilla and D. B. Fogel. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE Transactions on Neural Networks*, 10(6):1382–1391, Nov 1999. ISSN 1045-9227. doi:10.1109/72.809083.

[34] Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. Syntax-directed attention for neural machine translation. In *AAAI*, pages 4792–4799. AAAI Press, 2018.

[35] Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. Enhancing sentence embedding with generalized pooling. In *COLING*, pages 1815–1826. ACL, 2018.

[36] Federico Chesani, Andrea Galassi, Marco Lippi, and Paola Mello. Can deep networks learn to play by the rules? A case study on nine men's morris. *IEEE Transactions on Games*, 10(4):344–353, 2018. doi:10.1109/TG.2018.2804039. URL https://doi.org/10.1109/TG.2018.2804039.

[37] Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Multi-focus attention network for efficient deep reinforcement learning. In *AAAI Workshops*, 2017.

[38] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based Recurrent NN: First results. In *Deep Learning and Representation Learning Workshop, NIPS 2014*, 2014.

[39] Christopher Clark and Amos J. Storkey. Training deep convolutional neural networks to play go. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1766–1774. JMLR.org, 2015.

[40] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-4828. URL https://www.aclweb.org/anthology/W19-4828.

[41] Oana Cocarascu and Francesca Toni. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1374–1379, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1144.

[42] Oana Cocarascu and Francesca Toni. Combining deep learning and argumentative reasoning for the analysis of social media textual content using small data sets. *Computational Linguistics*, 44(4):833–858, 2018. doi:10.1162/coli_a_00338. URL https://doi.org/10.1162/coli_a_00338.

[43] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi:10.1177/001316446002000104. URL https://doi.org/10.1177/001316446002000104.

[44] Charles J Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25–30, 1984.

[45] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011. URL http://www.jmlr.org/papers/v12/collobert11a.html.

[46] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/E17-1104.

[47] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. In *ACL (1)*, pages 593–602. ACL, 2017. doi:10.18653/v1/P17-1055.

[48] Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. Frustratingly short attention spans in neural language modeling. In *ICLR*, 2017.

[49] Omid E. David, Nathan S. Netanyahu, and Lior Wolf. Deepchess: End-to-end deep neural network for automatic learning in chess. In Alessandro E. P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *ICANN (2)*, volume 9887, pages 88–96. Springer, 2016. doi:10.1007/978-3-319-44781-0_11. URL https://doi.org/10.1007/978-3-319-44781-0_11.

[50] Artur S. d'Avila Garcez, Tarek R. Besold, Luc De Raedt, Peter Földiák, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luís C. Lamb, Risto Miikkulainen, and Daniel L. Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *AAAI Spring Symposia*, pages 18–21, 2015. URL https://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10281.

[51] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled

integration of machine learning and reasoning. *FLAP*, 6(4):611–632, 2019. URL https://collegepublications.co.uk/ifcolog/?00033.

[52] Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. What is the essence of a claim? cross-domain claim identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2066. Association for Computational Linguistics, 2017. doi:10.18653/v1/D17-1218. URL http://aclweb.org/anthology/D17-1218.

[53] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016. doi:10.2200/S00692ED1V01Y201601AIM032.

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. ACL, 2019.

[55] Sander Dieleman et al. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878.

[56] Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017. ISSN 0004-3702. doi:https://doi.org/10.1016/j.artint.2015.08.011. URL https://www.sciencedirect.com/science/article/pii/S0004370215001344. Combining Constraint Solving with Mining and Learning.

[57] John Dinsmore. *The symbolic and connectionist paradigms: closing the gap*. Lawrence Erlbaum, 2014.

[58] Tobias Domhan. How much attention do you need? a granular analysis of neural machine translation architectures. In *ACL (1)*, pages 1799–1808. ACL, 2018.

[59] I. Donadello and L. Serafini. Compensating supervision incompleteness with prior knowledge in semantic image interpretation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi:10.1109/IJCNN.2019.8852413.

[60] Ivan Donadello, Luciano Serafini, and Artur D'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI*, page 1596–1602. AAAI Press, 2017. ISBN 9780999241103. doi:10.24963/ijcai.2017/221. URL https://doi.org/10.24963/ijcai.2017/221.

[61] Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016.

[62] J. Du, L. Gui, Y. He, R. Xu, and X. Wang. Convolution-based neural attention with applications to sentiment classification. *IEEE Access*, 7:27983–27992, 2019. doi:10.1109/ACCESS.2019.2900335.

[63] Jinhua Du, Jingguang Han, Andy Way, and Dadong Wan. Multi-level structured self-attentions for distantly supervised relation extraction. In *EMNLP*, pages 2216–2225. ACL, 2018. doi:10.18653/v1/D18-1245.

[64] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995. doi:10.1016/0004-3702(94)00041-X. URL http://www.sciencedirect.com/science/article/pii/000437029400041X.

[65] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Assumption-based argumentation. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 199–218. Springer US, Boston, MA, 2009. ISBN 978-0-387-98197-0. doi:10.1007/978-0-387-98197-0_10. URL https://doi.org/10.1007/978-0-387-98197-0_10.

[66] Mihai Dusmanu, Elena Cabrio, and Serena Villata. Argument mining on Twitter: Arguments, facts and sources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2317–2322, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1245. URL https://www.aclweb.org/anthology/D17-1245.

[67] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *ACL (1)*, pages 334–343. Association for Computational Linguistics, 2015. doi:10.3115/v1/P15-1033. URL http://aclweb.org/anthology/P15-1033.

[68] Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. On the role of discourse markers for discriminating claims and premises in argumentative discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242. Association for Computational Linguistics, 2015. doi:10.18653/v1/D15-1267. URL http://aclweb.org/anthology/D15-1267.

[69] Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22. Association for Computational Linguistics, 2017. doi:10.18653/v1/P17-1002. URL http://aclweb.org/anthology/P17-1002.

[70] Liat Ein-Dor, Eyal Shnarch, Lena Dankin, Alon Halfon, Benjamin Sznajder, Ariel Gera, Carlos Alzate, Martin Gleize, Leshem Choshen, Yufang Hou, Yonatan Bilu, Ranit Aharonov, and Noam Slonim. Corpus wide argument mining - A working solution. In *AAAI*, pages 7683–7691. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6270.

[71] Jonas K. Falkner and Lars Schmidt-Thieme. Learning to solve vehicle routing problems with time windows through joint attention. *CoRR*, abs/2006.09100, 2020. URL https://arxiv.org/abs/2006.09100.

[72] Feifan Fan, Yansong Feng, and Dongyan Zhao. Multi-grained attention network for aspect-level sentiment classification. In *EMNLP*, pages 3433–3442. ACL, 2018. doi:10.18653/v1/D18-1380.

[73] Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. Learning what data to learn. *CoRR*, abs/1702.08635, 2017.

[74] Beatriz Fisas, Francesco Ronzano, and Horacio Saggion. A multi-layered annotated corpus of scientific papers. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *LREC*, Paris, France, may 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1. URL http://www.lrec-conf.org/proceedings/lrec2016/summaries/174.html.

[75] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[76] A. Galassi, M. Lippi, and P. Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–18, 2020. doi:10.1109/TNNLS.2020.3019893. URL https://doi.org/10.1109/TNNLS.2020.3019893.

[77] Andrea Galassi, Marco Lippi, and Paolo Torroni. Argumentative link prediction using residual networks and multi-objective learning. In Noam Slonim and Ranit Aharonov, editors, *Proceedings of the 5th Workshop on Argument Mining, ArgMining@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 1–10. Association for Computational Linguistics, 2018. doi:10.18653/v1/w18-5201. URL https://doi.org/10.18653/v1/w18-5201.

[78] Andrea Galassi, Michele Lombardi, Paola Mello, and Michela Milano. Model agnostic solution of CSPs via deep learning: A preliminary study. In Willem Jan van Hoeve, editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research, CPAIOR 2018*, volume 10848 of *Lecture Notes in Computer Science*, pages 254–262. Springer, 2018. doi:10.1007/978-3-319-93031-2_18. URL https://doi.org/10.1007/978-3-319-93031-2_18.

[79] Andrea Galassi, Kristian Kersting, Marco Lippi, Xiaoting Shao, and Paolo Torroni. Neural-symbolic argumentation mining: An argument in favor of deep learning and reasoning. *Frontiers in Big Data*, 2:52, 2019. doi:10.3389/fdata.2019.00052. URL https://doi.org/10.3389/fdata.2019.00052.

[80] Andrea Galassi, Marco Lippi, and Paolo Torroni. Multi-task attentive residual networks for argument mining. *CoRR*, abs/2102.12227, 2021. URL https://arxiv.org/abs/2102.12227.

[81] Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11: 2001–2049, 2010. URL http://www.jmlr.org/papers/volume11/ganchev10a/ganchev10a.pdf.

[82] Artur S d'Avila Garcez, Krysia B Broda, and Dov M Gabbay. *Neural-symbolic learning systems: foundations and applications*. Springer, 2012.

[83] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory Pract. Log. Program.*, 4(2):95–138, January 2004. ISSN 1471-0684. doi:10.1017/S1471068403001674. URL http://dx.doi.org/10.1017/S1471068403001674.

[84] Ralph Gasser. Solving nine men's morris. *Computational Intelligence*, 12(1):24–41, 1996.

[85] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170, 2018. doi:10.1613/jair.5477.

[86] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI Magazine*, 26(2):62, Jun. 2005. doi:10.1609/aimag.v26i2.1813. URL https://www.aaai.org/ojs/index.php/aimagazine/article/view/1813.

[87] Ian P Gent, Christopher Jefferson, and Peter Nightingale. Complexity of n-queens completion. *Journal of Artificial Intelligence Research*, 59:815–848, 2017.

[88] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*, volume 1. MIT press Cambridge, 2007.

[89] Gábor Etele Gévay and Gábor Danner. Calculating ultrastrong and extended solutions for Nine Men's Morris, Morabaraba, and Lasker Morris. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):256–267, Sept 2016. ISSN 1943-068X. doi:10.1109/TCIAIG.2015.2420191.

[90] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/glorot11a.html.

[91] Yoav Goldberg. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017. doi:10.2200/S00762ED1V01Y201703HLT037. URL https://doi.org/10.2200/S00762ED1V01Y201703HLT037.

[92] Carla P. Gomes, Bart Selman, and Nuno Crato. Heavy-tailed distributions in combinatorial search. In Gert Smolka, editor, *Principles and Practice of Constraint Programming - CP97, Third International Conference, Linz, Austria, October 29 - November 1, 1997, Proceedings*, volume 1330 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 1997. doi:10.1007/BFb0017434. URL https://doi.org/10.1007/BFb0017434.

[93] Carla P. Gomes, Bart Selman, and Henry A. Kautz. Boosting combinatorial search through randomization. In Jack Mostow and Chuck Rich, editors, *AAAI/IAAI*, pages 431–437. AAAI Press / The MIT Press, 1998. URL http://www.aaai.org/Library/AAAI/1998/aaai98-061.php.

[94] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org.

[95] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[96] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471. JMLR.org, 2015.

[97] Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. A large-scale dataset for argument quality ranking: Construction and analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7805–7813, Apr. 2020. doi:10.1609/aaai.v34i05.6285. URL https://ojs.aaai.org/index.php/AAAI/article/view/6285.

[98] Chinnappa Guggilla, Tristan Miller, and Iryna Gurevych. Cnn- and lstm-based claim classification in online user comments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2740–2751. The COLING 2016 Organizing Committee, 2016. URL http://aclweb.org/anthology/C16-1258.

[99] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, August 2018. ISSN 0360-0300. doi:10.1145/3236009.

[100] Bernd Gutmann, Ingo Thon, and Luc De Raedt. Learning the parameters of probabilistic logic programs from interpretations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 581–596. Springer, 2011.

[101] Ivan Habernal and Iryna Gurevych. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *EMNLP*, pages 1214–1223. ACL, 2016. doi:10.18653/v1/D16-1129.

[102] Ivan Habernal and Iryna Gurevych. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics, 2016. doi:10.18653/v1/P16-1150. URL http://aclweb.org/anthology/P16-1150.

[103] Ivan Habernal and Iryna Gurevych. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179, 2017. doi:10.1162/COLI_a_-00276. URL http://aclweb.org/anthology/J17-1004.

[104] Shohreh Haddadan, Elena Cabrio, and Serena Villata. Yes, we can! mining arguments in 50 years of US presidential campaign debates. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4684–4690, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1463. URL https://www.aclweb.org/anthology/P19-1463.

[105] Richard W Hamming. *Coding and Theory*. Prentice-Hall Englewood Cliffs, 1980.

[106] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. doi:10.1109/ICCV.2015.123.

[107] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[108] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

[109] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. An unsupervised neural attention model for aspect extraction. In *ACL (1)*, volume 1, pages 388–397, 2017.

[110] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. Effective attention modeling for aspect-level sentiment classification. In *COLING*, pages 1121–1131. ACL, 2018.

[111] Jay Heo, Hae Beom Lee, Saehoon Kim, Juho Lee, Kwang Joon Kim, Eunho Yang, and Sung Ju Hwang. Uncertainty-aware attention for reliable interpretation and prediction. In *NeurIPS*, pages 917–926, USA, 2018. Curran Associates Inc.

[112] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[113] André Hottung, Shunji Tanaka, and Kevin Tierney. Deep learning assisted heuristic tree search for the container pre-marshalling problem. *Comput. Oper. Res.*, 113, 2020. doi:10.1016/j.cor.2019.104781. URL https://doi.org/10.1016/j.cor.2019.104781.

[114] George Hripcsak and Adam S. Rothschild. Agreement, the F-Measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, 05 2005. ISSN 1067-5027. doi:10.1197/jamia.M1733. URL https://doi.org/10.1197/jamia.M1733.

[115] Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. Attention-guided answer distillation for machine reading comprehension. In *EMNLP*, pages 2077–2086. ACL, 2018.

[116] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In *ACL (1)*, pages 2410–2420, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1228. URL https://www.aclweb.org/anthology/P16-1228.

[117] Xinyu Hua, Mitko Nikolov, Nikhil Badugu, and Lu Wang. Argument mining for understanding peer reviews. In *NAACL-HLT (1)*, pages 2131–2137, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1219. URL https://www.aclweb.org/anthology/N19-1219.

[118] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

[119] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017. doi:10.1109/CVPR.2017.243. URL https://doi.org/10.1109/CVPR.2017.243.

[120] YiYao Huang and William Yang Wang. Deep residual learning for weakly-supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1803–1807. Association for Computational Linguistics, 2017. doi:10.18653/v1/D17-1191. URL http://aclweb.org/anthology/D17-1191.

[121] Drew A. Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. In *ICLR*, 2018.

[122] Ching-Lai Hwang and Kwangsun Yoon. *Multiple attribute decision making: methods and applications a state-of-the-art survey*, volume 186. Springer Science & Business Media, 2012.

[123] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/ioffe15.html.

[124] Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *NAACL-HLT (1)*, pages 3543–3556. ACL, 2019.

[125] Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *ACL (1)*, pages 908–918. ACL, 2016.

[126] Guoliang Kang, Liang Zheng, Yan Yan, and Yi Yang. Deep adversarial attention alignment for unsupervised domain adaptation: the benefit of target expectation maximization. In *ECCV*, pages 401–416, 2018.

[127] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE CVPR*, 2018.

[128] Joo-Kyung Kim and Young-Bum Kim. Supervised domain enablement attention for personalized domain classification. In *EMNLP*, pages 894–899. ACL, 2018. doi:10.18653/v1/D18-1106.

[129] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *ICLR*, 2017.

[130] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[131] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016. URL http://aclweb.org/anthology/Q16-1023.

[132] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2019.

[133] Parisa Kordjamshidi, Dan Roth, and Kristian Kersting. Systems AI: A declarative learning based programming perspective. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5464–5471. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/771. URL https://doi.org/10.24963/ijcai.2018/771.

[134] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage publications, 2018.

[135] Matthew Lai. Giraffe: Using deep reinforcement learning to play chess. *CoRR*, abs/1509.01549, 2015. URL http://arxiv.org/abs/1509.01549.

[136] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *NIPS*, pages 1243–1251, 2010.

[137] Anne Lauscher, Goran Glavaš, and Simone Paolo Ponzetto. An argument-annotated corpus of scientific publications. In *Proceedings of the 5th Workshop on Argument Mining*, pages 40–46, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi:10.18653/v1/W18-5206. URL https://www.aclweb.org/anthology/W18-5206.

[138] Anne Lauscher, Goran Glavaš, Simone Paolo Ponzetto, and Kai Eckert. Investigating the role of argumentation in the rhetorical analysis of scientific publications with neural multi-task learning models. In *EMNLP*, pages 3326–3338. ACL, 2018. doi:10.18653/v1/D18-1370.

[139] Anne Lauscher, Lily Ng, Courtney Napoles, and Joel Tetreault. Rhetoric, logic, and dialectic: Advancing theory-based argument quality assessment in natural language processing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4563–4574, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. URL https://www.aclweb.org/anthology/2020.coling-main.402.

[140] John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818, 2020. doi:10.1162/coli_a_00364. URL https://doi.org/10.1162/coli_a_00364.

[141] Quoc V Le. Building high-level features using large scale unsupervised learning. In *IEEE ICASSP*, pages 8595–8598. IEEE, 2013.

[142] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[143] J. H. M. Lee, H. F. Leung, and H. W. Won. Extending genet for non-binary csp's. In *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 338–343, Nov 1995. doi:10.1109/TAI.1995.479651.

[144] Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *EMNLP (System Demonstrations)*, pages 121–126. ACL, 2017. doi:10.18653/v1/D17-2021.

[145] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240, 2020. doi:10.1093/bioinformatics/btz682. URL https://doi.org/10.1093/bioinformatics/btz682.

[146] John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *ACM Trans. Knowl. Discov. Data*, 13(6):62:1–62:25, November 2019. ISSN 1556-4681. doi:10.1145/3363574.

[147] Gaël Letarte, Frédérik Paradis, Philippe Giguère, and François Laviolette. Importance of self-attention for sentiment analysis. In *BlackboxNLP@EMNLP*, pages 267–275. ACL, 2018.

[148] Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. Context dependent claim detection. In Jan Hajic and Junichi Tsujii, editors, *COLING 2014, Dublin, Ireland*, pages 1489–1500. ACL, 2014. URL http://www.aclweb.org/anthology/C14-1141.

[149] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. In *ACL (1)*, pages 292–302. ACL, 2019. doi:10.18653/v1/P19-1028.

[150] Xiangju Li, Kaisong Song, Shi Feng, Daling Wang, and Yifei Zhang. A co-attention neural network model for emotion cause analysis with emotional context awareness. In *EMNLP*, pages 4752–4757. ACL, 2018. doi:10.18653/v1/D18-1506.

[151] Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. Hierarchical attention transfer network for cross-domain sentiment classification. In *AAAI*, pages 5852–5859. AAAI Press, 2018.

[152] Jian-Fu Lin, Kuo Yu Huang, Hen-Hsen Huang, and Hsin-Hsi Chen. Lexicon guided attentive neural network model for argument mining. In *Proceedings of the 6th Workshop on Argument Mining*, pages 67–73, Florence, Italy, August 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-4508. URL https://www.aclweb.org/anthology/W19-4508.

[153] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[154] Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. A pdtb-styled end-to-end discourse parser. *Nat. Lang. Eng.*, 20(2):151–184, 2014. doi:10.1017/S1351324912000307. URL https://doi.org/10.1017/S1351324912000307.

[155] Marco Lippi. Reasoning with deep learning: an open challenge. In *CEUR Workshop Proceedings*, volume 1802, pages 38–43. CEUR-WS, 2017.

[156] Marco Lippi and Paolo Frasconi. Prediction of protein β-residue contacts by markov logic networks with grounding-specific weights. *Bioinformatics*, 25(18):2326–2333, 2009. ISSN 1367-4803. doi:10.1093/bioinformatics/btp421. URL https://dx.doi.org/10.1093/bioinformatics/btp421.

[157] Marco Lippi and Paolo Torroni. Context-independent claim detection for argument mining. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 185–191. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL http://ijcai.org/papers15/Abstracts/IJCAI15-033.html.

[158] Marco Lippi and Paolo Torroni. Margot. *Expert Syst. Appl.*, 65(C):292–303, December 2016. ISSN 0957-4174. doi:10.1016/j.eswa.2016.08.050.

[159] Marco Lippi and Paolo Torroni. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Technol.*, 16(2):10:1–10:25, March 2016. ISSN 1533-5399. doi:10.1145/2850417. URL http://doi.acm.org/10.1145/2850417.

[160] Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. Neural machine translation with supervised attention. In *COLING*, pages 3093–3102. ACL, 2016.

[161] Shusen Liu, Tao Li, Zhimin Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. Visual interrogation of attention-based models for natural language inference

and machine comprehension. In *EMNLP (System Demonstrations)*, pages 36–41. ACL, 2018. doi:10.18653/v1/D18-2007.

[162] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

[163] Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artif. Intell.*, 244:343–367, 2017. doi:10.1016/j.artint.2016.01.005. URL https://doi.org/10.1016/j.artint.2016.01.005.

[164] Y. Long, R. Xiang, Q. Lu, C. Huang, and M. Li. Improving attention model based on cognition grounded data for sentiment analysis. *IEEE Trans. Affective Comput.*, pages 1–1, 2019. doi:10.1109/TAFFC.2019.2903056.

[165] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, pages 289–297. Curran Associates, Inc., 2016.

[166] Luca Lugini and Diane Litman. Argument component classification for classroom discussions. In *Proceedings of the 5th Workshop on Argument Mining*, pages 57–67. Association for Computational Linguistics, 2018. doi:10.18653/v1/W18-5208. URL http://aclweb.org/anthology/W18-5208.

[167] Luca Lugini and Diane Litman. Contextual argument component classification for class discussions. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1475–1480, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. URL https://www.aclweb.org/anthology/2020.coling-main.128.

[168] Luca Lugini, Christopher Olshefski, Ravneet Singh, Diane Litman, and Amanda Godley. Discussion tracker: Supporting teacher learning about students' collaborative argumentation in high school classrooms. In *Proceedings of the 28th International Conference on Computational Linguistics: System Demonstrations*, pages 53–58, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics (ICCL). URL https://www.aclweb.org/anthology/2020.coling-demos.10.

[169] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025v5, 2015.

[170] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive attention networks for aspect-level sentiment classification. In *IJCAI*, pages 4068–4074. IJCAI.org, 2017. doi:10.24963/ijcai.2017/568.

[171] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics, 2016. doi:10.18653/v1/P16-1101. URL http://aclweb.org/anthology/P16-1101.

[172] Yukun Ma, Haiyun Peng, and Erik Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *AAAI*, pages 5876–5883. AAAI Press, 2018.

[173] Suraj Maharjan, Manuel Montes, Fabio A. González, and Thamar Solorio. A genre-aware attention model to improve the likability prediction of books. In *EMNLP*, pages 3381–3391. ACL, 2018. doi:10.18653/v1/D18-1375.

[174] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Adolfo Velasco-Hernández, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In Kohei Arai and Supriya Kapoor, editors, *Advances in Computer Vision - Proceedings of the 2019 Computer Vision Conference, CVC 2019, Las Vegas, Nevada, USA, 25-26 April 2019, Volume 1*, volume 943 of *Advances in Intelligent Systems and Computing*, pages 128–144. Springer, 2019. doi:10.1007/978-3-030-17795-9_10. URL https://doi.org/10.1007/978-3-030-17795-9_10.

[175] Chaitanya Malaviya, Pedro Ferreira, and André F. T. Martins. Sparse and constrained attention for neural machine translation. In *ACL (2)*, pages 370–376. ACL, 2018.

[176] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 3753–3763. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7632-deepproblog-neural-probabilistic-logic-programming.pdf.

[177] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, volume 48, pages 1614–1623. JMLR.org, 2016.

[178] André F. T. Martins and Julia Kreutzer. Learning what's easy: Fully differentiable neural easy-first taggers. In *EMNLP*, pages 349–362. ACL, 2017. doi:10.18653/v1/D17-1036.

[179] Yann Mathet, Antoine Widlöcher, Karën Fort, Claire François, Olivier Galibert, Cyril Grouin, Juliette Kahn, Sophie Rosset, and Pierre Zweigenbaum. Manual corpus annotation: Giving meaning to the evaluation metrics. In *Proceedings of COLING 2012: Posters*, pages 809–818, Mumbai, India, December 2012. The COLING 2012 Organizing Committee. URL https://www.aclweb.org/anthology/C12-2079.

[180] Tobias Mayer, Elena Cabrio, Marco Lippi, Paolo Torroni, and Serena Villata. Argument mining on clinical trials. In Sanjay Modgil, Katarzyna Budzynska, and John Lawrence, editors, *Computational Models of Argument - Proceedings of COMMA 2018, Warsaw, Poland, 12-14 September 2018*, volume 305 of *Frontiers in Artificial Intelligence and Applications*, pages 137–148. IOS Press, 2018. doi:10.3233/978-1-61499-906-5-137. URL https://doi.org/10.3233/978-1-61499-906-5-137.

[181] Tobias Mayer, Elena Cabrio, and Serena Villata. Transformer-based argument mining for healthcare applications. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2108–2115. IOS Press, 2020. doi:10.3233/FAIA200334. URL https://doi.org/10.3233/FAIA200334.

[182] Stefano Menini, Elena Cabrio, Sara Tonelli, and Serena Villata. Never retreat, never retract: Argumentation analysis for political speeches. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4889–4896. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16393.

[183] Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. Coverage embedding models for neural machine translation. In *EMNLP*, pages 955–960. ACL, 2016. doi:10.18653/v1/D16-1096.

[184] Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. Supervised attentions for neural machine translation. In *EMNLP*, pages 2283–2288. ACL, 2016. doi:10.18653/v1/D16-1249.

[185] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024, 2019. URL http://papers.nips.cc/paper/9551-are-sixteen-heads-really-better-than-one.

[186] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116. Association for Computational Linguistics, 2016. doi:10.18653/v1/P16-1105. URL http://aclweb.org/anthology/P16-1105.

[187] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, pages 2204–2212. Curran Associates, Inc., 2014.

[188] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

[189] Raquel Mochales Palau and Marie-Francine Moens. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22, 2011. doi:10.1007/s10506-010-9104-x. URL http://dx.doi.org/10.1007/s10506-010-9104-x.

[190] Seungwhan Moon and Jaime G. Carbonell. Completely heterogeneous transfer learning with attention - what and what not to transfer. In *IJCAI*, pages 2508–2514. IJCAI.org, 2017. doi:10.24963/ijcai.2017/349.

[191] Stephen Muggleton, Aline Paes, Vítor Santos Costa, and Gerson Zaverucha. Chess revision: Acquiring the rules of chess variants through fol theory revision from examples. In Luc De Raedt, editor, *Inductive Logic Programming: 19th International Conference, ILP 2009, Leuven, Belgium, July 02-04, 2009. Revised Papers*, pages 123–130, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-13840-9. doi:10.1007/978-3-642-13840-9_12. URL http://dx.doi.org/10.1007/978-3-642-13840-9_12.

[192] Héctor Muñoz-Avila, Christian Bauckhage, Michal Bida, Clare Bates Congdon, and Graham Kendall. Learning and game ai. In *Dagstuhl Follow-Ups*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[193] Israel C. T. Ngoko, Amlan Mukherjee, and Boniface Kabaso. Abstractive text summarization using recurrent neural networks: Systematic literature review. In *ICICKM*, volume 13, pages 435–439, 2018.

[194] Vlad Niculae, Joonsuk Park, and Claire Cardie. Argument mining with structured svms and rnns. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 985–995. Association for Computational Linguistics, 2017. doi:10.18653/v1/P17-1091. URL http://aclweb.org/anthology/P17-1091.

[195] Feng Nie, Yunbo Cao, Jinpeng Wang, Chin-Yew Lin, and Rong Pan. Mention and entity description co-attention for entity disambiguation. In *AAAI*, pages 5908–5915. AAAI Press, 2018.

[196] Joonsuk Park and Claire Cardie. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-2105.

[197] Joonsuk Park and Claire Cardie. A corpus of eRulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L18-1257.

[198] Joonsuk Park, Cheryl Blake, and Claire Cardie. Toward machine-assisted participation in erulemaking: An argumentation model of evaluability. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 206–210, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3522-5. doi:10.1145/2746090.2746118. URL http://doi.acm.org/10.1145/2746090.2746118.

[199] Joonsuk Park, Arzoo Katiyar, and Bishan Yang. Conditional random fields for identifying appropriate types of support for propositions in online user comments. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 39–44. Association for Computational Linguistics, 2015. doi:10.3115/v1/W15-0506. URL http://aclweb.org/anthology/W15-0506.

[200] John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. Deeper attention to abusive user content moderation. In *EMNLP*, pages 1125–1135. ACL, 2017. doi:10.18653/v1/D17-1117.

[201] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014. doi:10.3115/v1/D14-1162. URL http://aclweb.org/anthology/D14-1162.

[202] Isaac Persing and Vincent Ng. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394. Association for Computational Linguistics, 2016. doi:10.18653/v1/N16-1164. URL http://aclweb.org/anthology/N16-1164.

[203] Isaac Persing and Vincent Ng. Unsupervised argumentation mining in student essays. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6795–6803, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.839.

[204] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

[205] M. Poggi and S. Mattoccia. Learning a general-purpose confidence measure based on o(1) features and a smarter aggregation strategy for semi global matching. In *3DV*, pages 509–518, 2016. doi:10.1109/3DV.2016.61.

[206] Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakob Uszkoreit, Ondřej Bojar, and Zdeněk Žabokrtskỳ. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature communications*, 11(1):1–15, 2020. doi:10.1038/s41467-020-18073-9. URL https://doi.org/10.1038/s41467-020-18073-9.

[207] Peter Potash, Alexey Romanov, and Anna Rumshisky. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1364–1373. Association for Computational Linguistics, 2017. doi:10.18653/v1/D17-1143. URL http://aclweb.org/anthology/D17-1143.

[208] Prakash Poudyal, Jaromir Savelka, Aagje Ieven, Marie Francine Moens, Teresa Goncalves, and Paulo Quaresma. ECHR: Legal corpus for argument mining. In *Proceedings of the 7th Workshop on Argument Mining*, pages 67–75, Online, December 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.argmining-1.8.

[209] Tingting Qiao, Jianfeng Dong, and Duanqing Xu. Exploring human-like attention supervision in visual question answering. In *AAAI*, pages 7300–7307. AAAI Press, 2018.

[210] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic prolog and its application in link discovery. In *IJCAI*, pages 2462–2467, 2007. URL https://www.aaai.org/Papers/IJCAI/2007/IJCAI07-397.pdf.

[211] Luc De Raedt, Sebastijan Dumancic, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4943–4950. ijcai.org, 2020. doi:10.24963/ijcai.2020/688. URL https://doi.org/10.24963/ijcai.2020/688.

[212] Iyad Rahwan and Guillermo Simari. *Argumentation in Artificial Intelligence*. Springer US, 2009. ISBN 978-0-387-98196-3. doi:10.1007/978-0-387-98197-0. URL https://www.springer.com/us/book/9780387981963.

[213] Dennis Reidsma and Jean Carletta. Squibs: Reliability measurement without limits. *Computational Linguistics*, 34(3):319–326, 2008. doi:10.1162/coli.2008.34.3.319. URL https://www.aclweb.org/anthology/J08-3001.

[214] Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics, 2017. doi:10.18653/v1/D17-1035. URL http://aclweb.org/anthology/D17-1035.

[215] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1410. URL https://www.aclweb.org/anthology/D19-1410.

[216] Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. Classification and clustering of arguments with contextualized word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 567–578, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1054. URL https://www.aclweb.org/anthology/P19-1054.

[217] J. Reisinger, E. Bahceci, I. Karpov, and R. Miikkulainen. Coevolving strategies for general game playing. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 320–327, April 2007. doi:10.1109/CIG.2007.368115.

[218] Mark O. Riedl. Human-centered artificial intelligence and machine learning. *Human Behavior and Emerging Technologies*, 1:33–36, 2019.

[219] Fabrizio Riguzzi. *Foundations of Probabilistic Logic Programming*. River Publishers, Gistrup, Denmark, 2018. ISBN 9788770220187. URL http://www.riverpublishers.com/book_details.php?book_id=660.

[220] Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450. Association for Computational Linguistics, 2015. doi:10.18653/v1/D15-1050. URL http://aclweb.org/anthology/D15-1050.

[221] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. Reasoning about entailment with neural attention. In *ICLR*, 2016.

[222] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *Nature Machine Intelligence*, 1:206–215, 2019.

[223] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, July 1959. ISSN 0018-8646. doi:10.1147/rd.33.0210. URL http://dx.doi.org/10.1147/rd.33.0210.

[224] Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. Argument extraction from news. In *Proceedings of the Second Workshop on Argumentation Mining*, pages 56–66. Association for Computational Linguistics, 2015. URL http://www.aclweb.org/anthology/W15-0508.

[225] Taisuke Sato and Yoshitaka Kameya. PRISM: A language for symbolic-statistical modeling. In *Proceedings of the Fifteenth International Joint Conference on Artificial*

Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes, pages 1330–1339. Morgan Kaufmann, 1997. URL http://ijcai.org/Proceedings/97-2/Papers/078.pdf.

[226] Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. Multi-task learning for argumentation mining in low-resource settings. In *NAACL-HLT (2)*, pages 35–41. Association for Computational Linguistics, 2018. doi:10.18653/v1/N18-2006. URL http://aclweb.org/anthology/N18-2006.

[227] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008. doi:10.1609/aimag.v29i3.2157. URL https://doi.org/10.1609/aimag.v29i3.2157.

[228] Luciano Serafini and Artur S. d'Avila Garcez. Learning and reasoning with logic tensor networks. In Giovanni Adorni, Stefano Cagnoni, Marco Gori, and Marco Maratea, editors, *AI*IA 2016 Advances in Artificial Intelligence*, pages 334–348, Cham, 2016. Springer International Publishing. ISBN 978-3-319-49130-1.

[229] Luciano Serafini and Artur S. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *CoRR*, abs/1606.04422, 2016. URL http://arxiv.org/abs/1606.04422.

[230] Sofia Serrano and Noah A. Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1282. URL https://www.aclweb.org/anthology/P19-1282.

[231] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, pages 5446–5455. AAAI Press, 2018.

[232] Ying Shen, Yang Deng, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. Knowledge-aware attentive neural network for ranking question answer pairs. In *SIGIR*, pages 901–904. ACM, 2018.

[233] Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural mt learn source syntax? In *EMNLP*, pages 1526–1534. ACL, 2016. doi:10.18653/v1/D16-1159.

[234] Eyal Shnarch, Carlos Alzate, Lena Dankin, Martin Gleize, Yufang Hou, Leshem Choshen, Ranit Aharonov, and Noam Slonim. Will it blend? blending weak and strong labeled data in a neural network for argumentation mining. In *Proceedings of*

the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 599–605. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/P18-2095.

[235] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi:https://doi.org/10.1038/nature16961.

[236] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[237] Mattia Silvestri, Michele Lombardi, and Michela Milano. Injecting domain knowledge in neural networks: a controlled experiment on a constrained problem. In Alessandro Saffiotti, Luciano Serafini, and Paul Lukowicz, editors, *Proceedings of the First International Workshop on New Foundations for Human-Centered AI (NeHuAI) @ ECAI*, volume 2659 of *CEUR Workshop Proceedings*, pages 52–58. CEUR-WS.org, 2020. URL http://ceur-ws.org/Vol-2659/silvestri.pdf.

[238] Noam Slonim, Yonatan Bilu, Carlos Alzate, Roy Bar-Haim, Ben Bogin, Francesca Bonin, Leshem Choshen, Edo Cohen-Karlik, Lena Dankin, Lilach Edelstein, Liat Ein-Dor, Roni Friedman-Melamed, Assaf Gavron, Ariel Gera, Martin Gleize, Shai Gretz, Dan Gutfreund, Alon Halfon, Daniel Hershcovich, Ron Hoory, Yufang Hou, Shay Hummel, Michal Jacovi, Charles Jochim, Yoav Kantor, Yoav Katz, David Konopnicki, Zvi Kons, Lili Kotlerman, Dalia Krieger, Dan Lahav, Tamar Lavee, Ran Levy, Naftali Liberman, Yosi Mass, Amir Menczel, Shachar Mirkin, Guy Moshkowich, Shila Ofek-Koifman, Matan Orbach, Ella Rabinovich, Ruty Rinott, Slava Shechtman, Dafna Sheinwald, Eyal Shnarch, Ilya Shnayderman, Aya Soffer, Artem Spector, Benjamin Sznajder, Assaf Toledo, Orith Toledo-Ronen, Elad Venezian, and Ranit Aharonov. An autonomous debating system. *Nature*, 591(7850):379–384, Mar 2021. ISSN 1476-4687. doi:10.1038/s41586-021-03215-w. URL https://doi.org/10.1038/s41586-021-03215-w.

[239] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 926–934. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/b337e84de8752b27eda3a12363109e80-Paper.pdf.

[240] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL (2)*, pages 231–235. Association for Computational Linguistics, 2016. doi:10.18653/v1/P16-2038. URL http://aclweb.org/anthology/P16-2038.

[241] Huan Song, Deepta Rajan, Jayaraman J. Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *AAAI*, pages 4091–4098. AAAI Press, 2018.

[242] Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245, 2016.

[243] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. Self-attentional acoustic models. In *Interspeech*, pages 3723–3727, 2018. doi:10.21437/Interspeech.2018-1910.

[244] Maximilian Spliethöver, Jonas Klaff, and Hendrik Heuer. Is it worth the attention? a comparative evaluation of attention layers for argument unit segmentation. In *Proceedings of the 6th Workshop on Argument Mining*, pages 74–82, Florence, Italy, August 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-4509. URL https://www.aclweb.org/anthology/W19-4509.

[245] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[246] Christian Stab and Iryna Gurevych. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56. Association for Computational Linguistics, 2014. doi:10.3115/v1/D14-1006. URL http://aclweb.org/anthology/D14-1006.

[247] Christian Stab and Iryna Gurevych. Parsing argumentation structures in persuasive essays. *Comput. Linguistics*, 43(3):619–659, 2017. doi:10.1162/COLI_a_00295. URL https://doi.org/10.1162/COLI_a_00295.

[248] Christian Stab, Tristan Miller, Benjamin Schiller, Pranav Rai, and Iryna Gurevych. Cross-topic argument mining from heterogeneous sources. In *EMNLP*, pages 3664–3674. ACL, 2018. doi:10.18653/v1/D18-1402.

[249] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *EMNLP*, pages 5027–5038. ACL, 2018. doi:10.18653/v1/D18-1548.

[250] Derwin Suhartono, Aryo Pradipta Gema, Suhendro Winton, Theodorus David, Mohamad Ivan Fanany, and Aniati Murni Arymurthy. Argument annotation and analysis using deep learning with attention mechanism in bahasa indonesia. *J. Big Data*, 7(1):90, 2020. doi:10.1186/s40537-020-00364-z. URL https://doi.org/10.1186/s40537-020-00364-z.

[251] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, pages 2440–2448. Curran Associates, Inc., 2015.

[252] Gongbo Tang, Mathias Mueller, Annette Rios, and Rico Sennrich. Why self-attention? a targeted evaluation of neural machine translation architectures. In *EMNLP*, 2018.

[253] Milagro Teruel, Cristian Cardellino, Fernando Cardellino, Laura Alonso Alemany, and Serena Villata. Increasing argument annotation reproducibility by using inter-annotator agreement to improve guidelines. In *LREC*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L18-1640.

[254] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38 (3):58–68, March 1995. ISSN 0001-0782. doi:10.1145/203330.203343. URL http://doi.acm.org/10.1145/203330.203343.

[255] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

[256] Tian Tian and Zheng (Felix) Fang. Attention-based autoencoder topic model for short texts. *Procedia Comput. Sci.*, 151:1134 – 1139, 2019.

[257] Stephen Edelston Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.

[258] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Trans. Neural Netw. Learn. Syst.*, 3(5):1407–1418, May 2019. ISSN 2162-237X. doi:10.1109/TNNLS.2018.2869225.

[259] Dietrich Trautmann, Johannes Daxenberger, Christian Stab, Hinrich Schütze, and Iryna Gurevych. Fine-grained argument unit recognition and classification. In *AAAI*, pages 9048–9056. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6438.

[260] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *ACL (1)*, pages 76–85. ACL, 2016. doi:10.18653/v1/P16-1008.

[261] Wil van der Aalst et al. *Process Mining Manifesto*, pages 169–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-28108-2. doi:10.1007/978-3-642-28108-2_19. URL http://dx.doi.org/10.1007/978-3-642-28108-2_19.

[262] Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. Attention Interpretability Across NLP Tasks. *arXiv e-prints*, art. arXiv:1909.11218, Sep 2019.

[263] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008. Curran Associates, Inc., 2017.

[264] Joost Vennekens, Sofie Verbaeten, and Maurice Bruynooghe. Logic programs with annotated disjunctions. In Bart Demoen and Vladimir Lifschitz, editors, *Logic Programming*, pages 431–445, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-27775-0. doi:10.1007/978-3-540-27775-0_30. URL https://doi.org/10.1007/978-3-540-27775-0_30.

[265] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

[266] Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. Context-aware neural machine translation learns anaphora resolution. In *ACL (1)*, pages 1264–1274. ACL, 2018.

[267] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, pages 5797–5808. ACL, 2019. doi:10.18653/v1/P19-1580.

[268] Douglas Walton, Christopher Reed, and Fabrizio Macagno. *Argumentation Schemes*. Cambridge University Press, 2008. ISBN 9780521723749.

[269] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *ACL (1)*, volume 1, pages 1288–1297, 2016. doi:10.18653/v1/P16-1122.

[270] C. J. Wang and E. P. K. Tsang. Solving constraint satisfaction problems using neural networks. In *1991 Second International Conference on Artificial Neural Networks*, pages 295–299, Nov 1991.

[271] Hao Wang, Zhen Huang, Yong Dou, and Yu Hong. Argumentation mining on essays at multi scales. In *COLING*, pages 5480–5493, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. URL https://www.aclweb.org/anthology/2020.coling-main.478.

[272] Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *AAAI*, volume 33, pages 7152–7159, 2019.

[273] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *AAAI*, pages 2532–2539. AAAI Press, 2018.

[274] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, and Neil Martin Robertson. Deep metric learning by online soft mining and class-aware attention. In *AAAI 2019*, pages 5361–5368. AAAI Press, 2019. doi:10.1609/aaai.v33i01.33015361. URL https://doi.org/10.1609/aaai.v33i01.33015361.

[275] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *EMNLP/IJCNLP (1)*, pages 11–20. ACL, 2019.

[276] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[277] Lijun Wu, Fei Tian, Li Zhao, Jianhuang Lai, and Tie-Yan Liu. Word attention for sequence to sequence text understanding. In *AAAI*, pages 5578–5585. AAAI Press, 2018.

[278] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 37, pages 2048–2057. JMLR.org, 2015.

[279] Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. Modeling localness for self-attention networks. In *EMNLP*, pages 4449–4458. ACL, 2018. doi:10.18653/v1/D18-1475.

[280] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, pages 2319–2328, 2017.

[281] Pengcheng Yang, Junyang Lin, Jingjing Xu, Jun Xie, Qi Su, and SUN Xu. Specificity-driven cascading approach for unsupervised sentiment modification. In *EMNLP-IJCNLP*, pages 5511–5520, 2019.

[282] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489. ACL, 2016. doi:10.18653/v1/N16-1174.

[283] G. N. Yannakakis and J. Togelius. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4): 317–335, Dec 2015. ISSN 1943-068X. doi:10.1109/TCIAIG.2014.2339221.

[284] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention networks. In *IJCAI*, pages 3926–3932. IJCAI.org, 2018. doi:10.24963/ijcai.2018/546.

[285] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.*, 13(3):55–75, August 2018. ISSN 1556-603X. doi:10.1109/MCI.2018.2840738.

[286] Jianfei Yu, Luis Marujo, Jing Jiang, Pradeep Karuturi, and William Brendel. Improving multi-label emotion classification via sentiment classification with dual attention transfer network. In *EMNLP*, pages 1097–1102. ACL, 2018. doi:10.18653/v1/D18-1137.

[287] L. Yu, H. Chen, Q. Dou, J. Qin, and P. A. Heng. Automated melanoma recognition in dermoscopy images via very deep residual networks. *IEEE Transactions on Medical Imaging*, 36(4):994–1004, April 2017. ISSN 0278-0062. doi:10.1109/TMI.2016.2642839.

[288] Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. Multi-attention recurrent network for human communication comprehension. In *AAAI*, pages 5642–5649. AAAI Press, 2018.

[289] Omar Zaidan, Jason Eisner, and Christine Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *HLT-NAACL*, pages 260–267. ACL, 2007.

[290] Biao Zhang, Deyi Xiong, and Jinsong Su. Battrae: Bidimensional attention-based recursive autoencoders for learning bilingual phrase embeddings. In *AAAI*, pages 3372–3378. AAAI Press, 2017.

[291] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, volume 97, pages 7354–7363. JMLR.org, 2019.

[292] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In Satinder P. Singh and Shaul Markovitch, editors, *AAAI*, pages 1655–1661. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14501.

[293] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018. doi:10.1002/widm.1253. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1253.

[294] Minghua Zhang and Yunfang Wu. An unsupervised model with attention autoencoders for question retrieval. In *AAAI*, pages 4978–4986. AAAI Press, 2018.

[295] Qi Zhang, Jinlan Fu, Xiaoyu Liu, and Xuanjing Huang. Adaptive co-attention network for named entity recognition in tweets. In *AAAI*, pages 5674–5681. AAAI Press, 2018.

[296] Ye Zhang, Iain Marshall, and Byron C. Wallace. Rationale-augmented convolutional neural networks for text classification. In *EMNLP*, pages 795–804. ACL, 2016. doi:10.18653/v1/D16-1076.

[297] Yinyuan Zhang, Ricardo Henao, Zhe Gan, Yitong Li, and Lawrence Carin. Multi-label learning from medical plain text with convolutional residual models. In Finale Doshi-Velez, Jim Fackler, Ken Jung, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 3rd Machine Learning for Healthcare Conference*, volume 85 of *Proceedings of Machine Learning Research*, pages 280–294, Palo Alto, California, 17–18 Aug 2018. PMLR. URL http://proceedings.mlr.press/v85/zhang18a.html.

[298] Shenjian Zhao and Zhihua Zhang. Attention-via-attention neural machine translation. In *AAAI*, pages 563–570. AAAI Press, 2018.

[299] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. In Doina Precup and Yee Whye Teh, editors, *ICML 2017*,

volume 70 of *Proceedings of Machine Learning Research*, pages 4189–4198. PMLR, 2017. URL http://proceedings.mlr.press/v70/zilly17a.html.