Abstract

We present an application of neural-symbolic learning to argument mining. We use Logic Tensor Networks to train neural models to jointly fit the data and satisfy specific domain rules. Our experiments on a corpus of scientific abstracts indicate that including symbolic rules during the training process improves classification performance, compliance with the rules, and robustness of the results.

1 Introduction

Argument Mining (AM) stemmed from Natural Language Processing (NLP) and Knowledge Representation and Reasoning (Cabrio and Villata, 2018), with the goal of automatically extracting arguments and their relations from natural language texts (Lippi and Torroni, 2016). Like in most areas of NLP, deep learning has recently pushed the envelope also in AM. Yet, many challenges still stand open, as argumentation involves tasks such as reasoning, debate and persuasion, which cannot be easily addressed by deep architectures alone, sophisticated as they may be. For that reason, Galassi et al. (2019) argue that a combination of symbolic and sub-symbolic approaches could leverage significant advances in AM. They illustrate the point using two neural-symbolic (NeSy) frameworks, DEEP-PROBLOG (Manhaeve et al., 2021) and Grounding-Specific Markov Logic Networks (Lippi and Frasconi, 2009), albeit without empirical evaluations.

Unfortunately, many of the existing frameworks are under continuous development and their applications are often limited to a few case studies in a single domain. Pacheco and Goldwasser (2021) analyze existing NeSy frameworks, observing that they are not specifically designed to support a variety of NLP tasks, and critically lack of a series of important features. We shall add to the list of shortcomings a lack of support for collective classification (Sen et al., 2008). This is a fundamental feature for AM, since argument analysis is typically context-dependent, meaning that classifying each argumentative component (or relation) requires considering not only the attributes of that component or relation, but also those of other connected components and relations. To address these limitations, Pacheco and Goldwasser introduce a NeSy framework based on factor graphs called DRAIL and show its application in the AM domain. To the best of our knowledge, no other NeSy approaches to AM have been investigated so far.

The present study focuses on neural-symbolic methods for AM. Besides the in-depth contribution, its aim is to pave the way for a broader application of such methods in the NLP domain. We address AM using a different NeSy framework, namely Logic Tensor Networks. We focus on the classification of argumentative component and prediction of links between component pairs. Importantly, LTNs allow to easily decouple the symbolic and sub-symbolic parts of the model, and enable collective classification during training. Our results indicate that the introduction of logic rules improves classification performance, compliance with the rules, and robustness of the results. To the best of our knowledge, this is the first application of LTNs to NLP.

2 Logic Tensor Networks (LTNs)

Logic Tensor Networks (LTNs) (Serafini and d’Avila Garcez, 2016; Donadello et al., 2017) integrate first-order many-valued logical reasoning (Bergmann, 2008) with tensor networks (Socher et al., 2013). The framework is implemented in TensorFlow (Abadi et al., 2016), LTNs belong to the “tensorization” class of undirect NeSy approaches (De Raedt et al., 2020) which embed First-Order Logic (FOL) entities, such as constants and facts, into real-valued tensors. The framework enables to combine data-driven machine learning with background knowledge expressed through first-order fuzzy logic represen-
tions. Therefore, one can use FOL to impose soft constraints at training time and investigate properties at test time. Once trained, neural architectures can be used independently of the framework. One can also use LTNs as a verification tool, to assess the ability of any given network to respect any given property, expressed as an FOL query.

LTN variables are an abstract representation of data. They must be linked to a set of real-valued vectors, which are all the possible groundings of that variable. A single data point of this set can be represented using LTN constants. LTN functions represent operations over variables and produce real-valued vectors. The evaluation is done by a set of TensorFlow operations, e.g., a neural network, defined together with the function. LTN predicates are a special class of functions whose output is a single real value between 0 and 1, which represents the degree of truth of the predicate. They can be used to represent classes of objects as well as properties that may hold between multiple objects. The learning setting is defined in terms of LTN axioms, i.e., formulas that specify logic conditions in terms of predicates, functions, and variables and can be used to assign labels to data and to specify soft constraints. Axioms can include logical connectives (∧, ∨, ~, ⇒)\(^1\) and quantifiers (∀, ∃).

Reasoning is performed in the form of approximate satisifiability, which means that the optimization process aims to maximize the level of satisifiability of a grounded theory, by minimizing the loss function (Serafini and d’Avila Garcez, 2016).

3 Argument Mining with LTNs

We frame component classification and link prediction as two classification tasks. To address them, we define two neural networks: NNCOMP and NNLINK. The first network takes a component and produces a probability distribution over the possible component classes. The second one receives two components and outputs a single value between 0 and 1, which represents the probability of there being an argumentative link between them.

Data-driven optimization is defined through three elements for each class of both tasks: a variable, a predicate, and an axiom. The variable is associated with all the data of the training set that belong to that class. The predicate is linked to the corresponding output of our networks. The axiom combines the previous elements and defines the optimization objective.

The rule-driven optimization is defined via variables linked to all the training data and through specific axioms that express the rules. For example, to enforce the antisymmetric property of links we define two variables (\(x\) and \(y\)), associate them with all the components of the training set, and specify the following axiom:

\[
∀ x, y : LINK(x, y) ⇒ ∼ LINK(y, x) \quad (1)
\]

4 Experimental Setting

Before we describe our experimental setup, a word is in order about the implementation of LTNs we used, which does not expose APIs to easily configure some aspects of the training procedure. In the current implementation, when a predicate is defined in LTN over a set of variables, all the possible groundings of such variables are used as part of the same batch. This is necessary in order for the LTN to evaluate the predicate’s truth degree. To clarify: given two components A and B, suppose one wants to determine if A and B are linked. This means evaluating LINK(A,B). In the current implementation, A and B need to belong to the same batch. Now, if we take a third component C and we want to determine LINK(A,C), A and C need to belong to the same batch. Also, Equation 2 creates a dependency between CLAIM and LINK, thus the optimization step must also consider the value of CLAIM(A), CLAIM(B) and CLAIM(C) alongside the value of the two LINK predicates. Since this applies to any pair of components, eventually all the data need to belong to the same batch. Accordingly, one cannot use mini-batches during training, which limits the scalability of the approach. Although this is not a theoretical limitation, it had a practical impact on our experimental setting, since it forced us to experiment with small-sized corpus, sentence embeddings, and neural architectures.

4.1 Data

The AbstRCT Corpus (Mayer et al., 2020, 2021) consists of 659 abstracts of scientific papers regarding randomized control trials for the treatment of specific diseases. The corpus includes three topical datasets: neoplasms, glaucoma, and mixed. The first one is divided into training, test, and validation splits, while the others are designed to be tests sets. The corpus contains about 4,000 argumentative components divided into two classes: EVIDENCE and CLAIM. Out of nearly 25,000 possi-
ble pairs of components that belong to the same
document, about 10% are connected through a di-
rect link. Claims only point to other claims. See
Appendix A for further details.

Sentence embeddings are created from 25-size
pre-trained GloVe embeddings (Pennington et al.,
2014), by averaging over the words of the sentence.
This simple method yields a low-dimensional rep-
resentation with no need to train new embeddings
or to rely on dimensionality reduction techniques.

4.2 Method
To evaluate whether the use of symbolic rules
within a neural model benefits argument mining
tasks, which is the aim of this work, we compare
the results obtained by two different models, that
differ only in the way they are trained. NEURAL is
the model trained in the usual way, i.e., by only ex-
ploring its sub-symbolic component. NESY is the
model obtained by training the same architecture
using also LTN axioms. We did not include compar-
isons with other state-of-the-art neural-symbolic ar-
chitectures, because we could find none that could
be taken off-the-shelf and used in our experiments.

For the NEURAL approach, we use three predi-
cates, corresponding to the classes of the dataset:
\textsc{Link, Evidence}, and \textsc{Claim}. For the
NESY approach, we include axioms reflecting prop-
erties of the corpus, stipulating that (i) no symmet-
ric link can exist (Eq. 1), and (ii) claims can be
linked only to other claims (Eq. 2). The latter ax-
iom connects the two tasks, thus inducing a joint-
learning setting. Appendix B lists in detail all the
LTN entities involved in our training setting.

\begin{equation}
\forall x, y : \text{\textsc{Link}}(x, y) \land \text{\textsc{Claim}}(x) \Rightarrow \text{\textsc{Claim}}(y)
\end{equation}

To avoid overfitting, we early-stop the process
using the F1 score of link prediction on the valida-
tion set, with a patience of 1,000 epochs. We focus
on link prediction because it is considered the most
challenging task, and arguably the one that would
benefit the most from the introduction of rules.

We evaluate the two models along the following
dimensions:

- **Performance**: we measure the F1 metrics re-
garding link prediction and component classi-
fication, to assess whether the rules improve
the performance of the models;
- **Robustness**: we compute the degree of agree-
ment between the networks, to assess if the
use of rules increases robustness against the
intrinsic randomness of the training process;
- **Compliance**: we test whether the prediction
of the models respects the desired properties.

4.3 Architecture
The aforementioned issues with the current LTNs
implementation and our limited computational re-
sources prevented us from integrating LTNs with
NLP state-of-the-art models. However, we can still
operate a meaningful comparison between NEU-
RAL and NESY, all else being equal. Accordingly,
we define a simple network composed of three
stacked fully-connected layers followed by a soft-
max classification layer. To obtain more robust
results with respect to the non-deterministic ele-
ments of the training procedure (Goodfellow et al.,
2016), we follow Galassi et al. (2021) and train
an ensemble of 20 networks both for \textsc{NNComp}
and \textsc{NnLink}, and evaluate the aggregated output.
We implemented and compared two aggregation
methods. Majority voting (MAJ) is a common one.
However, it provides a categorical output, prevent-
ing a probabilistic interpretation of the prediction.
Our alternative method is the average of the output
of the networks (AVG). That, however, is known to
be vulnerable to outliers.

4.4 Results
Table 1 summarizes the results of our experiments.\footnote{Since our focus is on evaluating the effect of rules, in our tables we did not include the performance results of state-of-
the-art approaches, as these figures would be misleading.}

For the classification tasks, we report the macro-
F1 score for component classification and the F1
score for the link class. Agreement is measured by
Krippendorff’s $\alpha$, while the degree of truth of
the properties is given as the ratio between the number
of instances where the clause holds and the number
of instances where only its left-hand side holds.

As far as the AM tasks, the difference between
the MAJ and AVG approaches is negligible in the
NESY setting, while it is more evident in the NEU-
RAL setting for link prediction, where the majority
voting achieves better performance.

As expected, rules seems to especially benefit
link prediction, where the networks trained with
rules perform consistently better than those trained
without. Conversely, the latter perform marginally
We presented the first application of LTN to NLP, allowing changing either of them without any direct impact on the other, except for the definition of key concepts such as the predicates/labels of the problem. Such a modularity may be highly beneficial in the context of AM, where one could use the same neural architecture with different corpora by expressing different symbolic rules. Indeed, the structural diversity of datasets and labeling schemes is a known issue in AM research, often leading to tailored solutions (Lippi and Torroni, 2016).

Performance-wise, the introduction of two symbolic rules increased link prediction performance without hindering component classification performance, whereas it boosted robustness and largely improved compliance. While the networks used in our experiments are much simpler than state-of-the-art models, and clearly they do not achieve comparable performance, we speculate that rules may benefit advanced models as well. However, it is difficult to determine how much improvement can carry over from weak baselines to stronger models (Denkowski and Neubig, 2017).

On the down side, we shall remark that one major challenge for this kind of approaches is scalability to larger domains, and the fact that they are not specifically designed for NLP tasks, so their development is yet in its infancy.

As future work, we are considering the weighting of soft rules, so as to distinguish between rules expressing preferences (or theories) and those expressing constraints. Another direction regards the recognition of properties that are not explicit in the training data but can be defined through logical rules. This could allow the network to infer information regarding components or relations without labeled training data: for example, finding which claim is the major claim of a document, or which components agree with each other.

### Table 1: Percentage scores obtained on the AbstRCT corpus. For classification and compliance, we report both the result obtained by the MAJ approach (before the dash) and by the AVG approach (after the dash).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Split</th>
<th>Approach</th>
<th>Classification</th>
<th>Agreement</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Comp.</td>
<td>Link</td>
<td>Comp.</td>
</tr>
<tr>
<td>Neoplasm</td>
<td>Val.</td>
<td>NEURAL</td>
<td>83 - 84</td>
<td>42 - 41</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESy</td>
<td>84 - 85</td>
<td>44 - 43</td>
<td>81</td>
</tr>
<tr>
<td>Neoplasm</td>
<td>Test</td>
<td>NEURAL</td>
<td>79 - 80</td>
<td>34 - 31</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESy</td>
<td>79 - 78</td>
<td>35 - 35</td>
<td>79</td>
</tr>
<tr>
<td>Glaucoma</td>
<td>Test</td>
<td>NEURAL</td>
<td>82 - 82</td>
<td>45 - 43</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESy</td>
<td>81 - 82</td>
<td>47 - 45</td>
<td>75</td>
</tr>
<tr>
<td>Mixed</td>
<td>Test</td>
<td>NEURAL</td>
<td>81 - 81</td>
<td>38 - 34</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESy</td>
<td>81 - 80</td>
<td>39 - 40</td>
<td>76</td>
</tr>
</tbody>
</table>
References


A Corpus and architectural choices

The scalability issues have influenced most of our choices in terms of experimental setting.

- Needing low-dimensionality features, we used GloVe embeddings rather than more advanced methods because it offers low-dimensional (25-size) pre-trained word embedding.
- Due to the limited computational power we could afford, we had to rule out state-of-the-art architectures with millions of parameters. We have thus used simple neural architectures with a much lower number of parameters.
- The choice of the AbstRCT corpus was motivated by the following reasons: 1) it comes from real-world documents, as opposed to data elicited or subject to guidelines; 2) it offers multiple test sets, allowing a general evaluation; 3) all the documents have an argumentative structure; 4) the argumentative model presents a specific domain-related rule; 5) it fits our computational requirements because the training set contains a limited number of documents; 6) it has been presented and published in two high-quality venues.

The AbstRCT corpus is available at https://gitlab.com/tomaye/abstrct. Its composition is reported in Table 2. Some of the documents of the neoplasm and glaucoma test set are also included into the mixed set.

We applied GloVe embeddings, downloaded from https://nlp.stanford.edu/projects/glove/, directly on the words of the documents, without pre-processing. We used random embeddings for OOV words.

We use a neural network made of three stacked fully-connected layers of size 10, 20, and 10, followed by a softmax classification layer with two outputs: CLAIM or EVIDENCE for NNCOMP and LINK or NOLINK for NNLINK. We use ReLU as activation function, and employ dropout with probability $p = 0.4$ after each layer. The two models have 712 (NNCOMP) and 962 (NNLINK) trainable parameters.

B LTN training entities

Figure 1 lists the LTN entities involved in our training setting.

C Infrastructure and Runtime Details

All our experiments run on an ASRock Z370 Pro4 motherboard, GeForce GTX 1080 Ti GPU, Intel Core i7-8700K @ 3.70GHz CPU.

Using the baseline approach, the average training time for each network is less than one minute. Using our NeSy approach, the average training time for each network is 14 minutes, with a standard deviation of about 3 minutes. Inference on the whole ensemble of 20 networks in all the considered test datasets and approaches requires less than 30 seconds.

D Deeper Architecture

To better investigate whether the application of LTNs may impact also more complex architectures, we have repeated the experiments using deeper networks with 11 layers instead of 3, and more than 3,000 parameters instead of less than 1,000. Due to the aforementioned scalability limitations, we used only roughly 60% of the training documents.

It is important to remark that the less training data may heavily impact performance, possibly favoring the NeSy setting, and therefore it is not possible to draw definitive conclusions based on these results. It is also important to underline that these architectures are nowhere near the complexity of Transformers-based architectures, such as the 100M parameters of BERT models.

The results are reported in Table 3. Overall, the use of the NeSy setting has a positive impact on the networks, both for the classification task and the respect of the properties, coherently with our previous experiments. The main differences concerns robustness, which is negatively impacted both for component classification and link prediction. The other difference is limited to the Glaucoma test set, where the NeURAL setting achieves better scores for Eq. 1.
VARIABLES: # in LTN variable names are preceded by "?"
- ?train_claim # components belonging to the claim class
- ?train_premise # components belonging to the premise class
- ?train_data # all the training components
- ?train_data2 # all the training components
- ?train_links # pairs of components that are linked
- ?train_nolinks # pairs of components that are not linked

FUNCTIONS:
- concat # given two components, merges them to obtain a pair

PREDICATES:
- CLAIM
- PREMISE
- LINK

DATA-BASED AXIOMS: # used in all the setting
- for all ?train_claim: CLAIM(?train_claim)
- for all ?train_premise: PREMISE(?train_premise)
- for all ?train_nolinks: ~LINK(?train_nolinks)

RULE-BASED AXIOMS: # used only in the NeSy setting
- for all ?train_data, ?train_data2:
  - LINK(concat(?train_data, ?train_data2)) ->
  - ~LINK(concat(?train_data2, ?train_data))
- for all ?train_data, ?train_data2:
  - LINK(concat(?train_data, ?train_data2)) & CLAIM(?train_data) ->
  - CLAIM(?train_data2)

Figure 1: LTN objects involved in our training setting.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Split</th>
<th>Approach</th>
<th>Classification</th>
<th>Agreement</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Comp.</td>
<td>Link</td>
<td>Comp.</td>
</tr>
<tr>
<td>Neoplasm</td>
<td>Val.</td>
<td>NEURAL</td>
<td>80 - 74</td>
<td>41 - 35</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESY</td>
<td>81 - 80</td>
<td>46 - 37</td>
<td>53</td>
</tr>
<tr>
<td>Neoplasm</td>
<td>Test</td>
<td>NEURAL</td>
<td>78 - 74</td>
<td>33 - 26</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESY</td>
<td>79 - 77</td>
<td>33 - 28</td>
<td>52</td>
</tr>
<tr>
<td>Glaucoma</td>
<td>Test</td>
<td>NEURAL</td>
<td>80 - 76</td>
<td>39 - 34</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESY</td>
<td>80 - 75</td>
<td>42 - 37</td>
<td>52</td>
</tr>
<tr>
<td>Mixed</td>
<td>Test</td>
<td>NEURAL</td>
<td>80 - 77</td>
<td>35 - 30</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NESY</td>
<td>82 - 80</td>
<td>39 - 29</td>
<td>54</td>
</tr>
</tbody>
</table>

Table 3: Percentage scores obtained on the AbstRCT corpus using a deeper architecture and a smaller training set. For classification and compliance, we report both the result obtained by the MAJ approach (before the dash) and by the AVG approach (after the dash).