Neural Unification for Logic Reasoning over Natural Language

Gabriele Picco[*]
Hoang Thanh Lam[*]
Marco Luca Sbodio
Vanessa Lopez Garcia
IBM Research / Europe
`gabriele.picco@ibm.com`
`{t.l.hoang, marco.sbodio, vanlopez}@ie.ibm.com`

Abstract

Automated Theorem Proving (ATP) deals with the development of computer programs being able to show that some conjectures (queries) are a logical consequence of a set of axioms (facts and rules). There exists several successful ATPs where conjectures and axioms are formally provided (e.g. formalised as First Order Logic formulas). Recent approaches, such as (Clark et al., 2020), have proposed transformer-based architectures for deriving conjectures given axioms expressed in natural language (English). The conjecture is verified through a binary text classifier, where the transformers model is trained to predict the truth value of a conjecture given the axioms. The RuleTaker approach of (Clark et al., 2020) achieves appealing results both in terms of accuracy and in the ability to generalize, showing that when the model is trained with deep enough queries (at least 3 inference steps), the transformers are able to correctly answer the majority of queries (97.6%) that require up to 5 inference steps. In this work we propose a new architecture, namely the *Neural Unifier*, and a relative training procedure, which achieves state-of-the-art results in term of generalisation, showing that mimicking a well-known inference procedure, the *backward chaining*, it is possible to answer deep queries even when the model is trained only on shallow ones. The approach is demonstrated in experiments using a diverse set of benchmark data. The source code is available at this location[1].

1 Introduction

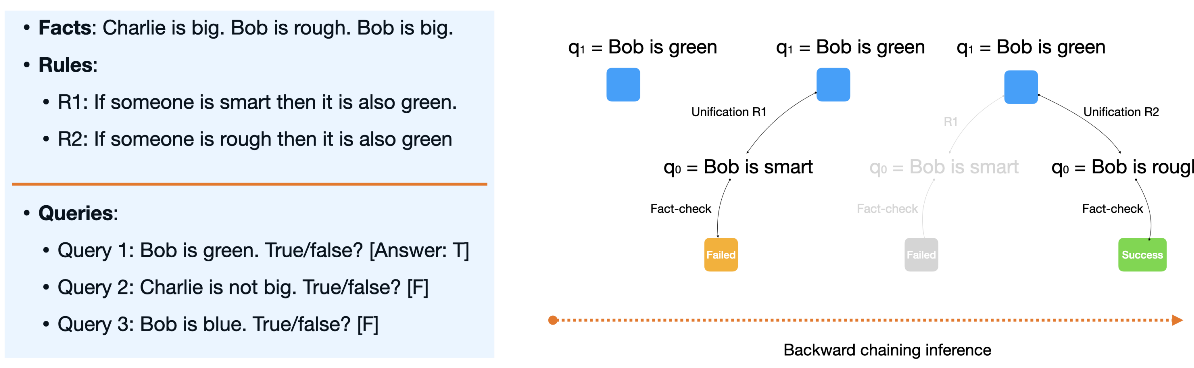
Automated Theorem Proving (ATP) deals with the development of computer programs being able to show that some conjectures (queries) are a logical consequence of a set of axioms (facts and rules) (Sutcliffe et al., 2004). This problem has wide applications in many domains, including problem solving (Green, 1981) and question answering (MacCartney and Manning, 2007; Furbach et al., 2010; Hermann et al., 2015; Clark et al., 2020), and is being actively studied, an extensive reference can be found in Loveland (1986) and Nawaz et al. (2019). Recent approaches, such as RuleTaker (Clark et al., 2020), uses transformers (Vaswani et al., 2017) as automated theorem prover over queries, facts and rules expressed in natural language (English). The theorem proving problem is translated into a binary text classification problem, where the transformers model is trained to predict the truth value (True/False) of a textual query q given an input knowledge base κ consisting of textual facts and rules.

This class of ATP is especially interesting since it does not require the explicit translation of axioms and conjecture to formal logical (e.g First Order Logic) or probabilistic rules, making it possible to reason on knowledge expressed verbatim. Furthermore, these models do not specify an explicit reasoning procedure, but learn to implicitly demonstrate a query from example instances during the learning phase. Figure 1 shows an example of an instance of the logic reasoning problem in natural language illustrated in Clark et al. (2020).

The results reported in Clark et al. (2020) demonstrated that state-of-the-art pretrained language models, such as ROBERTA (Devlin et al., 2019) or BERT (Liu et al., 2019), can be fine-tuned with labeled data to achieve appealing results both in terms of accuracy and in the ability to generalize, showing that when the model is trained with deep enough queries (at least 3 inference steps), the transformers are able to correctly answer the majority of queries (97.6%) that require up to 5 inference steps. This interesting result holds not only for training and test data in the same domain, but also for zero-shot testing on texts in other domains.

In this paper we propose an architecture that is

[*]Equal contribution.
[1]`https://github.com/IBM/Neural_Unification_for_Logic_Reasoning_over_Language`


- **Facts**: Charlie is big. Bob is rough. Bob is big.
- **Rules**:
  - R1: If someone is smart then it is also green.
  - R2: If someone is rough then it is also green

- **Queries**:
  - Query 1: Bob is green. True/false? [Answer: T]
  - Query 2: Charlie is not big. True/false? [F]
  - Query 3: Bob is blue. True/false? [F]

Figure 1: A machine reasoning problem with an informally represented knowledge and query, and an example of backward chaining inference to prove a query statement: "Bob is green."

able to answer deep queries (having large inference depths) even when it is trained only with shallow queries at depth 1 or 2. Our main assumption is that by inducing a neural network to mimic some elements of an explicit general reasoning procedure, e.g. the backward chaining, we can increase the ability of the model to generalize. In particular we focus on mimicking the iterative process in which, at each step, a query is simplified by unifying it with an existing rule to create a new but simpler query for further checking (Baader and Snyder, 2001). In a unification step, when the query matches with the consequent (*Then* clause) of a rule, the antecedent (*If* clause) of the rule is combined with that query via symbolic substitution to create a new query. For example, for the query "Bob is green" shown in Figure 1, the following steps lead to the answer (proof):

- Fact checking step 0: No fact in our knowledge base matches with the query "Bob is green"
- Unification step 1: Given "Bob is green" and the rule: "If someone is smart then it is also green.", a new query is created "Bob is smart"
- Fact checking step 2: No fact matches with the new query "Bob is smart"
- Unification step 3: Given that "Bob is green" and the rule: "If someone is rough then it is also green", a new query "Bob is rough" is created
- Fact checking step 4: "Bob is rough" matches with a fact in the knowledge base, the proof completes and returns the answer: "Bob is green" is a true statement.

As we can see in the given example, the query "Bob is green" is simplified iteratively with the help of the unification steps and is transformed into a factual query "Bob is rough", which is then checked by the fact-checking step via a simple look-up in the knowledge base. These sequences of inference steps are the basis of the famous backward chaining inference in formal logic (Russell and Norvig, 2010) illustrated in Figure 1.

The main building blocks of such inference methods are the unification and the fact checking algorithms. While backward chaining inference with formal representation can be formulated as a tree search problem (Russell and Norvig, 2010), emulating these algorithms for textual input data using neural networks is still an open research problem, mainly due to the ambiguity in mapping entities and relations expressed in natural language to corresponding mentioning entities and relations in free-text knowledge bases.

In the following sections we describe the Neural Unifier architecture, that mimics the unification and the fact checking algorithms, in order to improve generalisation on answering deep queries. We test our approaches with publicly available datasets where the Neural Unifier is trained with depth-1 or depth-2 queries, and demonstrate that it can answer queries at higher depths with high accuracy (up to five inference steps). In particular, the proposed approach achieved state-of-the-art results in these benchmark datasets and outperformed the state-of-the-art algorithms with a significant margin.

2 Preliminaries and problem definition

This section provides a formal problem definition and introduces the main intuition of the approach used for mimicking the backward chaining.

The backward chaining algorithm, described extensively in (Russell and Norvig, 2010), is one of the most used algorithms for reasoning with inference rules: it is based on a depth-first strategy to explore the search space, and it generates a proof including a sequence of unification and fact checking operations. An example of execution of this algorithm is shown in Figure 1.

Let q denote a the query (conjecture) and κ be the knowledge base that consists of a set of rules R and a set of facts F; In this work we consider all q, R, and F expressed in natural language, where queries, facts and rules can be very simple lexicalizations of logical formulas (e.g, Figure 1); or they can be paraphrased in a more creative way. Experiments with both simple lexicalizations and paraphrases are reported in Section 4.3 and Section 4.5 respectively. Let q_n denote as depth-n query that requires at least n inference steps in order to provide an answer. For example, query 1 in Figure 1 is a depth-1 query because it requires a unification with R_2, plus a fact checking step (shown as the success path in Figure 1).

The fact checking function denoted as f can now be formalised as the operation that takes as input a depth-0 query q_0 and returns True if the corresponding fact is present in the given κ, and False otherwise:

$$f(q_0, \kappa) = \begin{cases} True & q_0 \in \kappa \\ False & q_0 \notin \kappa \end{cases}$$

The unification operation denoted as u can be formalised as the operation that takes as input a query q_n and the set of facts and rules κ and provides as output a simpler query q_{n-1} at depth-(n-1):

$$u(q_n, \kappa) = q_{n-1}$$

while the application of k unification steps consecutively is denoted as $u^k(q_n, \kappa) = q_{n-k}$, and for the special case where $n = k$ we have that:

$$u^n(q_n, \kappa) = q_0$$

Let now assume the existence of a perfect unification operator denoted as u_*, that is when it explores the search tree defined by backward chaining, always chooses the branch corresponding to the optimal path (where an optimal path for a query q_n is a series of unification operations leading to a query q_0 with the same truth value of q_n in n steps). Considering a relaxed version of the problem where the queries do not require closed or open word assumption to prove, the truth value of a query q_n can therefore be found with n unification steps plus a fact checking operation: $f(u^n_*(q_n, \kappa), \kappa)$.

With a symbolic representation, the unification and the fact checking operations can be done via explicit mathematical transformations. However, when the input is represented in natural language without an explicit structure, it requires machines to learn these tasks by examples under the presence of language ambiguity. In this work, we propose a neural network architecture called Neural Unifier (NU) aiming at learning to approximate the function $f(u^n_*(q_n, \kappa), \kappa)$ with input expressed in natural language. Details about our approach are discussed in the next section.

3 Training Procedure

With a slight abuse of notation, in order to simplify the discussion, when we use q to denote the query, we also refer it as a notation of the embedding vector of the query, because a textual query in a neural network is represented as an embedding vector. The main idea behind the approach presented is an architecture composed of two units trained in two separate phases:

1. The first unit is the **Fact-checking Unit (FU)**: it approximates the fact checking operator $f(q_0, \kappa)$. The model is pre-trained in a supervised manner on only depth-0 queries and the related κ. After this initial training phase the FU weights are frozen and the model is used solely for predictions.

2. The second unit is the **Unification Unit (UU)**: it is trained in a second subsequent phase and the goal is to approximates the $u^n_*(q_n, \kappa)$ operator. The model is trained on depth-n queries (with $n > 0$) to produce an embedding vector q_0. The embedding vector q_0 is then fed as an input to the pretrained FU unit whose output prediction (True or False) is used for back-propagating the error in the NU model.

While the first phase of training is rather intuitive, the second phase teaches the unification unit, starting from a query q_n, to transform it into a vector embedding q_0 such that the neural fact checking unit can predict the correct truth value for the query q_n. Since the FU unit is pretrained to perform fact-checking tasks, the UU unit is forced to produce a query q_0 from a complex query q_n so that the answer given by the FU unit is correct, hence

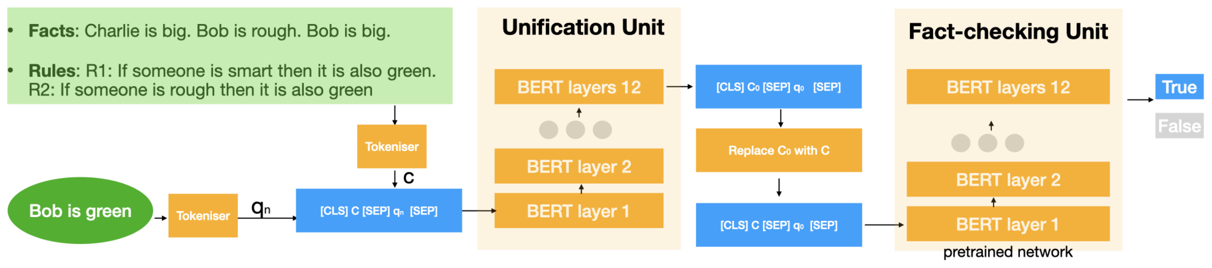


Figure 2: A Neural Unification network consists of a pretrained fact-checking unit and a unification unit. The fact-checking unit is a model trained to check whether an embedding vector of depth-0 query q_0 is true/false given a knowledge base embedding vector C. The unification unit takes an embedding vector q_n of a depth-n query and an embedding vector C of the knowledge base as an input. It transforms the input vector q_n, thanks to the multi-attention layers, to predict an embedding vector q_0 such that the pretrained fact look-up unit can make a correct prediction of the query's label.

mimicking the unification operations. The detailed implementation and input/output of the two units are described in detail in the following subsections.

3.1 Neural fact checking unit

Figure 2 illustrates the core components of our proposed Neural Unifier (NU) architecture: the Fact checking Unit (FU) and the Unification Unit (UU). The FU component is implemented through a standard Bert transformers model (Devlin et al., 2018) with a binary classification head. The inputs during the training phase are textual tuples containing the set of facts and rules κ (concatenated in a single string), and a related depth-0 query. The Bert tokenizer is used to transform κ into the corresponding embedding vector denoted as C (C stands for "context") and the depth-0 query into its embedding vector representation q_0. The output of the tokenization step therefore follows the format: $[CLS]\ C\ [SEP]\ q_0\ [SEP]$ where $[CLS]$ and $[SEP]$ are embedding vectors of special tokens added by the BERT tokenizer to separate context and query (Devlin et al., 2018).

The transformer model is then fed with the tokenized input and fine-tuned to output `True` if the query is a correct conjecture with respect to the set of textual facts and rules provided in the input, `False` otherwise. Note that after the first learning phase the weights of the FU unit are frozen when the UU unit is being trained.

3.2 Neural unification unit

In our implementation, we also use a Bert transformers as a unification unit, with the only difference that in this case the output is an embedding vector (while the neural fact checking unit has a binary classification head and output True/False labels). The inputs for the unification unit are textual tuples containing the set of facts and rules κ, and the related depth-n query ($n > 0$).

The Bert tokenizer is used to transform κ into the corresponding embedding vector C and the depth-n query into the corresponding embedding vector q_n. The output of the tokenization step therefore follows the format: $[CLS]\ C\ [SEP]\ q_n\ [SEP]$. This input is fed into the transformers to get the output (the hidden states of the transformers last layers) in the format $[CLS]\ C_0\ [SEP]\ q_0\ [SEP]$. Where C_0 and q_0 are the transformations of C and q_n via transformers respectively.

3.3 Wiring unification and fact checking

Figure 2 shows the complete architecture when two units are wired into one network. In particular, the output of the UU unit is fed into the FU unit. However, in our implementation, instead of using $[CLS]\ C_0\ [SEP]\ q_0\ [SEP]$ as a direct input to the fact checking unit, we replace C_0 by the original context embedding C, hence the corrected input to the FU unit is $[CLS]\ C\ [SEP]\ q_0\ [SEP]$. In doing so the unification unit can focus on optimizing the prediction of the query embedding q_0 rather than trying to reconstruct the original context. We observed in experiments that this approach simplifies the learning process and helps converging faster to the optimal solutions. Detailed examples of inputs and outputs can be found in the appendix.

4 Experiments and Results

This section discusses experimental settings and results.

4.1 Datasets and experimental settings

We used three datasets provided in (Clark et al., 2020) to validate our approach. These datasets were selected because they contains test queries that require up to 5 inference steps, where we can validate the induced generalization capability of our approach. The three datasets are:

- Rule reasoning data: synthetically created from a synthetic knowledge base (see (Clark et al., 2020) for more information about the data creation process). It consists of 5 folders with depths ranging from 0 to 5. We only use depth-0 training queries in the depth-0 folder for training the fact-checking unit. Depth-1 and depth-2 training queries in the corresponding folders are used for training the unification models. The folders depth-3, depth-4 and depth-5 are used as holdout sets only for testing purposes.
- Paraphrased data: created from rule reasoning data where the questions, facts and rules are paraphrased by crowd-workers. Paraphrased dataset contains more complex and longer sentences, such as: *"Alan is young and green, and seems to be cold and rough, but time will round him into a decent person"* (see (Clark et al., 2020) for details). We use this dataset to test the zero shot generalisation capability to a larger variety of more natural linguistic forms.
- Electricity data: synthetically created from a set of rules on an electrical circuit, describing the conditions for an appliance to function. Contains queries that require up to 4 inference steps (see (Clark et al., 2020) for details).

All datasets, with the exception of the *Electricity data*, are divided into training, validation and testing sets. More details can be found in the appendix.

4.2 Methodology

In all experiments, we used BERT (specifically *bert-base-uncased*) as our backbone model both for the FU and the UU unit. For both training phases, described in Section 3, we used the Adam optimisation algorithm (Kingma and Ba, 2014) with logloss (Vovk, 2015), we set the mini-batch size as 8 to fit our GPU memory and manually fine-tuned the learning rate in the range $[10^{-6}, 10^{-2}]$, choosing the best learning rate by looking at the accuracy of the prediction in the validation set during training (0.0001). It is important to notice that the test sets used for reporting experimental results are different from the validation sets used for hyper-parameter optimization to ensure that the comparison is fair. In all the training we used early stopping technique on the validation for avoiding overfitting, by setting the maximum number of epochs to 20.

4.2.1 Neural Fact-checking unit (FU)

The FU unit used in the experiments is trained on 58,844 depth-0 queries (all depth-0 training queries in the depth-0 folder of *Rule reasoning data*). The training task on the *Rule reasoning data* turns out to be particularly simple, after a few epochs (around 3 using the early stopping strategy) the model is able to solve the task perfectly on the training and validation set and the report an accuracy on the set (1,6751 queries) is close to one (0.99968). The FU also achieves an accuracy of 0.71 and 0.99 on the respective *Paraphrased data* (2,968 queries) and *Electricity data* (2,812 queries) test sets without any further fine-tuning (zero shot setting). Achieving high accuracy in the fact-checking unit is particularly important as the subsequent UU training assumes the FU prediction as ground truth to backpropagate the gradient.

4.2.2 Neural unification unit

Since our work focuses on learning a general inference mechanism on shallow queries and applying it to solve queries at higher depths (up to 5 inference steps), we report the experiments on two variants of the Neural Unifier, trained on queries with a maximum depth of 2:

- NU (D = 1): that is a Neural Unifier network composed of an UU unit trained on 27429 depth-1 queries (all depth-1 training queries in the depth-1 folder of *Rule reasoning data*) and the previously trained FU as described in section 4.2.1
- NU (D = 2): that is a Neural Unifier network composed of a UU unit trained on 14254 depth-2 queries (all depth-2 training queries in the depth-2 folder of *Rule reasoning data*) and the previously trained FU as described in section 4.2.1

Both the models try to induce the inference mechanism, using the procedure described in section 3. NU (D = 1) reduces 1-depth queries to 0-depth equivalent vector embedding and then use the fact checker to derive the truth value of the original query. NU (D = 1) only observes queries at depth 0 and 1 during the training phase. NU (D = 2) instead turns depth-2 queries to depth-0 equivalent vector embedding and then uses the fact checker to prove the conjecture. Therefore NU (D = 2) only observes queries at depth 0 and 2 during the training phase. In the following sections, the results of the two models will be reported (trained with the settings described in subsection 4.2), focusing on the generalization capacity to queries at depths not observed during training.

4.3 Inference on queries deeper than those observed at training time

Table 1 reports the performances of the NU $D = 1$ and NU $D = 2$ models and compares them with the state-of-the art RuleTaker approach. There are two version of RuleTaker in our experiments: RT is our implementation that uses *bert-base* as backbone architecture, while RR corresponds to the implementation of (Clark et al., 2020) with *roberta-large* and is reported for completeness when the results are available in the original paper.

As can be seen in this experiment, the NU models with $D = 1$ and $D = 2$ accurately answer queries at unseen depths, and consistently outperforms the state-of-the-art approaches on those depths. The significant result is particularly evident for NU $D = 2$ over depth-5 queries. More interesting is that the model NU $D = 2$ not only learns to transform depth-2 queries to depth-0 equivalent vector embedding, but it can reduce a q^n queries with depths ranging from 3 up to 5 to depth-0 equivalent vector embedding effectively. Our hypothesis is that the transformers-based architecture, which in several applications has been shown to efficiently learn recursive tasks (Vaswani et al., 2017), effectively approximates the unification operator $u^n_*(q_n, \kappa)$ with its multi-layer architecture described in section 2.

Although our implementation uses Bert, several transformers can be used successfully while maintaining the properties analyzed, as demonstrated in Clark et al. (2020) and the additional experiments are reported in the appendix.

4.4 Inference on provable queries deeper than those observed at training time

Observing the table 1, it may be counter-intuitive that the reported accuracy for NU (D = 1) and NU (D = 2) increases as the depth of the queries increases. This fact can be explained by observing a bias present in the distribution of the queries that does not have a proof, and its truth value is assigned

Table 1: Accuracy on the Rule reasoning test sets when the depths of the test queries are varied. NU $D = 1$ and NU $D = 2$ are compared with our implementation of the-state-of-the-art RuleTaker (RT) approach with *bert-base-uncased* back-bone and the original implementation of the RuleTaker (RR) with *roberta-large* back-bone pretained on the RACE dataset as reported in Clark et al. (2020).

Rule reasoning data						
Depth	NU D = 1	RT D <= 1	RT(RR) D <= 1	NU D = 2	RT D <=2	RT(RR) D <= 2
0	46.2	**99.7**	100	34.4	**99.9**	100
1	51.0	**94.2**	99.0	44.0	**95.0**	98.8
2	**55.2**	40.3	36.8	64.9	**87.4**	98.9
3	**59.6**	30.7	23.1	**79.2**	72.4	71.1
4	**61.6**	19.3	11.4	**88.0**	59.7	43.4
5	**66.0**	17.6	12.3	**95.0**	53.0	37.2

based on the closed word assumption (CWA). We call those queries CWA while the other ones, which have at least one successful proof, are called provable queries in the test set.

Table 2: Distribution of CWA queries in the test data.

Rule reasoning: CWA and provable query distribution						
Depth	D = 0	D = 1	D = 2	D = 3	D = 4	D = 5
# queries	6299	4434	2915	2396	2134	2003
CWA (%)	69.2	56.3	33.6	19.2	9.3	3.4

Table 2 shows the distribution of CWA and provable queries in our test data. The statistics shows an inversely proportional relationship between the number of CWA questions and the accuracy of the models reported in Table 1, thus suggesting that NU network are especially effective on provable queries, while they do not work so well on queries that does not have a proof (CWA).

This assumption is verified by testing the models on the subset of provable queries, as reported in Table 3.

These results show that NU (in particular NU $D = 2$) is able to answer provable queries, at all depths very accurately. Based on this observation, in the following subsections, we will demonstrate that by combining NU with RuleTaker to form an ensemble model, we are able to outperform each individual model for both CWA and provable queries.

4.5 Zero-shot generalization

In order to verify the ability to generalize to deep unseen provable queries, we test the NU ($D = 2$) on the provable queries in the Paraphrased and

Table 3: Accuracy of NU $D = 1$, NU $D = 2$, RT $D <= 1$ and RT $D <= 2$ on the provable queries in the Rule reasoning test sets (queries that does not require closed word assumption to prove the conjectures).

	Rule reasoning data (provable queries)			
Depth	NU D = 1	RT D <= 1	NU D = 2	RT D <= 2
0	30.4	**100**	94.6	**99.9**
1	**96.4**	95.7	**98.2**	93.3
2	**38.3**	15.6	**98.2**	91.6
3	**34.5**	19.5	**99.3**	85.9
4	**28.6**	15.6	**98.7**	81.1
5	**30.6**	18.7	**98.4**	74.4

Electricity datasets, without fine-tuning the model (zero shot setting).

The results, reported in Table 4, shows that NU ($D = 2$) can effectively outperform the current state-of-the-art by a significant margin in both accuracy and generalisation capability.

Table 4: Accuracy of NU $D = 2$ and RT $D <= 2$ on the subset of provable queries in the Paraphrased and Electricity test sets (queries that does not require closed word assumption to prove the conjectures).

	Paraphrased Data (provable queries)		Electricity Data (provable queries)	
Depth	NU D = 2	RT D <= 2	NU D = 2	RT D <= 2
0	**100**	25.3	98.9	**100**
1	**99.6**	10.2	82.8	**87.9**
2	**100**	9.2	79.8	**90.9**
3	**100**	9.0	**77.7**	25.9
4	**100**	13.9	**100**	77.7
5	**100**	0	**-**	-

4.6 Weighted Ensemble methods

While sections 4.3 and 4.5 show the effectiveness of the NU approach on provable queries, we also propose a linear weighted ensemble approach that combines the prediction of both NU and the RuleTaker. The key idea behind the ensemble method is that NU demonstrated working very well for provable queries while the RuleTaker was very good at answering CWA queries. Therefore, by combining these approaches we are able to handle both types of queries effectively. In order to choose the weights, we tune them to optimize the accuracy on the available validation sets for each depth. The results is reported on a separate test set.

Table 5 reports the results of the ensemble approach (W-NU-RT) on the test data. It can be seen that the weighted ensemble of NU $D = 2$ and RT $D <= 2$ effectively leverages the advantages of the two approaches to answer both CWA and provable queries at all depths effectively. The ensemble method outperforms the RuleTaker in both datasets and at most depths.

Table 5: Accuracy of the weighted ensemble of NU ($D = 2$) and RT ($D <= 2$) on the Synthetic and Paraphrased test sets.

	Rule reasoning data		Paraphrased data	
Depth	W-NU-RT D <= 2	RT D <= 2	W-NU-RT D <= 2	RT D <= 2
0	**99.9**	**99.9**	**74.8**	69.2
1	90.9	**95.0**	**61.1**	60.3
2	84.7	**87.4**	**66.8**	33.6
3	**84.9**	72.4	**93.0**	7.4
4	**89.6**	59.7	**95.7**	5.6
5	**95.0**	53.0	**92.3**	7.6

4.7 Significance tests

We highlight below some significant results obtained with statistical tests:

- Table 3: the model NU $D = 2$ has significantly better (p-value 0.020 computed with randomization test) results than RT $D <= 2$ (previous state-of-the-art) on provable queries.
- Table 4: the model NU $D = 2$ has significantly better (p-value 0.002 computed with randomization test) results than RT $D <= 2$ (previous state-of-the-art) on paraphrased provable queries.
- Table 5: W-NU-RT has significantly better (p-value 0.008 computed with randomization test) results than RT on paraphrased data (columns 3 and 4 of table 5) without compromising its performance on rule-reasoning data (columns 1 and 2 of table 5).

We also highlight an aspect not evident from the statistical tests, our proposed model outperforms in all experiments, at depth 3,4,5 not seen during training, previous state-of-the-art.

5 Related work

While our work is, to the best of our knowledge, the first proposed architecture that emulates backward chaining inference over rule sets and facts expressed in natural language, there are several methods which explore this research area. (Clark et al., 2020) introduce the use of transformers to reason over explicitly stated rule sets expressed in natural language; their approach show that transformers are able to solve the problem with high accuracy when the neural network is trained with sufficiently deep reasoning paths, without imposing any structure on the neural reasoning. Our main contribution with respect to (Clark et al., 2020) consists in emulating, through a neural network, a general reasoning mechanism inspired by the backward chaining algorithm used in formal logic programming. Also, our method demonstrate better accuracy for high depth queries, even when trained only with shallow queries.

(Saha et al., 2020) and (Tafjord et al., 2020) modify the (Clark et al., 2020) approach in order to generate proof together with the predicted truth value, these methods however require the explicit knowledge of the proof during the training phase.

Furthermore, our work is substantially different from the methods that focus on an initial translation of knowledge expressed in textual form to a formal specification, with the aim of applying classic reasoning algorithms, such as the architecture proposed in (Singh et al., 2020) for translating text into first order logic formulas. Our work is also different from (Socher et al., 2013): the authors present a neural network suitable for reasoning over relationships between two entities of a knowledge base, focusing specifically on predicting additional true facts using only vector representations of existing entities in the knowledge base. Other approaches have combined neural and symbolic reasoning methods. One notable example is the Neural Theorem Proving (NTP) presented in (Rocktäschel and Riedel, 2017). The authors propose an end-to-end differentiable prover, operating on symbolic representations, for automated completion of a knowledge base: they recursively construct neural networks to prove queries on the knowledge base by following Prolog's backward chaining algorithm. Additionally, they introduce a differentiable unification operation between vector representations of symbols. (Minervini et al., 2020) describes an NTP capable of jointly reasoning over KBs and natural language corpora. Although the method is versatile, explicit mapping to entities in the KB is required. Other relevant methods implement forms of neural reasoning starting from a formal knowledge base, including (Serafini and d'Avila Garcez, 2016), (Guha, 2014), and (Dong et al., 2019), or starting from an ontology (which usually define not just the predicates, but also rules) (Hohenecker and Lukasiewicz, 2017). Conversely, we focus on using transformers both as a fact look-up model (over a knowledge base expressed in natural language), and as a unification unit for transforming queries, which may require many steps of inference, into factual queries that can be answered with the fact look-up model. Some early work on simulating the first-order algorithm of unification using neural networks is presented in (Komendantskaya, 2011). The author shows how error-correction learning algorithm can be used for the purposes of unification. However, this work considers a version of the problem where the knowledge is represented using a formal first order logic language, and uses an explicit mapping of each symbol of the language into a input vector. Similarly to our work, (Weber et al., 2019) approach the problem of reasoning over natural language emulating unification. They present a model combining neural networks with logic programming for solving multi-hop reasoning tasks over natural language. In the proposed approach the authors extend a Prolog prover to use a similarity function over pretrained sentence encoders. A substantial difference with respect to our work is that (Weber et al., 2019) approach requires the transformation of natural language text into triples (by using co-occurrences of named entities), and then embedding representations of the symbols in a triple using an encoder.

In this paper, we take inspiration from the method presented in (Hudson and Manning, 2018), a recurrent cell that simulates a reasoning step, although the architecture is specially designed for performing visual reasoning given a textual query.

Our goal is quite different from answering complex multi-hop questions using a corpus of documents as a virtual knowledge base as proposed in the recent work by (Dhingra et al., 2020), which requires selecting spans from paragraphs of texts. Our work can be described as the formalization of a model and a training process that leads the neural network emulate a backward chaining inference process for answering deep queries.

6 Conclusion and Future Work

In this paper we have shown (in a limited, but not trivial setting) that machines can be trained to perform deep reasoning over language, even if trained only on shallow reasoning. The presented approach performs inference without the need for a translation phase from natural language to a formal specification, and it obtains high accuracy on the datasets considered. Furthermore, with a particular learning architecture that brings learning closer to a deductive argument form, help improving the ability to generalize to deep queries. Although this work is a step in the direction of combining the ability of neural networks to emulate reasoning on non-formal data with the explanatory power of a formal demonstration procedure, further work is needed to fill the gaps. In an ideal situation, a machine should perform n inference steps (with explicit reference to the parts of the text concerned) to answer a query with depth n. Moreover, the reasoning procedure should be able to reason on any possible textual expression of rules or facts, excluding ambiguous and irrelevant information. With further advances, we may potentially be able to:

- Understand if there exists a relationship between the output embedding of the Neural Unification unit and an interpretable representation.
- Apply the Neural Unifier approach on other types of logical inference (e.g. inductive and abductive) on a different type of datasets, for example with an open word assumption.
- Complement the answer to a deep query, produced by our Neural Unification unit, with a (possibly approximate) formal and human interpretable proof of the answer and identify the parts of the text involved in the n inference steps that led to a conclusion. One approach could be to modify the architecture by explicitly requesting evidence as input, in line with the ideas presented in (Saha et al., 2020) and (Tafjord et al., 2020).

References

Franz Baader and Wayne Snyder. 2001. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *Accepted for publication in IJCAI.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. 2020. Differentiable reasoning over a virtual knowledge base. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. 2019. Neural logic machines. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ulrich Furbach, Ingo Glöckner, and Björn Pelzer. 2010. An application of automated reasoning in natural language question answering. *Ai Communications*, 23(2-3):241–265.

Cordell Green. 1981. Application of theorem proving to problem solving. In *Readings in Artificial Intelligence*, pages 202–222. Elsevier.

Ramanathan Guha. 2014. Towards a Model Theory for Distributed Representations. *AAAI Spring Symposium - Technical Report*, SS-15-03:22–26.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28:1693–1701.

Patrick Hohenecker and Thomas Lukasiewicz. 2017. Deep learning for ontology reasoning. *CoRR*, abs/1705.10342.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ekaterina Komendantskaya. 2011. Unification neural networks: unification by error-correction learning. *Logic Journal of the IGPL*, 19(6):821–847.

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

D W Loveland. 1986. Automated theorem proving: Mapping logic into ai. In *Proceedings of the ACM SIGART International Symposium on Methodologies for Intelligent Systems*, ISMIS '86, page 214–229, New York, NY, USA. Association for Computing Machinery.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200.

Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. 2020. Differentiable reasoning on large knowledge bases and natural language. In Ilaria Tiddi, Freddy Lécué, and Pascal Hitzler, editors, *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, volume 47 of *Studies on the Semantic Web*, pages 125–142. IOS Press.

M. Saqib Nawaz, Moin Malik, Yi Li, Meng Sun, and M. Ikram Ullah Lali. 2019. A survey on theorem provers in formal methods.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30*, pages 3788–3800.

Stuart Russell and Peter Norvig. 2010. *Artificial intelligence: a modern approach*, 3 edition. Prentice Hall.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. In *EMNLP*.

Luciano Serafini and Artur S. d'Avila Garcez. 2016. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In *Proc. of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy'16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016)*, volume 1768.

Hrituraj Singh, Milan Aggrawal, and Balaji Krishnamurthy. 2020. Exploring Neural Models for Parsing Natural Language into First-Order Logic.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.

Geoff Sutcliffe, Jürgen Zimmer, and Stephan Schulz. 2004. *TSTP Data-Exchange Formats for Automated Theorem Proving Tools*, pages 201–215.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2020. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *arXiv preprint arXiv:2012.13048*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Vladimir Vovk. 2015. The fundamental nature of the log loss function. *CoRR*, abs/1502.06254.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. Nlprolog: Reasoning with weak unification for question answering in natural language. In *Proc. of 57th Conference of the Association for Computational Linguistics, ACL*, volume 1, pages 6151–6161.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A Input/Output of the Neural Unifier's units

This section provides a detailed example of the input and output of the core units of the architecture.

A.1 Neural fact checking unit

A.1.1 Training phase

- Input: (κ, q_0),

  where κ is the set of facts and rules concatenated in a single string and q_0 is the depth-0 query, for example:

  $$\kappa = \text{"Bob is big.Gary is not cold. ..."}$$

  $$q_0 = \text{"Bob is big?"}$$

  In the training phase, the input of the fact checking unit is furthermore tokenized and transformed into a numerical vector (using the BERT embedding layer) that follows the format: $[CLS]\ C\ [SEP]\ Q_0\ [SEP]$,

  where C is the embedding of κ and Q_0 is the embedding of q_0. $[CLS]$ and $[SEP]$ are embedding vectors of special tokens added by the BERT tokenizer to separate context and query (Devlin et al., 2018).

- Output: $(True/False)$

A.1.2 Inference phase

When the fact checking unit is used for training the neural unification unit or in the inference phase (both with frozen weights), the input skips the tokenization phase.

- Input: $[CLS]\ C\ [SEP]\ NU_O\ [SEP]$,

  where C is the embedding of κ and NU_O is the vector embedding given in output by Neural unification unit.

- Output: $(True/False)$

A.2 Neural unification unit

- Input: (κ, q_n),

  As for the other unit, the input of the unification unit is furthermore tokenized and transformed in the corresponding BERT embeddings, following the format: $[CLS]\ C\ [SEP]\ Q_n\ [SEP]$,

  where C is the embedding of κ and Q_n is the embedding of q_n. $[CLS]$ and $[SEP]$ are embedding vectors of special tokens added by the BERT tokenizer to separate context and query (Devlin et al., 2018).

- Output: $[CLS]\ C_0\ [SEP]\ NU_O\ [SEP]$,

  where C_0 is the embedding vector in output corresponding to the tokens of the given input context.

  As explained in the paper, C_0 is replaced with C in the embedding, before fed it to the fact checking unit.

B Datasets

Tables 6, 7, 8 and 9 report some additional statistics of the three datasets used. Detailed examples of dataset instances are shown in (Clark et al., 2020).

Table 6: Distribution of CWA queries in the train data in folder 1 (F=1).

Rule reasoning: CWA and provable query distribution (train F=1)						
Depth	D = 0	D = 1	D = 2	D = 3	D = 4	D = 5
# queries	41203	27429	1020	225	69	13
CWA (%)	56	33.9	100	100	100	100

Table 7: Distribution of CWA queries in the train data in folder 2 (F=2).

Rule reasoning: CWA and provable query distribution (train F=2)						
Depth	D = 0	D = 1	D = 2	D = 3	D = 4	D = 5
# queries	34199	21093	14254	322	77	30
CWA (%)	63.0	40.1	11.4	100	100	100

Table 8: Distribution of CWA queries in the Paraphrased test data.

Paraphrased data: CWA and provable query distribution (test)						
Depth	D = 0	D = 1	D = 2	D = 3	D = 4	D = 5
# queries	2968	2406	1443	1036	142	13
CWA (%)	67.5	59.9	33.1	6.9	4.2	7.6

Table 9: Distribution of CWA queries in the Electricity test data.

Electricity data: CWA and provable query distribution (test)						
Depth	D = 0	D = 1	D = 2	D = 3	D = 4	D = 5
# queries	2812	1392	736	234	96	-
CWA (%)	64.8	61.7	80.4	76.9	81.2	-

C Results with different types of transformers

Besides BERT, we also tried to use ROBERTA and BERT fine-tuned with the scale adversarial dataset for grounded commonsense inference (Zellers et al., 2018). The results are illustrated in Table 10, which shows that our results are not specific to BERT, but instead our approach works well also for other types of models.

Table 10: Results on the rule reasoning and the paraphrased datasets with different types of transformers used as the basis of the NU neural network

	Rule reasoning data			Paraphrased data		
Depth	NU BERT D = 2	NU Roberta D = 2	NU SWAG D = 2	NU BERT D = 2	NU Roberta D = 2	NU SWAG D = 2
0	34.4	**44.8**	32.2	32.6	**44.7**	32.4
1	44	**45.8**	43.6	40	**42.3**	39.9
2	64.9	**65.7**	65.5	66.7	65.3	**66.8**
3	79.2	79	**80.1**	**93**	88.2	92.9
4	88	87.4	**89.9**	95	91.5	**95.7**
5	95	92.3	**95.4**	92.3	92.3	92.3

D Runtime information and computing infrastructure

The experiments reported in the paper were performed on a cloud cluster with a Tesla v100 GPU, 16 GB of RAM and SSD. The training of the fact checking unit on this instance takes less than 60 minutes, while the training of the unification unit takes less than 240 minutes (4 hours). The inference times are less than two minutes for all datasets.