ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Neuro-symbolic Artificial Intelligence

A NEUROSYMBOLIC FRAMEWORK FOR MARKOV LOGIC NETWORKS

CANDIDATE
Arcangelo Alberico

SUPERVISOR
Prof. Paolo Torroni

CO-SUPERVISOR
Prof. Marco Lippi

Academic year 2020-2021

Session 2nd

C'è stato un momento in cui ho pensato:
"non ce la faccio, non ce la posso fare da solo"
Poi ho chiuso gli occhi
e ho immaginato me stesso mentre agivo.
E ce l'ho fatta: ho superato la paura e ce l'ho fatta.

Abstract

The impact of Deep Learning is due to the ability of its algorithm to mimic purely instinctive decisions, by reaching human-like performance in many isolated tasks, like image recognition or speech recognition. However, on the way to general artificial intelligence, which comprehend the set of cognitive abilities that gives machines a human-like intelligence it is extremely difficult to believe that these techniques, in an isolated way, can lead to a turning point. This because humans are still capable of performing more abstract and conscious reasoning processes on top of these instinctive tasks. This condition makes the need of a more complex and general theories, where the deep learning techniques constitutes only an ingredient of the final recipe.

In this thesis, we propose an implementation of a neuro-symbolic framework to merge symbolic and sub-symbolic reasoning and we aim to investigate how this integration improves deep learning systems with the use of additional background knowledge in form of symbolic rules. Starting from Markov Logic Networks we introduce neural networks to provide different weights for different grounding of the same formula and we inject sub-symbolic capabilities into MLNs.

Then, to test our implementation, we move to Argument Mining, a complex NLP task whose goal is the extraction of structured information from raw textual sources. We compare our approach to a baseline using only data without rules and to another neuro-symbolic framework, and establish that using logical rules during the training process gives a positive contribution to the task.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Artificial Intelligence (AI) is expected to be the next huge technological revolution, which is already shaping each and every aspect of our everyday lives. Automotive, banking, energy, fashion, healthcare, manufacturing are only a few among all the fields in which AI is making improvements and this trend can only increase in the next years.

Its impact has only been felt by the world in these last years, thanks to the advancement of a class of techniques, called deep learning, whose explosion in research and together with a huge improvement of the hardware, made possible the solution of tasks, such as image recognition, speech recognition, natural language understanding, which only few years earlier were deemed solvable only by human intelligence.

The advantage of the deep learning algorithms is their ability to mimic those human decisions that are more purely instinctive rather than due to a careful analysis. These kind of decisions include the associative tasks: if we look at an image of a car, we will say that it is a car without consciously considering each single part of it and analyze them, being a pure associative answer. This success of deep learning can be attributed to a paradigm shift from the original AI which was more interested in miming the conscious reasoning

process typical of human intelligence.

At this point we can ask ourselves if deep learning can be considered on the way to general artificial intelligence, which comprehend the set of cognitive abilities that gives machines a human-like intelligence. Even though there is no doubt that deep learning is making a lot of progress in this direction, it is also true that humans are still capable of performing more abstract and conscious reasoning processes on top of these instinctive tasks. In fact, recalling the image example, after having recognized the car, the human viewer will also try to check if a logical analysis of its answer make sense or not. This analysis goes in the direction of the reasoning process which is the AI approach before deep learning and is likely to be beneficial for this and other similar tasks in the future.

This example highlights how the human intelligence is a continuous interaction between these two mechanism: the former that is instinctive, fast, due to many similar experiences, and the latter that is slow and careful, generalizing single experience in general rules.

1.2 State-of-the-art

Neural-symbolic integration [15] is the task of merging symbolic and subsymbolic reasoning. This work has a long history but it has taken a new look after the deep learning explosion as a new AI research subfield. There are two major ways of performing this integration.

- Exploit neural techniques to improve purely symbolic tasks. This can be done in several ways. To begin with, thanks to their particular layered-structure of hidden variables, neural models are usually much faster in performing inference and they can be effectively used to perform a fast approximate inference [6, 7, 41]. In addition, neural models usually deal with subsymbolic representations of elements of the world under investigation allowing the exploitation of the particular geometry of the

perceptual world to simplify inference [10]. Indeed, at the end of the day, except very isolated cases, humans themselves reason about the world around them and not about abstract entities. Moreover, when this perceptual space is not known, lot of neural models still assume its existence and they encode a vectorized representation to symbolic entities, which is optimized as a parameter of the learning problem.

- Exploit symbolic techniques to relate multiple neural tasks. In multi-task learning, there are neural models solving multiple tasks simultaneously and in most of (if not all) the cases tasks are not isolated but are related to each other. This is sometimes referred to as structured prediction or structured learning, like in argument mining as we will see later. Structure is an high level source of knowledge, much more clean and valuable than single examples and it could improve neural models since it describes a lot of data in a compact way. So, embedding structure in a deep learning task would, in theory, bring a boost in performance.

Despite the consciousness of the importance of such an integration between the approaches, the development of a unifying theory is still missing. There are many valuable contribution but, probably due to the heterogeneity of skills of researchers needed, also a lot of confusion on how to compare the methods to underline the common features and the differences. This condition makes the need of a unified and broad theory describing multiple approaches and able to recover a symbolic approach in purely symbolic tasks and a subsymbolic approach in purely subsymbolic tasks.

1.3 Contributions

The major contributions of the thesis are as follows:

- Implementation of a neuro-symbolic framework combining the nerual networks with the symbolic method of the Markov Logic Networks

based on [23]

- Proposal of a method for learning weights of formula in combination with neural networks
- Application of the framework to an argumentation mining task

1.4 Outline

This thesis is structured as follows.

Chapter 2 introduces basic concepts necessary for the rest of the dissertation. In particular, the formalism of First Order Logic which is the basis of many approaches for expressing complexly structured knowledge. Then, Markov Logic Networks are introduced to apply first-order logic to practical AI problems and they are the basis of our neuro-symbolic integration.

Chapter 3 begins by describing ground-specific Markov Logic Networks [23] and introduces our approach to implement the integration of neural networks into MLNs. Then it goes more in details into the implementation describing the features, the syntax and the learning algorithm.

Chapter 4 describes the experiments done with the system. It first begins with a pure neural task, a multi-class classification problem, showing that without logic rules the framework behave like a standard neural network. Then, after introducing argument mining, our approach is applied to a related task to show how the introduction of rules improves the performances of the model.

Chapter 5 draws the conclusions of the presented work and introduces possible future improvements.

Chapter 2

Literature Review

2.1 First Order Logic

A logic is a formal system of sentences, supplied with syntax and semantics, together with mechanisms for asserting and deducting the truth of sentences. By making different ontological assumptions and by restricting the set of well-formed sentences, different logics can be introduced. The simplest, and most abstract logic we can study is called propositional logic.

Definition 2.1.1 (Proposition)**.** A proposition is a statement that can be either true or false; it must be one or the other, and it cannot be both.

First-Order logic is a generalization of propositional logic. Propositional logic can represent propositions, whereas first-order logic can represent individuals and propositions about individuals. For example, in propositional logic from *Socrates is a man* and *If Socrates is a man then Socrates is mortal* the conclusion *Socrates is mortal* can be drawn. In first-order logic this can be represented much more fine-grained. From *Socrates is a man* and *All man are mortal* the conclusion *Socrates is mortal* can be drawn.

In first order logic, formulas are constructed using four types of symbols: constants, variables, functions, and predicates. Constant symbols represent

objects in the domain of interest (e.g., people: `Anna`, `Bob`, `Chris`, etc.). Variable symbols range over the objects in the domain. Function symbols (e.g., `MotherOf`) represent mappings from tuples of objects to objects. Predicate symbols represent relations among objects in the domain (e.g., `Friends`) or attributes of objects (e.g., `Smokes`). An *interpretation* specifies which objects, functions and relations in the domain are represented by which symbols. Variables and constants may be typed, in which case their values range only over objects of the corresponding type. For example, the variable x might range over people (e.g., Anna, Bob, etc.).

A *term* is recursively defined as consisting of either a constant, a variable or a function applied to a tuple of terms. A term is said to be *ground*, if it contains no variables. An *atomic formula* or *atom* is a predicate symbol applied to a tuple of terms (e.g., `Friends(x, MotherOf(Anna))`). Formulas are recursively constructed from atomic formulas using logical connectives (disjunction $\vee$, conjunction $\wedge$, negation $\neg$, implication $\rightarrow$ and equivalence $\leftrightarrow$) and quantifiers (universal $\forall$ and existential $\exists$). A *positive literal* is an atomic formula; a *negative literal* is a negated atomic formula. A *ground atom* or *ground predicate* is an atomic formula all of whose arguments are ground terms. The *Herbrand base* of a FOL theory (set of sentences in First Order Logic) is the set of all ground atoms constructed using the predicates, functors and constants of the theory. A *Herbrand interpretation*, also called a (possible) world, is an assignment of a truth value to all atoms in the Herbrand base. A formula is satisfiable iff there exists at least one world in which it is true.

A *first-order knowledge base* (KB) is a set of sentences or formulas in first-order logic. Logical inference in first-order logic is the problem of determining if a knowledge base KB entails a given formula F, denoted $KB \models F$, which means that F is true in every world where all formulae in KB are true. For automated inference, it is often convenient to convert formulas to a more regular form, typically *Conjunctive Normal Form* (CNF), also known as clausal form.

A KB in *clausal form* is a conjunction of clauses, which consist of a disjunction of literals. Inference in first-order logic is only semidecidable. Because of this, knowledge bases are often constructed using a restricted subset of first-order logic with more desirable properties. The most widely used restriction is to *Horn clauses*, which are clauses containing at most one positive literal.

2.2 Markov Logic Networks

Markov Logic Networks (MLNs) [35] implement a probabilistic logic providing a general interface to integrate learning and probabilistic inference. In particular, first-order logic is used to define boolean Markov Random Fields (MRFs). A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in MLNs is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal. MLNs can be exploited to mostly carry out both inference and weight learning of the logical rules involved in a learning process. MLNs incorporate logical semantics defining feature functions into probability distributions to create models that capture both the structure and the uncertainty in machine learning tasks.

In particular, MLNs rely on the notion of Markov random field. An MRF is a probabilistic graphical model for the joint distribution of a set of variables and it is composed of an undirected graph expressing the variable dependencies and a set of potential functions. For each variable it is considered a node in the graph while a potential function (i.e. a non-negative function of the state of the corresponding clique) is associated to any clique of the graph.

Definition 2.2.1 (MRF)**.** Let $x = (x_1, \ldots, x_n) \in X$ be a vector of random

variables and let $\phi = (\phi_1, \ldots, \phi_m)$ be a vector of potentials, where each potential ϕ_j assigns a real-valued score to any configuration of the variables. Given $\omega = (\omega_1, \ldots, \omega_m)$ a vector of real-valued weights, a Markov Random Field is a probability distribution of the form:

$$P(x) = \frac{1}{Z} \exp\left(\sum_{j=1}^{m} \omega_j \phi_j(x)\right) \tag{2.1}$$

where $Z = \int_X \exp\left(\sum_{j=1}^{m} \omega_j \phi_j(x')\right) dx'$ is known as the partition function.

The integration with logic is carried out in MLNs as follows. Each potential function ϕ_j is associated to a first-order logic formula F_j in a knowledge base KB. A knowledge base can be seen as a set of constraints on the set of possible assignment, the fewer formulas an assignment violates, the more probable it is, while it has the lowest probability if it violates all the formulas. Each formula has to be considered either as hard (infinite weight) or can be weighted to penalize differently the assignments with respect to the formula satisfaction, the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not.

Definition 2.2.2 (MLN)**.** A Markov logic network L is a set of pairs (F_j, ω_j), where F_j is a FOL formula and ω_j is a real number. Relatively to a set of constants $K = k_1, \ldots, k_{|K|}$, it defines an MRF $M_{L,K}$ as follows:

- $M_{L,K}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground atom is true, and 0 otherwise;
- $M_{L,K}$ contains one feature for each possible grounding of each formula F_j in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the ω_j associated with F_j in L.

The graphical structure of $M_{L,K}$ follows from 2.2.2: there is an edge between two nodes of $M_{L,K}$ if and only if the corresponding ground atoms appear together in at least one grounding of some formula in L [35]. To better

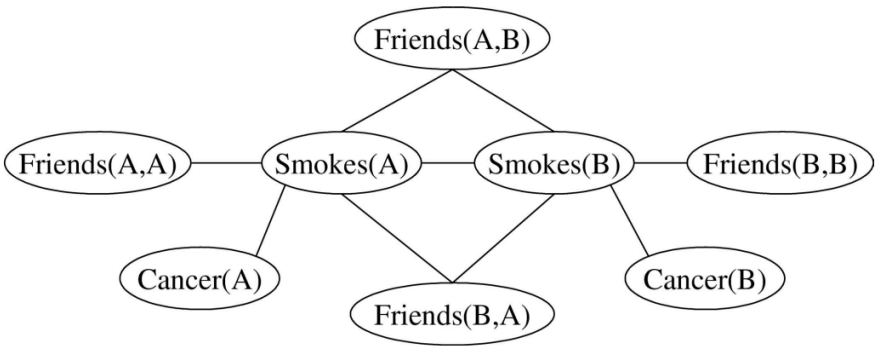


Figure 2.1: Ground Markov Network obtained by applying the formulas in 2.2, to the constants Ann (A) and Bob (B)

visualise this, consider the follow example: we want to build a model to infer whether a person smokes, and whether a person has cancer with a some probability. And we have a first-order KB that, for simplicity, has only the two formulas presented in (2.2).

$$\begin{aligned} &\texttt{Smokes(a)} \rightarrow \texttt{Cancer(a)} \\ &\texttt{Friends(a,b)} \rightarrow (\texttt{Smokes(a)} \leftrightarrow \texttt{Smokes(b)}) \end{aligned} \tag{2.2}$$

The first step in a MLN is the grounding of the formulas presented in the KB. Figure 3.1 shows the graph of the ground Markov network defined by the formulas in 2.2 and the constants Ann (A) and Bob (B). Each node in this graph is a ground atom (e.g., Friends(A, B)). The graph contains an arc between each pair of atoms that appear together in some grounding of one of the formulas. $M_{L,K}$ can now be used to infer the probability that Ann and Bob have cancer given whether or not they smoke and they are friends. Each state of $M_{L,K}$ represents a possible world. A possible world is a set of objects, a set of functions (mapping from tuples of objects to objects), and a set of relations that hold between those objects; together with an interpretation, they determine the truth value of each ground atom. The following assumptions ensure that the set of possible worlds for (L, K) is finite, and that $M_{L,K}$ represents a unique, well-defined probability distribution over those worlds, irrespective

of the interpretation and domain. These assumptions are:

Unique names Different constants refer to different objects;

Domain closure The only objects in the domain are those representable using the constant and function symbols in (L, K);

Known functions For each function appearing in L, the value of that function applied to every possible tuple of arguments is known, and is an element of K [35].

The possible groundings of a predicate in 2.2.2 are obtained simply by replacing each variable in the predicate with each constant in K, and replacing each function term in the predicate by the corresponding constant.

2.2.1 Learning

Once the grounded Markov Network is constructed, the next step in a MLN, is to calculate the weights associated to each formula in the KB. There are two approaches to weight learning in MLNs: generative and discriminative. In generative learning, there is no separate notion of query or evidence atoms. The weights are learned by maximizing the log-likelihood of the entire set of ground atoms [30]. In particular, given a set of formulas and a database of atoms, a training set, we wish to find the formulas' maximum a posteriori (MAP) weights, e.g., the weights that maximize the product of their prior probability (prior probability represents what we originally believed before new evidence is uncovered) and the data likelihood. Since optimization is typically posed as error function minimization, we will equivalently minimize the negative log-likelihood [25].

Then, there is another weight learning approach called discriminative learning. In discriminative learning, we know a priori which predicates will be used to supply evidence and which ones will be queried [30]. In many applications, like in classification problems, we know a priori which atoms will be evidence

and which ones will be queried, and the goal is to correctly predict the latter given the former.

2.2.2 Limitation on MLNs

One of the main application of supervised machine learning is to find associations between given information (features) and the target (the class, in the case of classification) which has to be predicted. In the case of (probabilistic) first-order logic, this association can be interpreted as an implication, containing some query predicate in the right-hand side and some evidence predicates in the left, as showed in the following formula:

$$\texttt{Feature(x,a}_1\texttt{,…,a}_n\texttt{)} \wedge \texttt{Feature(y,b}_1\texttt{,…,b}_n\texttt{)} \rightarrow \texttt{Target(x,y)} \quad (2.3)$$

Following the standard approach of the MLNs we would associate a single weight to such a formula and this would generate a too simple model considering all the features equal and the respective MLN would merely learn the classifier that predicts the most frequent realization of the predicate Feature in the dataset. This approach would not take advantage of the discriminative power of the features: it is instead necessary to learn different weights for different combinations of features. A way to tacke this issue is to learn a different weight for each possible combination of constants but this would lead to a huge number of formulae: assuming that each variable $a_1, \ldots, a_n, b_1, \ldots, b_n$ has k possible realizations (constants), then the number of generated formulae would be $N = k^{2n}$. The exponential growth of the number of parameters makes this approach unsuitable for the present application in almost any real world problem. Another alternative is to avoid predicates of high arity by splitting the equation (2.3) in a set of $2n$ different formulae. This can be done

by replacing the predicate `Feature` in several different predicates $\texttt{Feature}_j$:

$$\begin{aligned} \texttt{Feature}_1\texttt{(x,+a}_1\texttt{)} &\rightarrow \texttt{Target(x, y)} \\ &\dots \\ \texttt{Feature}_n\texttt{(y,+b}_n\texttt{)} &\rightarrow \texttt{Target(x, y)} \end{aligned} \tag{2.4}$$

Considering the formulae in (2.4) would create an MLN with $2n \times k$ weights, resulting in a much simpler model. Such an MLN would use only a linear combination of the features, being exactly equivalent to a logistic regression model with $2n$ multinomial input attributes [18]. On the other hand, in several related applications of machine learning to real world problems, it is fundamental to use a non-linear combination of the features in order to achieve accurate predictions.

Another limitation of standard MLNs is that they cannot compare different constants of the same type. If we consider for example a discrete attribute, where there is an ordering between the outcome of the variable, for example a rating system, in a logic representation there will be no direct information that one value is closer to another one. This means that also real-valued attributes cannot be directly used in Markov logic, while very often they represent a key information for many tasks [23].

Chapter 3

Neural Integration of Markov Logic Networks

3.1 Markov Logic Networks with grounding specific weights

The solution proposed in [23] to overcome standard MLNs limitations consists in assigning different weights to different groundings of the same first-order logic rule. In particular, each weight will depend on the specific groundings of a subset of the variables appearing in the first-order logic formula: these variables are called selected variables. In [23], for weight learning, a discriminative approach is used and, therefore, the conditional probability of query atoms Y given evidence X can be expressed with the following:

$$P(Y = y|X = x) = \frac{e^{\sum_{F_i \in F_y} \sum_j \omega_i(g_{ij},\theta_i)\cdot n_{ij}(x,y)}}{Z_x} \tag{3.1}$$

where g_{ij} denotes the j-th ground configuration of the selected variables in the i-th formula, ω_i a real-valued parameterized function returning the weight attached to each ground formula after the selected variables have been bound to the constants in g_{ij} and $n_{ij}(x, y)$ is the number of true groundings in (x, y)

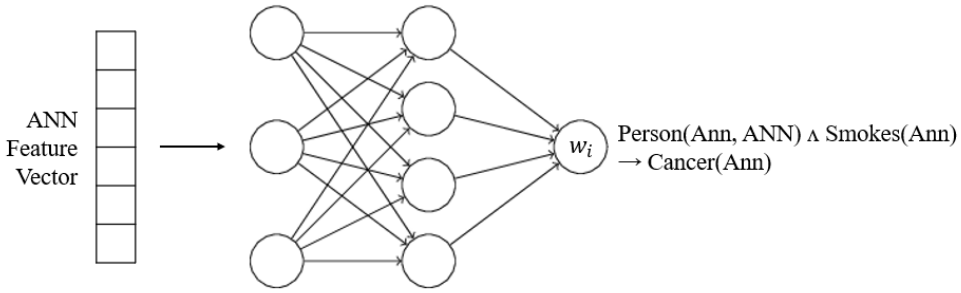


Figure 3.1: Weight assignment to a specific grounding of a first-order logic rule

matching g_{ij}. In general, θ_i is a parameter vector. The special case where θ_i is a scalar and $\omega_i = \theta_i$ recovers standard MLN. In general, ω_i can be any function, computed for example using kernel machines or multilayer perceptrons. In case of kernel logistic regression (KLR), for example, the weight attached to ground formula g_{ij} will be computed as:

$$\omega_i(g_{ij}) = \theta_i^T \phi(g_{ij}) \tag{3.2}$$

being ϕ the feature mapping induced by the kernel. In the following the grounding-specific weights will be computed by neural networks as showed in Fig.3.1. The introduction of neural networks to provide different weights for different grounding of the same formula brings to the definition of predicates which purpose is only to link a constant with its relative feature vector. These predicates are called Feature Predicates. The truth value of these predicates is always known and it is true for correct matching and false for incorrect ones. This is important to provide the neural networks always with the correct inputs for a specific grounding of a formula.

3.2 Learning

We want to train the neural networks in order to maximize the pseudo log-likelihood of the entire set of ground atoms. This reflects the generative settings of the MLN weight learning. We can consider all the ground atoms of an MLN as a set of random variables $Y = (Y_1, \ldots, Y_l, \ldots, Y_n)$ where Y_l can assume values in $\{0, 1\}$ depending on the truth value of the lth ground atom. We learn MLN weights from one or more relational databases. We make a closed world assumption: if a ground atom is not in the database, it is assumed to be false. A database is effectively a vector $y = (y_1, \ldots, y_l, \ldots, y_n)$ where y_l is the truth value of the lth ground atom. Given a database, MLN weights can in principle be learned using standard methods. Unfortunately, counting the number of true groundings of a formula in a database is intractable, even when the formula is a single clause. A more efficient alternative is to optimize the pseudo-likelihood, which is an approximation to the joint probability distribution of a collection of random variables. However, the pseudo-likelihood ignores non-local interactions between variables, and may lead to poor results when inference across non-neighboring variables is required [39]. This is not our case because in our experiment we are going to use a simple MLN with few and simple rules.

Given a set of random variables $Y = (Y_1, Y_2, \ldots, Y_n)$ the pseudolikelihood of $Y = y = (y_1, y_2, \ldots, y_n)$ is:

$$P_w^*(Y = y) = \prod_{l=1}^{n} P_w(Y_l = y_l | MB_y(Y_l)) \tag{3.3}$$

where $MB_y(Y_l)$ is the state of the Markov blanket of Y_l in the database.

To calculate the pseudolikelihood $P_w(Y_l = y_l | MB_y(Y_l))$ of each variable

given the state of its Markov Blanket in the database, we have to recall the Local Markov property of the Markov Random Fields, which states that a variable is conditionally independent of all other variables given its neighbours:

$$X_v \perp\!\!\!\perp X_{V \setminus N[v]} | X_{N(v)} \tag{3.4}$$

In a undirected graphical model, one can show that a node's Markov blanket is its set of immediate neighbors. So, in our case, considering Y_l the node correspondent to a certain variable and X all the other variables, we will have:

$$Y_l \perp\!\!\!\perp X \setminus cl(Y_l) | MB(Y_l) \tag{3.5}$$

where $cl(Y_l) \triangleq MB(Y_l) \cup \{Y_l\}$ is the closure of node Y_l. By the independence we have that:

$$P_w(Y_l = y_l | X = x) = P_w(Y_l = y_l | MB_y(Y_l)) \tag{3.6}$$

This result allows us to use the equation (3.1) for the computation of the pseudolikelihood of the single variable, considering the variable itself as the query atom and the variables of the Markov Blanket as the evidence atoms.

We can summarize (combining (3.3), (3.1) and (3.6)) the final result with the closed form of the pseudo-likelihood of the world $y = (y_1, y_2, \ldots, y_n)$

$$P_w^*(Y = y) = \prod_{l=1}^{n} \frac{e^{\sum_{F_i \in F_{y_l}} \sum_j \omega_i (g_{ij}, \theta_i) \cdot n_{ij}(MB_y(Y_l), y_l)}}{Z_{MB_y(Y_l)}} \tag{3.7}$$

Defining $l(\theta) = \log P(Y = y; \theta)$ as the pseudo log-likelihood, it can be shown that the gradient with respect to a generic parameter θ_{ik} contributing to weight ω_i is:

$$\frac{\partial l(\theta)}{\partial \theta_{ik}} = \frac{\partial l(\theta)}{\partial \omega_i}\frac{\partial \omega_i}{\partial \theta_{ik}} = \\ = \left(\sum_{l=1}^{n} \Big[n_i(x) - P_w(Y_l = 0 | MB_y(Y_l)) n_i(y_{[Y_l=0]}) \right. \\ \left. - P_w(Y_l = 1 | MB_y(Y_l)) n_i(y_{[Y_l=1]}) \Big] \right) \frac{\partial \omega_i}{\partial \theta_{ik}} \tag{3.8}$$

where $n_i(y_{[Y_l=0]})$ is the number of true groundings of the ith formula when we force $Y_l = 0$ and leave the remaining data unchanged, and similarly for $n_i(y_{[Y_l=1]})$. The partial derivatives $\partial \omega_i / \partial \theta_{ik}$ can be obtained by the backpropagation of the neural network.

One of the main advantages in using this approach is that computing (3.8) (or (3.7)) does not require inference over the model and the computation can be made more efficient noting these things:

- Considering only the Markov Blanket, the sum in (3.8) can be sped up by ignoring predicates that do not appear in the ith formula;
- We can compute only once the counts $n_i(x)$, $n_i(y_{[Y_l=0]})$ and $n_i(y_{[Y_l=1]})$ given that they do not change with the weights;
- We can ignore ground formulas whose truth value is unaffected by changing the truth value of any single literal leading to the equality of the counts in the summation, specifically $n_i(x) = n_i(y_{[Y_l=0]}) = n_i(y_{[Y_l=1]})$. This can often be the great majority of ground clauses because it holds for any clause which contains at least two true literals.

3.3 Implementation

3.3.1 Syntax and notation

In order to show the implementation of Markov logic networks with grounding specific weights, we use a purposely modified version of the "Smokers" dataset [35] which is a classical example in statistical relational learning literature. Here, three relations are defined on a set of constants representing people: the unary predicate `Smokes` identifies those people who smoke, the unary predicate `Cancer` identifies those people who have a cancer and the binary predicate `Friends` indicates that two people are friends. This dataset is often used to show how a statistical relational learning algorithm can model a distribution by finding a correlation of smoking habits of friends.

MLN and Database definition

The main files used for running a simulation are the MNL and the DB file. The MLN file contains all the predicates and the formulas used as we can see here

```
   # Predicate declaration
Person(id,&person)
Cancer(id)
Friends(id,id)
Smokes(id)
   # Formula declaration
Person(a,$pa) ∧ Smokes(a) → Cancer(a)
Person(a,$pa) ∧ Person(b,$pb) ∧ Friends(a,b) → (Smokes(a) ↔
Smokes(b))
```

while the DB file contains the evidences of our world

```
Person(Ann, ANN)
Person(Bob, BOB)
Cancer(Ann)
```

```
!Cancer(Bob)
!Friends(Ann,Bob)
```

Predicate declaration

In predicate declaration, we allow a type T to be preceded by an ampersand (`&` character): this indicates that constants $C_1, \ldots, C_n$ of type T found in the .db file are not just simple constants, but pointers to feature vectors. In particular, the name of constant C_k will be the index of the feature vector associated with that constant within the features file.

For example, if we declare the following predicate

```
Person(id, &person)
```

and in the .db file we find the following ground predicate

```
Person(Ann, ANN)
```

then, constant named `ANN` will point to vector of features: in this case, the features associated to the person Ann.

Rule declaration

In order to declare a rule with grounding-specific weights, in rule declaration we allow the presence of dollar symbols (`$` character) preceding variables: for example, if in a certain rule two variables x and y are both preceded by a dollar sign `$`, then the first-order formula will be associated to a neural network which will compute grounding-specific weights, taking in input the information given by the realizations of variables x and y. Let us clarify this idea with an example. Consider the following predicates and formula:

```
Person(id, &person)
Friends(id, id)
Smokes(id)
Person(a,$pa) ∧ Person(b,$pb) ∧ Friends(a,b)
```

$$\rightarrow \texttt{(Smokes(a)} \leftrightarrow \texttt{Smokes(b))}$$

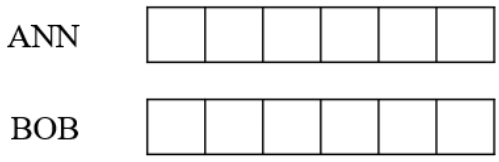
In this case, we have a formula with two dollars, corresponding to two `person` variables, pa and pb. Type `person` has a `&` in `Person` predicate declaration, so `person` constants will point to feature vectors (see Section 3.3.1), which will be the input to the neural network. Suppose for example that in the .db file the following two ground predicates exist:

```
Person(Ann,ANN)
Person(Bob,BOB)
```

Then, the weight of the following *ground* formula

$$\texttt{Person(Ann,ANN)} \wedge \texttt{Person(Bob,BOB)} \wedge \texttt{Friends(Ann,Bob)}$$
$$\rightarrow \texttt{(Smokes(Ann)} \leftrightarrow \texttt{Smokes(Bob))}$$

will be computed by a neural network which takes in input the vectors of features indexed by `ANN` and `BOB` constants in the global features file. In this way, each grounding of the first-order clause will have a different weight. The features file will be organized one vector per line, indexed by the constant names, as follows:



3.3.2 Grounding

One of the innovation introduced in the framework regards the grounding side. In fact, in general, for each formula all the possible groundings are made, leading also to some meaningless configurations. We tackled this issue by acting on the `Feature Predicate` and considering acceptable only the ones inside the DB file. In this way, considering the smokers example, the grounding of the Feature Predicate

```
Person(id,&person)
```

will only generate as groundings

```
Person(Ann,ANN)
Person(Bob,BOB)
```

and not also

```
Person(Ann,BOB)
Person(Bob,ANN)
```

that will lead to feed the neural network with the wrong features.

3.3.3 Learning algorithm

The learning part of the framework consists of two main parts: the preparation and the training.

Preparation

This is done only once before starting the training of the neural networks and creates the variables that will be used, at training time, for taking into account the logical part of the problem. The output dictionaries of this part are:

- A dictionary containing, for each variable, the indexes of the grounding of the formulas that the variable makes true;
- A dictionary containing, for each grounding of each formula, the number of times that each value of the variable makes that grounding true.

In fact, in this part an iteration on all the groundings of the formulas is done, for each grounded formula all the values of the variable associated to each grounded atom are considered. In our case we only consider binary variables because the atoms can be true or false. Then, for each of these values, the evidence world is updated accordingly and the truth value of the corresponding formula is checked. In case it is true the dictionaries are updated properly.

Training

The training is implemented with a standard training loop where the forward pass consist of feeding all the neural networks with the respective vectors of features. Then the output weights are used to calculate the likelihood (3.7) of the evidence database using the dictionaries obtained in the preparation phase. The backpropagation is done automatically with the automatic differentiation implemented in Pytorch and it is enhanced by multiplying all the weights by the gradient of the MLN.

Chapter 4

Experiments

The framework presented has been thought with a versatility principle: *able to adapt or be adapted to many different functions or activities*. With this in mind, it can be applied to any deep learning task regardless of the symbolic rules. In fact, if there are no constraints to be satisfied, it will behave like a standard neural network. In this chapter we will go through one of these examples, showing how to translate a typical deep learning task into an MLN and an experiment performed on an Argument Mining task which will benefit of the symbolic rules introduced with the framework. This happens because 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.

4.1 Multiclass Classification with MNIST

Multiclass classification is the problem of classifying instances into one of three or more classes. Each sample can only be labeled as one class. For example, classification using features extracted from a set of images of fruit,

where each image may either be of an orange, an apple, or a pear. Each image is one sample and is labeled as one of the 3 possible classes. Multiclass classification makes the assumption that each sample is assigned to one and only one label - one sample cannot, for example, be both a pear and an apple.

The MNIST database (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger NIST Special Database 3 (digits written by employees of the United States Census Bureau) and Special Database 1 (digits written by high school students) which contain monochrome images of handwritten digits. The digits have been size-normalized and centered in a fixed-size image. The original black and white (bilevel) images from NIST were size normalized to fit in a 20×20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28×28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28×28 field [22]. The task consists to classify the image of a digit into the relative digit.

Figure 4.1 shows the Markov Logic Network related to this task. We can see that we only need two predicates: one Feature Predicate `Digit` to link each sample to the feature vector representing the image and a predicate `Number` used as the variable for the classification. For the network part, we have only a neural formula which takes care of the classification of the digit.

For example, if in the Database we have `Digit(ID0,F0)`, the framework will generate the following groundings for the neural formula:

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 0)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 1)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 2)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 3)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 4)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 5)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 6)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 7)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 8)`

`Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 9)`

The weight of each grounded formula corresponds to the relative output of the Convolutional Neural Network (CNN) used for the classification (e.g the output 0 is the weight of the formula `Digit(ID0, F0)` $\rightarrow$ `Number(ID0, 0)`) and in case the Database contains `Number(ID0,5)` the training will push the output 5 of the CNN to an higher value with respect to the others.

```
Digit(id, &digit)
Number(id, n)

n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Digit(a, $fa) → Number(a, n)
```

Figure 4.1: MLN of the MNIST experiment

4.1.1 Experiments

To show the capability of our approach to recover a subsymbolic approach in purely subsymbolic tasks, we make an experiment on the MNIST dataset. Due to the fact that for each digit the framework generates 11 variables (1 Feature Predicate and 10 predicates for each possible digit), using the whole dataset of 60,000 samples for training would have generated 660,000 variables, making the whole task unfeasible in the grounding phase. So, we have limited the number of training samples to 2,000 since we are only interested in showing that the behaviour, in comparison the pure neural network approach, is the same.

For what concerns the networks' architecture, we rely on a simple architecture made of two convolutional layers followed by two fully-connected layers for the classification. We use ReLU as activation function and employ dropout after the two central layers. To avoid overfitting both methods are trained with early-stopping on the accuracy score of the class classification, using a patience of 100 epochs.

Table 4.1 reports the results of our experiment on the whole validation set. The evaluation has been done on the accuracy of the predictions. We can notice, from the results, that the two method are completely equivalent and the negligible difference can be attributed to stochastic variations in training phases. This confirms that our method can behave like a pure neural model in absence of symbolic rules.

	Neural Nets	GSMLN
Validation set	96.36	**96.55**

Table 4.1: Experiment on MNIST dataset. The first column presents the pure neural training and the second one the training involving GSMLN. Scores are reported as percentage values.

4.2 Argument Mining

Argument Mining (AM) is a discipline that stems from Natural Language Processing (NLP) and Knowledge Representation and Reasoning (KRR) [5] with the goal to automatically extract arguments and their relations from a given natural language document [24]. 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* [17]. AM's goal is therefore the extraction of structured information from raw textual sources, giving an understanding of the relations between single arguments and the complex structure they create. Among the many useful and practical

applications of such a discipline, it can be used to perform fact-checking and recognize misleading content [8, 13], support decision making in healthcare application [29] and the legal domain, improve the understanding of the position of political candidates [31], support teachers in an educational context [26], and support debate technology [40]. AM can be generally divided into a series of subtasks [5, 21, 24] that are often addressed in a pipeline fashion. Typically, there is a first phase of component detection followed by a relation prediction task. Component 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 former may be opinions and hypotheses expressed by the author, while the latter are facts and objective information that are reported in the document. 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). Two examples of relationships are support and attack. The former is when an evidence component does provide information based on which a claim can be made, while the latter is when two claims contradict each other and therefore can not both be true at the same time.

All these subtasks can be addressed independently, but, since the information obtained by one of them can give a meaningful insight on the other, it is beneficial to address them together. In fact, 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. This is the reason why most of the approaches adopt a pipeline scheme in order to exploit the knowledge gathered in the component detection to perform the more challenging task of relation prediction. Another method is to use 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. However, more work is still required and despite the already made attempts to integrate this knowledge into DL approaches, the proper formal integration of the two worlds in this domain has not been carried on yet.

4.2.1 Neuro-symbolic Argument Mining

Argumentation mining requires an ability to exploit common sense knowledge, typical of human reasoning, to understand whether a given piece of evidence supports a given claim, or whether two claims attack each other. The tasks involved in the AM systems, from the simpler one like argument component detection to more complex one like argumentation structure prediction, have seen a great improvement in the recent years due to the development of new deep learning techniques in the field of NLP. Despite this great advance of deep neural networks in NLP, we claim that, in case of AM, using these strategies alone will no longer be enough to tackle such complicated issues. On the other hand, structured arguments have been studied and formalized for decades using models expressed in a logic framework [2] with knowledge expressed in the form of rules and constraints to AM. So far, these two worlds have advanced largely independently of each other, but we count on that a large advancement in AM ought to come from the combination of symbolic and sub-symbolic approaches, such as these developed in the Neural Symbolic (NeSy) [4] or Statistical Relational Learning (SRL) [9, 16, 19] communities. These approaches, differently from simple techniques of applying a set of rules on the output of a neural network, can directly enforce (hard or soft) constraints during training and, in so doing, penalizing a solution that does not satisfy them. Consequently, we can train different neural networks for different tasks (e.g one to classify argument components and one to detect links between them) and, during the training process, impose global constraints to tune the weights of the networks to have valid solutions. Apart from the work

described in this thesis, there are many systems that enable this scheme, such as DeepProblog [27] and Logic Tensor Network [37], that we are going to use for comparison in the experiments.

Logic Tensor Networks (LTN) [11, 12, 38, 37] is a framework that integrates first-order many-valued logical reasoning [3] with tensor networks [34], implemented in TensorFlow [1]. By combining neural networks with first-order fuzzy logic it allows efficient learning from noisy data in the presence of logical constraints, and reasoning with logical formulas describing general properties of the data. It uses a differentiable first-order logic language, called Real Logic, to incorporate data and logic. LTN belongs to the "tensorization" approaches, a class of undirect neuro-symbolic approaches [33] which embed First Order Logic entities, such as constants and facts, into real-valued tensors. Hence, it is possible to use FOL to impose soft constraints at training time and to verify and investigate properties at test time. LTNs are limited to the application of the knowledge at training time [36].

4.3 Dataset and Rules

4.3.1 AbstRCT

The AbstRCT Corpus [29] extends a previous work [28], 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. Components are labeled as EVIDENCE (2808) and CLAIM (1390), while relations are labeled

as SUPPORT (2259) and ATTACK (342). Out of 25,000 possible pairs of components, about 10% of them have a relationship.

4.3.2 Rules and MLN definition

Following the directives in 3.3.1 we have to define a Feature Predicate `Text` which links each statement to the relative feature vector, then we have a predicate `Type` that define the type of the statement among `Claim` and `Evidence` and a third predicate `Link` to establish which components are in an argumentative relationship.

Given that we are interested in component classification and link prediction, we define two neural formulas linked to two neural networks whose purpose is to output the weight relative to the classification variable.

For what concerns the hard rules that characterize the GSMLN approach, from the properties of the dataset we can extract two axioms:

- no symmetric link can exist, so if there is a link between two components there must be no inverse link. This rule can be stated in first order logic formalism in this way:

$$\texttt{Link(id1, id2)} \rightarrow \texttt{!Link(id2, id1)} \tag{4.1}$$

- transitivity property of claims, or better claims can be linked only to other claims, so if a component is classified as a claim and it is linked to another component, then also this component is of type claim. This rule can be stated in first order logic formalism in this way:

$$\texttt{Type(id1, Claim)} \wedge \texttt{Link(id1, id2)} \rightarrow \texttt{Type(id2, Claim)} \tag{4.2}$$

Figure 4.2 summarize the Markov Logic Network previously described.

```
Text(id, &feature)
Type(id, type)
Link(id, id)

type = {Claim, Premise}

Text(id1, $f1) → Type(id1, type)
Text(id1, $f1) ∧ Text(id2, $f2) → Link(id1, id2)

Link(id1, id2) → !Link(id2, id1).
Type(id1, Claim) ∧ Link(id1, id2) → Type(id2, Claim).
```

Figure 4.2: MLN of the AbstRCT experiment

4.4 Method and Architecture

4.4.1 Architecture

We tackle both component classification and link prediction as classification tasks. We define two neural networks TypeNetwork and LinkNetwork, 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 probability distribution over 0 and 1 which represents the probability of the existence of an argumentative link between two components. For sentence embeddings, we have decided to use GloVe [32] 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. 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.

4.4.2 Method

To compare the approaches and evaluate if the use of symbolic rules yields positive results, we perform a baseline training relying only on the data. In order to compare, also, our system with an already existent framework we decided to use Logic Tensor Network with the same architecture as done in [14].

Then, for using our approach, we have tried to introduce the logical rules in different ways: training the whole model from scratch with the rules or doing a neural pre-training and a logical fine tuning with the axioms. We saw that doing everything from the beginning was not only slower but also worse in terms of performance with respect to the baseline. So, we tought that pre-training the models would benefit because it brings the model on the right path and then the fine tuning allow the model to refine itself with logical rules. To avoid overfitting, the baseline is trained with early-stopping on the F1 score of the link prediction, using patience of 1000 epochs, while the GSMLN consist of a further training with early-stopping, on the same metric, with a patience of 100 epochs.

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. We have therefore decided to extend our initial analysis, repeating the training procedure 20 times, with different seeds, obtaining for each configuration 20 trained neural networks. We evaluate our models in two different ways. At first, we consider for each metric the average of the scores (AVG) obtained by every single network. Then, we evaluate the predictions obtained using all the 20 models in ensemble voting. In our ensemble setting the class of each entity is assigned as the class voted by the majority (MAJ) of the networks. We train an ensemble of 20 networks both for TypeNetwork and LinkNetwork, and evaluate the aggregated output.

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. Then, we will compute the degree of agreement between the networks related to the predictions of the ensemble.

4.5 Experimental Results

Table 4.2 summarizes the results of our experiments. The results for the LTN experiments are taken from [14]. For the classification tasks, we report the F1 score for each class and the macro-F1 score for component classification. For the link prediction task we report the F1 score for each class. To measure the agreement between the models of the ensemble we use the Krippendorff's alpha [20]. For what concerns the AM tasks, the difference between the MAJ and AVG approaches is negligible in both the rule-based and the no-rules setting either for component classification and for link prediction, where the difference is at most 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. In comparison with LTNs, our approach performs better in predicting the link relation in all the datasets. Conversely, our approach has performed slightly worse on component classification with respect to the other two methods, but such a difference is not present in all the datasets. The agreement between the networks is at least acceptable in all the settings, with higher values for component classification. The use of rules leads to a decrease in terms of agreement for link prediction despite being still acceptable, while it brings no difference for component classification, except for LTNs where we can see a drop.

			Only Data		LTN		GSMLN	
Test set	Aspect	Criterion	MAJ	AVG	MAJ	AVG	MAJ	AVG
Neoplasm	Classification	Components	**80**	79	79	78	**80**	79
		Evidence	**86**	**86**	84	83	**86**	**86**
		Claim	74	73	74	74	**75**	73
		Link	35	35	35	35	**37**	**37**
		No-Link	83	83	85	85	**88**	**88**
	Agreement	Component class.	**86**		79		**86**	
		Link prediction	**79**		70		70	
Glaucoma	Classification	Components	79	78	81	**82**	78	78
		Evidence	**88**	**88**	**88**	**88**	**88**	**88**
		Claim	69	68	**75**	**75**	68	68
		Link	46	47	47	45	48	**49**
		No-Link	88	88	90	89	**92**	**92**
	Agreement	Component class.	**81**		75		**81**	
		Link prediction	**79**		71		69	
Mixed	Classification	Components	79	78	**81**	80	78	78
		Evidence	**86**	**86**	**86**	85	**86**	**86**
		Claim	71	70	**76**	75	70	70
		Link	39	39	39	**40**	39	**40**
		No-Link	85	85	87	87	**89**	**89**
	Agreement	Component class.	**84**		76		**84**	
		Link prediction	**79**		69		70	

Table 4.2: Results of neuro-symbolic AM on AbstRCT. The first two columns presents the baseline approach, the following two the approach involving LTN and the last two the approach involving GSMLN. Scores are reported as percentage values.

4.6 Discussion

In the framework, logic rules play an important role during the training process of the neural networks. The definition of training rules 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. This holds for both the hybrid approaches that has been considered. Furthermore, in our work, the symbolic part can be plugged in at any time of the training giving rise to different learning configurations by switching between the two even from one epoch to another. Such modularity would be also 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 [14].

Our results show how the introduction of two symbolic rules has given a positive contribution to the task, increasing the accuracy of the link prediction. Even if the architecture used for the experiment is relatively simple and the results may be not comparable with the state of the art, we think that the impact of rules may hold even for more advanced models.

Chapter 5

Conlusions and Future Works

Nowadays there is increasing awareness of the importance of the subfield of neuro-symbolic in the AI community. Probably, this is also occurring at a point in time when we are starting to discover and recognize the inherent limitations of pure deep learning approaches. The use of additional background knowledge is a natural way to try to further improve deep learning systems, and much of this line of work falls into the neuro-symbolic AI field. The general opinion appears to be that combining neural and symbolic approaches is at least a path forward to much stronger AI systems and towards human-level artificial intelligence.

The purpose of this thesis has been the implementation of a neuro-symbolic framework to merge symbolic and sub-symbolic reasoning. We started from Markov Logic Networks, which already combine logic and probability by attaching weights to first-order clauses, then we introduced neural networks to provide different weights for different grounding of the same formula and, in so doing, we injected sub-symbolic capabilities into MLNs.

To test the integration, we have decided to use a task of Argumentation Mining, a research field that both symbolic and sub-symbolic approaches find challenging. We have compared our approach to a baseline using only data without rules and to another neuro-symbolic framework, Logic Tensor Networks. Our results have shown that using a neural-symbolic framework to

introduce logical rules during the training process gives a positive contribution to the task, increasing the accuracy of the link prediction and performs slightly better than LTN.

The framework is still in an early stage and a future development could be an effort to optimize the computation of the pseudo-likelihood. The current implementation provides a nesting of three for-loops for the computation of the pseudo-likelihood of a single variable, and, since it has to be computed for all variables, it produces a sequential bottleneck on the whole training step. This is due to the iterations on the dictionaries that embed the symbolic part. Given that they are based on values computed at the beginning of the training algorithm, they can be redesigned into data structures (e.g., three dimensional sparse matrices with formaulas as rows, groundings as columns and variables as pages) that can avoid a sequential computation and enrich the feature vector of each variable with information on the symbolic part (e.g., add the variable, formula and grounding indices to the feature vector) to retrieve, in a constant computational cost, the data needed. This could also benefit of the parallel computation on GPUs and increase a lot the speed of the process.

Another aspect to consider is that, given that the implementation of the learning algorithm follows the generative approach, we do not take advantage of known predicates which can be used to supply evidence. Moreover, the implementation with pseudolikelihood may lead poor results when inference across non-neighboring variables is required. This can be tackled by extending the framework with a discriminative learning approach using the conditional likelihood, which will not be so different from what we already do with the Feature Predicates whose truth value is always known.

In conclusion, an improvement can be done also on the grounding of the Database optimizing the process for using larger datasets. After this, it would be interesting to test our approach on problems with a richer, more expressive domain than the datasets we used here.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: large-scale machine learning on heterogeneous distributed systems, 2016. arXiv: 1603.04467 [cs.DC].

[2] T. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10):619–641, 2007. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2007.05.001. URL: https://www.sciencedirect.com/science/article/pii/S0004370207000793. Argumentation in Artificial Intelligence.

[3] M. Bergmann. In *An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems*. Cambridge University Press, 2008, pages 321–326. DOI: 10.1017/CBO9780511801129.020.

[4] T. R. Besold, A. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha. Neural-symbolic

learning and reasoning: a survey and interpretation, 2017. arXiv: 1711.03902 [cs.AI].

[5] E. Cabrio and S. Villata. Five years of argument mining: a data-driven analysis:5427–5433, July 2018. DOI: 10.24963/ijcai.2018/766. URL: https://doi.org/10.24963/ijcai.2018/766.

[6] J. Chang and D. Blei. Relational topic models for document networks. *Journal of Machine Learning Research - Proceedings Track*, 5:81–88, January 2009.

[7] J. Choi and E. Amir. Lifted relational variational inference, 2012. arXiv: 1210.4867 [cs.LG].

[8] O. Cocarascu and F. 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, December 2018. DOI: 10.1162/coli_a_00338. URL: https://doi.org/10.1162/coli%5C_a%5C_00338.

[9] L. De Raedt, K. Kersting, S. Natarajan, and D. Poole. Morgan & Claypool Publishers, 2016. DOI: 10.2200/S00692ED1V01Y201601AIM032.

[10] M. Diligenti, M. Gori, and C. Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244, September 2015. DOI: 10.1016/j.artint.2015.08.011.

[11] 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.

[12] I. Donadello, L. Serafini, and A. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1596–1602, 2017. DOI: 10.24963/ijcai.2017/221. URL: https://doi.org/10.24963/ijcai.2017/221.

[13] M. Dusmanu, E. Cabrio, and S. Villata. Argument mining on Twitter: arguments, facts and sources:2317–2322, September 2017. DOI: 10.18653/v1/D17-1245. URL: https://aclanthology.org/D17-1245.

[14] A. Galassi. *Deep Networks and Knowledge: from Rule Learning to Neural-Symbolic Argument Mining*. PhD thesis, alma, Maggio 2021. URL: http://amsdottorato.unibo.it/9842/.

[15] A. S. d. Garcez, K. B. Broda, and D. M. Gabbay. In *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2012.

[16] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[17] I. Habernal and I. Gurevych. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179, April 2017. DOI: 10.1162/COLI_a_00276. URL: https://aclanthology.org/J17-1004.

[18] S. Kok, P. Singla, M. Richardson, P. M. Domingos, and M. S. H. Poon. The alchemy system for statistical relational ai: user manual. In 2007.

[19] P. Kordjamshidi, D. Roth, and K. Kersting. Systems ai: a declarative learning based programming perspective. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 5464–5471, Stockholm, Sweden. AAAI Press, 2018. ISBN: 9780999241127.

[20] K. Krippendorff. *Content Analysis An Introduction to Its Methodology*. Sage publications, 2018.

[21] J. Lawrence and C. Reed. Argument Mining: A Survey. *Computational Linguistics*, 45(4):765–818, January 2020. ISSN: 0891-2017. DOI: 10.1162/coli_a_00364. URL: https://doi.org/10.1162/coli%5C_a%5C_00364.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. DOI: 10.1109/5.726791.

[23] M. Lippi and P. Frasconi. Prediction of protein β-residue contacts by Markov logic networks with grounding-specific weights. *Bioinformatics*, 25(18):2326–2333, July 2009.

[24] M. Lippi and P. Torroni. Argumentation mining: state of the art and emerging trends. 16(2), 2016. ISSN: 1533-5399. DOI: 10.1145/2850417. URL: https://doi.org/10.1145/2850417.

[25] D. Lowd and P. Domingos. Efficient weight learning for markov logic networks. In volume 4702, pages 200–211, September 2007. ISBN: 978-3-540-74975-2. DOI: 10.1007/978-3-540-74976-9_21.

[26] L. Lugini, C. Olshefski, R. Singh, D. Litman, and A. Godley. Discussion tracker: supporting teacher learning about students' collaborative argumentation in high school classrooms:53–58, December 2020. DOI: 10.18653/v1/2020.coling-demos.10. URL: https://aclanthology.org/2020.coling-demos.10.

[27] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. D. Raedt. Deepproblog: neural probabilistic logic programming, 2018. arXiv: 1805.10872 [cs.AI].

[28] T. Mayer, E. Cabrio, M. Lippi, P. Torroni, and S. Villata. Argument mining on clinical trials:137–148. DOI: 10.3233/978-1-61499-906-5-137.

[29] T. Mayer, E. Cabrio, and S. Villata. Transformer-based argument mining for healthcare applications:2108–2115. DOI: 10.3233/FAIA200334.

[30] H. Mittal, S. S. Singh, V. Gogate, and P. Singla. Fine grained weight learning in markov logic networks, 2015.

[31] Never retreat, never retract: argumentation analysis for political speeches:4889–4896, February 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16393.

[32] J. Pennington, R. Socher, and C. Manning. GloVe: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics, October 2014. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[33] L. d. Raedt, S. Dumančić, R. Manhaeve, and G. Marra. From statistical relational to neuro-symbolic artificial intelligence. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4943–4950. International Joint Conferences on Artificial Intelligence Organization, July 2020. DOI: 10.24963/ijcai.2020/688. URL: https://doi.org/10.24963/ijcai.2020/688.

[34] S. Richard, C. Danqi, D. M. Christopher, and A. Ng. *Reasoning with neural tensor networks for knowledge base completion*. In *Advances in Neural Information Processing Systems*. B. C. J. C., B. L., W. M., G. Z., and K. Q. Weinberger, editors. Volume 26. Curran Associates, Inc., 2013, pages 926–934. URL: https://proceedings.neurips.

cc/paper/2013/file/b337e84de8752b27eda3a12363109e80-Paper.pdf.

[35] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[36] S. Roychowdhury, M. Diligenti, and M. Gori. Regularizing deep networks with prior knowledge: a constraint-based approach. *Knowledge-Based Systems*, 222:106989, 2021. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106989. URL: https://www.sciencedirect.com/science/article/pii/S0950705121002525.

[37] L. Serafini and A. d'Avila Garcez. Logic tensor networks: deep learning and logical reasoning from data and knowledge, 2016. arXiv: 1606.04422 [cs.AI].

[38] L. Serafini and A. S. d'Avila Garcez. Learning and reasoning with logic tensor networks. In G. Adorni, S. Cagnoni, M. Gori, and M. Maratea, editors, *AI*IA 2016 Advances in Artificial Intelligence*, pages 334–348, Cham. Springer International Publishing, 2016. ISBN: 978-3-319-49130-1.

[39] P. Singla and P. Domingos. Discriminative training of markov logic networks. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 868–873, Pittsburgh, Pennsylvania. AAAI Press, 2005. ISBN: 157735236x.

[40] N. Slonim, Y. Bilu, C. Alzate, R. Bar-Haim, B. Bogin, F. Bonin, L. Choshen, E. Cohen-Karlik, L. Dankin, L. Edelstein, L. Ein-Dor, R. Friedman-Melamed, A. Gavron, A. Gera, M. Gleize, S. Gretz, D. Gutfreund, A. Halfon, D. Hershcovich, R. Hoory, Y. Hou, S. Hummel, M. Jacovi, C. Jochim, Y. Kantor, Y. Katz, D. Konopnicki, Z. Kons, L. Kotlerman, D. Krieger, D. Lahav, T. Lavee, R. Levy, N. Liberman, Y. Mass, A. Menczel, S. Mirkin, G. Moshkowich, S. Ofek-Koifman, M. Orbach, E. Rabinovich, R. Rinott, S. Shechtman, D. Sheinwald,

E. Shnarch, I. Shnayderman, A. Soffer, A. Spector, B. Sznajder, A. Toledo, O. Toledo-Ronen, E. Venezian, and R. Aharonov. An autonomous debating system. *Nature*, 591(7850):379–384, March 2021. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03215-w. URL: https://doi.org/10.1038/s41586-021-03215-w.

[41] L. Tu and K. Gimpel. Learning approximate inference networks for structured prediction, 2018. arXiv: 1803.03376 [cs.CL].

Acknowledgements

Questa tesi rappresenta solo un tassello, forse l'ultimo, del mio percorso universitario magistrale che è iniziato ben 4 anni fa in un'altra università ed anche in un'altra città. Un percorso fatto di tanta crescita, di fatica e soddisfazioni, duro lavoro e tanto divertimento, lacrime e forti risate. Ma, in particolare, un percorso fatto di persone splendide senza le quali non sarebbe stato lo stesso.

Prima di tutto, grazie ai miei genitori per avermi fatto sentire più che mai come in questi anni l'importanza di avere una famiglia che ti sostiene in qualsiasi cosa, cercando sempre di capire anche nelle situazioni difficili. Siete i miei pilastri.

Grazie a mio fratello Ivan per essere sempre presente nonostante la distanza, perché so di avere una persona su cui contare in qualsiasi momento. L'ammirazione nei tuoi confronti non si può misurare, sei secondo solo per ordine di nascita ma primo in tutto il resto.

Grazie a tutto il resto della famiglia perché è sempre un piacere ritrovarsi a passare del tempo con voi.

Grazie ai miei relatori Prof. Paolo Torroni e Prof. Marco Lippi per avermi dato la possibilità di lavorare su un argomento così stimolante, attuale e per la vostra gentilezza. Grazie ad Andrea per la sua disponibiltà a risolvere qualsiasi dubbio durante il progetto di tesi.

Grazie ad Alessandro, Gianluca e Matteo con i quali ho condiviso la maggior parte delle attività e dei progetti durante questo percorso. Le nostre chiamate la domenica mattina alle 10 per i vari progetti resteranno impresse nella mia mente.

Grazie ad Eleonora con la quale ho condiviso (putroppo) virtualmente la maggior parte di questo percorso. Grazie per aver condiviso i successi per gli esami, i dubbi sulla tesi, le paure sul futuro e, dulcis in fundo, anche il giorno della laurea. Un grande in bocca al lupo per quello che verrà.

Grazie a Vanessa che ha accompagnato la mia avventura a Bologna sin dall'inizio e con la quale ho convissuto (fortunatamente) solo i primi 4 mesi. Grazie per tutto il tempo passato insieme, gli aperitivi, le cene, le chiacchierate, le nottate (ma non le colazioni), per tutta la corrente usata per asciugarmi i vestiti e soprattutto per le cadute, il tutto sempre condito con un pizzico (so che a voi piace) di sano nervosismo.

Grazie ad Edoardo, Massimo, Tommaso, Federico, Matteo e Stu perché senza di voi quest'anno non sarebbe stato lo stesso. Grazie per tutte le serate (uniche) passate ad Irma e non, per le goliardate e le cose estremamente stupide, che quando stiamo insieme ci riescono particolarmente bene. Grazie perché è stato quasi un piacere passare le zone rosse insieme e vi porterò sempre nel cuore.

Grazie a Marco (messo per primo) e Roberto che mi accompagnano sin dall'inizio del mio percorso universitario. Grazie perché colleghi e amici come voi se ne trovano raramente e, nonostante la distanza, ogni volta che ci rivediamo è come se il tempo non fosse mai passato.

Grazie a Vincenzo e all'amicizia che ci lega. Grazie per gli scambi di opinioni sempre molto piacevoli, per i consigli, gli incoraggiamenti, le abbuffate, che spero replicheremo quanto prima qui a Bologna.

Grazie ad Antonella perché non posso non riconoscere e non ringraziarti di tutto il supporto che mi hai sempre dato durante questi anni.

Grazie a Tania e Gianluca perché è sempre bello rivedersi, quando ritorno in patria, per scambiare quattro chiacchiere e aggiornarci sulle nostre disavventure.

Grazie a Pisa che mi ha ricordato che è bene puntare in alto, ma che siamo anche esseri umani con dei limiti che vanno riconosciuti e, proprio come la

torre che la caratterizza, dobbiamo piegarci alle intemperie ma mai crollare.

Grazie a Bologna che mi ha stregato sin dal primo portico e mi ha accolto come solo lei sa fare quando ero spaesato. Grazie perché non solo hai rincarato la dose, ma mi hai dato tante possibilità di crescita, mostrandomi una visione totalmente diversa, e sono certo che me ne darai ancora.