Ontology Learning in the Deep

Giulio Petrucci[1,2], Chiara Ghidini[1], Marco Rospocher[1]

[1] FBK-irst, Via Sommarive, 18, I-38123 Trento, Italy
[2] University of Trento, Via Sommarive, 14, I-38123 Trento, Italy
`{petrucci,ghidini,rospocher}@fbk.eu`

Abstract. Recent developments in the area of deep learning have been proved extremely beneficial for several natural language processing tasks, such as sentiment analysis, question answering, and machine translation. In this paper we exploit such advances by tailoring the ontology learning problem as a transductive reasoning task that learns to convert knowledge from natural language to a logic-based specification. More precisely, using a sample of definitory sentences generated starting by a synthetic grammar, we trained Recurrent Neural Network (RNN) based architectures to extract OWL formulae from text. In addition to the low feature engineering costs, our system shows good generalisation capabilities over the lexicon and the syntactic structure. The encouraging results obtained in the paper provide a first evidence of the potential of deep learning techniques towards long term ontology learning challenges such as improving domain independence, reducing engineering costs, and dealing with variable language forms.

1 Introduction

Along the years, the ontology engineering community has been pursuing the ambitious goal to automatically encode increasingly expressive knowledge from text. Even if the fully automatic acquisition of logic-based knowledge is still a long term goal, several approaches have been proposed, achieving remarkable performances but, at the same time, still experiencing some limitations. In a nutshell, approaches that rely on controlled languages such as Attempto [8] pose rigid limitations on the syntax that the text has to adopt, thus constraining the text producers to express knowledge in pre-defined formats and moreover making the approach unsuitable to process already written text. Approaches that instead target expressive knowledge (that is, complex axiom) extraction from written text (e.g., LExO [17]) heavily rely on catalogs of hand-crafted lexico-syntactic patterns that are *rigid* w.r.t. the grammatical structure of the text they can process. The downside of these approaches is that extending a catalog of hand-crafted rules to handle the variability of natural language can be particularly hard, taking also into account that language forms evolve over time, can be domain-specific, and that patterns need to be specifically produced and adapted to the different mother tongues.

In the last years, deep neural networks have been successfully exploited in several Natural Language Processing (NLP) tasks, from the most foundational, like part-of-speech tagging or semantic role labeling, to the most complex ones, like question-answering, sentiment analysis, and statistical machine translation. Such systems have the advantage to be cheap in terms of feature engineering and can learn how to deal with language variability in a flexible manner.

Stemming from such experiences, we approached ontology learning as a machine transduction task, as described in Section 3. We train statistically — i.e. by examples — and in an end-to-end fashion a neural network based system to translate definitory text into Description Logic (DL) [1] formulae. The main contributions of our work are:

- a **general architecture** that enables to formulate the ontology learning problem as a machine transduction task exploiting Recurrent Neural Networks (Sections 3 and 4). To the best of our knowledge, such approach has never been applied before to ontology learning tasks;
- a **dataset** for a statistical learning based formulation of the OWL complex axioms learning task (Section 5.1). The creation of an extensive dataset was necessary as, to the best of our knowledge, no commonly accepted, large-size, dataset exists for this task. Its availability can facilitate the adoption of statistical learning approaches within the ontology learning community;
- a **customisation** of the general architecture for the specific dataset, and its evaluation (Section 5.2). The evaluation shows that our model manifest the capability to generalise over sentence structure, as well as, tolerance to unknown words, as discussed in Section 6.

Related works are presented in Section 2, and concluding remarks are provided in Section 7. While our work was performed and evaluated on the specific language model defined by our dataset, the results illustrated in the paper provide a first evidence that deep learning techniques can contribute in a fruitful manner to the ontology engineering community to tackle some of its long term challenges, especially in domain adaptation or extension.

2 State of the Art

In this section, we briefly review the state of the art in ontology learning, with a special focus on those approaches that aim to extract expressive knowledge. For a wider overview, see [5, 9, 14].

As pointed out in [16], state-of-the-art methods are *able to generate ontologies that are largely informal*, that is, *limited in their expressiveness.* Indeed, tasks like taxonomy construction or factual knowledge extraction (i.e., assertions, entity recognition, and so on) can be largely automatized, thanks to the effectiveness reached by such methods. The success of OWL as the *de facto* standard ontology language for the web paved the way for some advances in automatic extraction of more expressive knowledge, such as terminological axioms or complex class definitions. Some methods and tools have been developed along the years to tackle these problems.

A first collection of tools is based on Controlled Natural Language. The best known one is Attempto Controlled English (ACE) [8], a restricted version of standard English, both in terms of syntax and semantics. By adopting a small set of construction and interpretation rules, sentences written in ACE can be automatically translated in some formal language, including OWL. However, its applicability for ontology learning from generic, available texts is limited, as arbitrary sentences are typically not written in ACE. Furthermore, it works only for English language.

A second collection of tools is based on catalogs of hand-crafted lexico-syntactic patterns. A well known system in this category is LExO [17]. The main idea behind LExO is to transform a set of natural language definitions into a set of OWL axioms as a suggestion to the ontology engineer. Each definition is parsed into a dependency and then transformed into a set of OWL axioms by the application of a set of hand-crafted rules over the syntactic features of the text. The approach has been used in [16] as the foundational idea to build a methodology with the ambitious goal to cover the whole ontology life-cycle management. In particular, a machine learning based approach to determine disjointness of two classes is presented, based on the measurement of their *taxonomic overlap*, i.e. how much is likely an instance of both of them to exist in the ontology: the lower this value, the more two classes are likely to be disjoint. Such metric is estimated starting from their mutual distance in some background taxonomy (like WordNet[3]), the similarity between their lexical contexts, their Pointwise Mutual Information (PMI) over the web and their matching a set of predefined lexico-syntactic patterns. Regarding limitations, pattern-based approaches for ontology learning are *rigid* w.r.t. the grammatical structure of the text they can process: hence, several linguistic phenomena such as conjunctions, negations, disjunctions, quantifiers scope, ellipsis, anaphora, etc., can be particularly hard to parse and interpret. Extending a catalog of hand-crafted rules to handle all such phenomena can be an extremely expensive task, leading to unsustainable costs in engineering, maintaining and evolving the system.

A different approach is taken in [10], where a system for the extraction of $\mathcal{EL}$++ concepts definitions from text is presented. Text fragments involving concepts from the SNOMED CT[4] ontology are matched and their lexical and ontological features are used to train a maximum entropy classifier to predict the axiom describing the involved entities. User feedback can be exploited to adaptively correct the underlying model. However, this approach is tightly bounded to the specific domain considered.

Summing up, state-of-the-art methods for expressive knowledge extraction still experience severe limitations: the approach we propose in this work aims to overcome some of them.

[3] https://wordnet.princeton.edu/
[4] http://www.ihtsdo.org/snomed-ct/

3 Ontology Learning as a Transduction task

The main intuition underpinning our approach is that ontology learning can be seen as a *transduction* process, as defined in [7]: a string from a source language is converted into another string of a target language. In our specific case, we want to convert a sequence of words, that is a *sentence* in natural language, into a sequence of logical symbols, namely a *formula*. As an example, let us consider the following sentence:

"A bee is an insect that has 6 legs and produces honey." (1)

This sentence provides a brief description of a bee and of its main characteristics and can be encoded by means of the following DL formula which, so-to-speak, represents its conversion into the required logical format:

$$\texttt{Bee} \sqsubseteq \texttt{Insect} \sqcap = 6\ \texttt{have.Leg} \sqcap \exists \texttt{produce.Honey} \quad (2)$$

One of the problems we have in converting natural language sentences into logical formulae is the variability of natural language. Consider for instance:

"If something is a bee, then it is an insect
with exactly 6 legs and it also produces honey." (3)

Despite being lexically and syntactically different from (1), (3) provides a description of a bee which is, from the standpoint of an ontological formulation, equivalent to the one provided in (1). Thus, we would like to be able to transform it into the same formula (2). A second, dual problem we have is the fact that sentences often share a similar structure while conveying a completely different meaning. An exemplification of this problem is provided by the sentence:

"A cow is a mammal that has 2 horns and eats grass". (4)

This sentence shares several similarities with sentence (1), and would be translated into a DL formula such as:

$$\texttt{Cow} \sqsubseteq \texttt{Mammal} \sqcap = 2\ \texttt{have.Horns} \sqcap \exists \texttt{eat.Grass} \quad (5)$$

which is, from a structural point of view, very similar to the one in (2). These two phenomena, which can be considered analogous to well known problems of homography (same term different meaning) and synonymity (different terms same meaning) are recurrent when dealing with descriptions (of the world) and the meaning they denote. Being able to deal with them both is one of the challenges that expressive ontology learning has to face. Our approach in dealing with these two problems relies on the following observations:

1. Nouns like `Bee`, `Insect`, `Leg` and `Honey` denote the involved *concepts* while verbs like `have` and `produce` describe relationships among such concepts, or *roles*. From a linguistic standpoint, nouns and verbs – together with adjectives and some adverbs – are *content words*, as they describe the actual

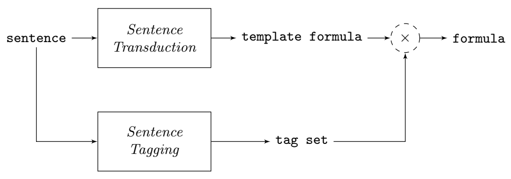


Fig. 1. The whole pipeline

content of a sentence. Dually, words such as articles, pronouns, conjunctions, most adverbs, and so on, are considered *function words*, as they express grammatical relationships between words and carry almost no lexical meaning.

2. In the case of different sentences that convey ontological equivalent formulations, such as (1) and (3), we notice that content words are *the same*,[5] while function words change. In the dual case of sentences with similar structures and different meanings (such as (1) and (4)), we note that the sentences share the same function words while content words are radically different.

Stemming from these two observations, our idea is to deal with these two problems by splitting the transduction process of sentences in two parallel phases: a **sentence transduction** phase and a **sentence tagging** phase, as depicted in Figure 1. The sentence transduction phase focuses on the identification of the logical structure of the formula corresponding to the natural language specification. The output of this phase is a structure that we name *formula template*. The sentence tagging phase focuses on identifying and recognising all the words for what they act in the sentence: a concept, a role, a number, or a generic word.

Going back to our example, the output of the sentence transduction phase for sentences (1), (3), and (4) is the (same) formula template:

$$\mathtt{C}_0 \sqsubseteq \mathtt{C}_1 \sqcap =\mathtt{N}_0\ \mathtt{R}_0.\mathtt{C}_2 \sqcap \exists \mathtt{R}_1.\mathtt{C}_3 \tag{6}$$

where C, R and N have been respectively used for concepts, roles and numbers, with a subscript indicating their order. Sentence transduction is therefore the phase which tackles the challenge of identifying a common unifying structure among different natural language sentences that may differ both lexically and syntactically. The output of sentence tagging would instead produce three different structures:

$$\begin{array}{l} A\ [bee]_{\mathtt{C}_0}\ is\ an\ [insect]_{\mathtt{C}_1}\ that\ [has]_{\mathtt{R}_0}[6]_{\mathtt{N}_0}\ [legs]_{\mathtt{C}_2}\ and \\ \quad [produces]_{\mathtt{R}_1}\ [honey]_{\mathtt{C}_3} \end{array} \tag{7}$$

$$\begin{array}{l} If\ something\ is\ a\ [bee]_{\mathtt{C}_0}\ is\ an\ [insect]_{\mathtt{C}_1}\ that\ [has]_{\mathtt{R}_0}\ exactly \\ \quad [6]_{\mathtt{N}_0}\ [legs]_{\mathtt{C}_2}\ and\ it\ also\ [produces]_{\mathtt{R}_1}\ [honey]_{\mathtt{C}_3} \end{array} \tag{8}$$

$$\begin{array}{l} A\ [cow]_{\mathtt{C}_0}\ is\ a\ [mammal]_{\mathtt{C}_1}\ that[has]_{\mathtt{R}_0}\ [4]_{\mathtt{N}_0}\ [legs]_{\mathtt{C}_2}and \\ \quad [eats]_{\mathtt{R}_1}\ [grass]_{\mathtt{C}_3} \end{array} \tag{9}$$

[5] Possibly after resolving anaphora, coreference, or other linguistic phenomena

where all the words other than C, R, or N are intended to be tagged with a generic word label, w. This step is close to the slot filling problem, as tackled in [11]: we need to detect the role each word assumes within a certain semantic scope, even a whole sentence. The final step in the pipeline is to combine the outputs of the two phases to obtain the resulting logical formula. Thus, the formula template (6) combined with the tagged sentence (7) will provide formula (2) as a logical conversion of (1); the formula template (6) combined with the tagged sentence (8) will also provide formula (2) as a logical conversion of sentence (3); and finally the formula template (6) combined with the tagged sentence (9) will provide formula (5) as the logical conversion of sentence (4). In the next section we illustrate in detail how RNNs are used to implement the sentence transduction and sentence tagging phases.

4 An RNN-based Architecture for Ontology Learning

In this section we will provide a brief overview of our computational model for ontology learning, describing the main intuitions behind it. A detailed description of the theoretical framework is provided in [15].

4.1 RNNs and the Gated Recursive Unit model

Speaking of neural networks, the adjective *recurrent* referred to one of its layers, means that the activation of the layer at time t, say $\mathbf{h}^{\langle t \rangle}$, depends not only on the inputs, say $\mathbf{x}^{\langle t \rangle}$, but also on its previous value, $\mathbf{h}^{\langle t-1 \rangle}$, as in:

$$\mathbf{h}^{\langle t \rangle} = g(\mathbf{x}^{\langle t \rangle}, \mathbf{h}^{\langle t-1 \rangle}; \theta), \tag{10}$$

where g is the so called *cell function* and θ is the set of function parameters to be learnt during the training phase. Dealing with natural language, the t-th timestep is the t-th word in a sentence. The recurrent structure makes such class of models an adequate choice when dealing with sequences, and in particular with natural language, where each word depends on the previous ones.[6] To handle long-term dependencies – syntactic dependencies, speaking of natural language – the cell function can be endowed with some memory effect. Different models have been proposed. Among them, the Gated Recursive Unit (GRU), synthetically depicted in Fig. 2, shows good memory capabilities combined with a higher simplicity w.r.t. other cell functions. The cell (layer unit) state is controlled by two gates: the *reset gate* $\mathbf{r}$ and the *update gate* $\mathbf{z}$, according to the equations in Fig. 3. Intuitively, the update gate balances the amount of information to be kept from the previous state; at the same time such memory can be erased when the reset gate approaches zero. Such structure is repeated for each word. The

[6] The goal of our work is to show that the ontology learning task can be tackled using neural network based models trained in a end-to-end fashion. Assessing the best neural network architecture to implement statistical learning for this task is beyond the scope of our work.

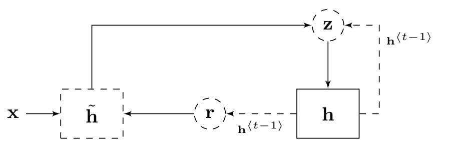


Fig. 2. Gated Recursive Unit

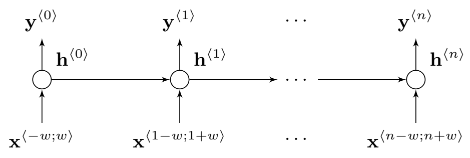
$$\mathbf{r}^{\langle t\rangle} = sigmoid(\mathbf{W}_r\mathbf{x}^{\langle t\rangle} + \mathbf{U}_r\mathbf{h}^{\langle t-1\rangle})$$
$$\mathbf{z}^{\langle t\rangle} = sigmoid(\mathbf{W}_z\mathbf{x}^{\langle t\rangle} + \mathbf{U}_z\mathbf{h}^{\langle t-1\rangle})$$
$$\tilde{\mathbf{h}}^{\langle t\rangle} = tanh(\mathbf{W}_h\mathbf{x}^{\langle t\rangle} + \mathbf{r} \odot \mathbf{U}\mathbf{h}^{\langle t-1\rangle})$$
$$\mathbf{h}^{\langle t\rangle} = \mathbf{z} \odot \mathbf{h}^{\langle t-1\rangle} + (\mathbf{1} - \mathbf{z}) \odot \tilde{\mathbf{h}}^{\langle t\rangle}$$

Fig. 3. GRU equations

model parameters to be learnt during the training are $\theta = [\mathbf{W}_r, \mathbf{U}_r, \mathbf{W}_z, \mathbf{U}_z]$. The symbol $\odot$ indicate the Hadamar product.

4.2 Network Model for Sentence Tagging

The sentence tagging task can be formulated as follows: given a natural language sentence corresponding to some formal representation, we want to apply a tag to each word. The tag identifies the role the word has in the formal representation. We used a regular RNN, a snapshot of which is depicted in Fig. 4. Each word is represented by its index within the vocabulary. The most straightforward representation of such index as a vector would be a one-hot vector: the i-th word in a vocabulary of $|V|$ words will be a vector of $|V|$ components, all with value zero but the i-th, with value 1. Such vector representation is impractical for two main reasons: (i) vectors can become huge, as their dimension is the same of the vocabulary; and (ii) it can be hard to find a meaningful way to compose vectors as they are extremely sparse. Therefore, we map each index within the vocabulary into a low-dimension vector of real numbers, called an *embedding vector*. Embedding vectors act as distributed representations trained to capture all the meaningful features of a word in the context of a given task (see [12]). Moreover, their dimension can be significantly smaller than the number of words in the lexicon, avoiding the *curse of dimensionality*, as in [2]. In our network, each index in the dictionary is associated to an embedding vector. Such vectors are learnt during the training phase. At the t-th step, we feed the network with a *window of words* of width w, i.e. a short sequence of words centered on the t-th word of the sentence. All the embedding vectors corresponding to the words in a window are concatenated in a unique vector and fed into the recurrent layer. We indicate such window as $\mathbf{x}^{\langle t-w;t+w\rangle}$. The activation of such layer is given by equation (10). The output of the recurrent layer is then fed into a linear layer that outputs, at each time step, a vector of the same size of the number of possible tags: each component holds a sort of *score* of the corresponding tag. At the timestep t, our network must predict the appropriate tag to apply to the current input word. So we apply a softmax over such scores, modeling a probability distribution across all the possible tags. The tag applied to the t-th word will be the most probable one, i.e. the *argmax* over such output vector $\mathbf{y}^{\langle t\rangle}$. The network is trained minimizing the categorical cross entropy between the expected sequence and the predicted one.



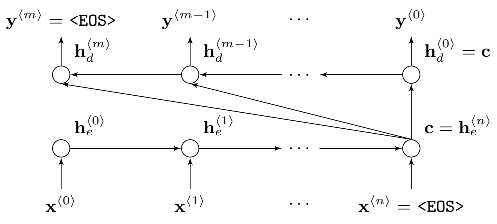
$\mathbf{y}^{\langle 0 \rangle}$
$\mathbf{y}^{\langle 1 \rangle}$
$\mathbf{y}^{\langle n \rangle}$
$\mathbf{h}^{\langle 0 \rangle}$
$\mathbf{h}^{\langle 1 \rangle}$
$\mathbf{h}^{\langle n \rangle}$
$\mathbf{x}^{\langle -w;w \rangle}$
$\mathbf{x}^{\langle 1-w;1+w \rangle}$
$\mathbf{x}^{\langle n-w;n+w \rangle}$

Fig. 4. The RNN used for sentence tagging

4.3 Network Model for Sentence Transduction

The sentence transduction task can be formulated as follows: given a natural language sentence corresponding to some formal representation, we want to identify the structure of such formal representation, how its concepts and roles are connected, and which connectors are used. The input, the word embeddings, and the output are handled like in the previous model, using single words —represented with $\mathbf{x}^{\langle t \rangle}$— instead of windows in the input layer, and with the output vector that has the same dimension of the catalogue of all the formal representation terms that can be produced. We use also the same training objective, minimizing the categorical cross entropy between the expected sentence and the predicted one.

Both this model and the previous one receive a sequence of words and turn it into another sequence of different symbols. This allows us to use the same training objective and the same training procedure for both architectures. The pivotal difference is that each output symbol of the sentence tagging model corresponds exactly to one word of the input sentence, while this is not the case in the sentence transduction task. We can deal with such situation using two stacked recurrent layers in the so called Recurrent Encoder-Decoder (RED) configuration, a snapshot of which is depicted in Fig. 5. The main intuition behind such architecture is the following: the first RNN *encodes* the input sequence, so that its hidden state at the last time step —the vector $\mathbf{c}$ in Fig. 5— holds the distributed representation *of the whole sentence*. Such vector is subsequentially fed as a constant input to a second RNN which *decodes* the content of such vector into a new sequence. In this way, the two sequences are at the same time: (i) *tightly coupled* w.r.t. their meaning, as the distributed representation of the whole input sequence is constantly fed into each decoding step; and (ii) *loosely*



$\mathbf{y}^{\langle m \rangle} =$ <EOS>
$\mathbf{y}^{\langle m-1 \rangle}$
$\mathbf{y}^{\langle 0 \rangle}$
$\mathbf{h}_d^{\langle m \rangle}$
$\mathbf{h}_d^{\langle m-1 \rangle}$
$\mathbf{h}_d^{\langle 0 \rangle} = \mathbf{c}$
$\mathbf{h}_e^{\langle 0 \rangle}$
$\mathbf{h}_e^{\langle 1 \rangle}$
$\mathbf{c} = \mathbf{h}_e^{\langle n \rangle}$
$\mathbf{x}^{\langle 0 \rangle}$
$\mathbf{x}^{\langle 1 \rangle}$
$\mathbf{x}^{\langle n \rangle} =$ <EOS>

Fig. 5. The RNN Encoder-Decoder network model for sentence transduction

coupled w.r.t. their structure, since there is no direct correspondence between the input and the output symbols. The cell function of the decoder can be written as:

$$\mathbf{h}^{\langle t \rangle} = g(\mathbf{c}, \mathbf{h}^{\langle t-1 \rangle}; \theta), \tag{11}$$

which is a slight simplification of the one used in [4]. The simplification consists in omitting the feedback of the output of the previous step. This helps us to keep the model as simple as possible, both from a conceptual and an implementation point of view (less parameters to deal with, which means a model easier to train and implement) without significant performance loss.

5 Learning expressive OWL axioms with RNNs

In this section we show how the general architecture presented in Section 4 can be deployed to learn expressive OWL axioms from natural language sentences.

5.1 A dataset for learning of OWL axioms

To the best of our knowledge, there is no commonly accepted dataset for a statistical learning-based formulation of the OWL ontology learning task, especially when focusing on complex class definitions, as we do. Such dataset would consist of pairs of natural language sentences and their corresponding DL axiom, and should be adequately large to enable the training (and evaluation) of Machine Learning based approaches for ontology learning. The creation of such a collection of data would be extremely beneficial for the community and for those, like us, aiming to exploit statistical learning based methods.

As manually building such dataset can be extremely expensive —requiring considerable human-effort to collect, annotate, and validate a large quantity of data— we followed the practice of some notable examples in literature (e.g., [3, 6, 7, 13, 18, 19, 20]) of verifying the approach over *appropriate* synthetic data.

In details, we started by verbalizing with ACE a set of OWL class definitions in order to have a first seed of definition-like sentences like (1), as typically found in encyclopedias. We extended this seed by manually adding variations of every verbalization and other equivalent structures. So, for the sentence *"all the dogs are mammals"*, we added *"every dog is a mammal"*, *"dogs are mammals"*, *"any dog is also a mammal"* and so on. Or, for *"a bass guitar has at least 4 strings"*, we added *"bass guitars have more than 3 strings"*, *"A bass guitar don't have less than 4 strings"* and so on. Finally, we built a grammar capable to generate all such sentences, with placeholders instead of *concepts*, *roles*, and numbers used in cardinality restriction clauses. All the definitions have a left hand side class description and a right hand side class description. Relations between such class descriptions can be subsumption, disjunction or equivalence. Each side can be itself made of one or two *atomic* definitions, in conjunction or disjunction, which can be a *concept* definition or a *role* definition. Roles can be bound to one or two cardinality constraints, conjuncted or disjuncted. To sum it up, the constructs used in our definitions are concepts, roles, subsumption, disjunction, negation,

Table 1. Examples of sentence templates and formula templates

sentence template	**formula template**
anything that $\mathtt{R}_0$ *at least* $\mathtt{N}_0$ $\mathtt{C}_0$ *and is also* $\mathtt{C}_1$ *, is* $\mathtt{C}_2$	$\geqslant \mathtt{N}_0\mathtt{R}_0.\mathtt{C}_0 \sqcap \mathtt{C}_1 \sqsubseteq \mathtt{C}_2$
every $\mathtt{C}_0$ *is also something that* $\mathtt{R}_0$ *less than* $\mathtt{N}_0$ *or more than* $\mathtt{N}_1$ $\mathtt{C}_1$	$\mathtt{C}_0 \sqsubseteq < \mathtt{N}_0\mathtt{R}_0.\mathtt{C}_1 \sqcup > \mathtt{N}_1\mathtt{R}_0.\mathtt{C}_1$
anything that $\mathtt{R}_0$ *less than* $\mathtt{N}_0$ *or more than* $\mathtt{N}_1$ $\mathtt{C}_0$ *is* $\mathtt{C}_1$	$< \mathtt{N}_0\mathtt{R}_0.\mathtt{C}_0 \sqcup > \mathtt{N}_1\mathtt{R}_0.\mathtt{C}_0 \sqsubseteq \mathtt{C}_1$

intersection, union, existential restriction, cardinality restrictions maybe in conjunction or disjunction.[7]

From our grammar, we generated more than 123 millions different *sentence templates*, each of which has been associated to its equivalent *formula template*. We obtained 261189 different formula templates in total.[8] Some examples of sentences and corresponding formulae are in Table 1.

5.2 Training and Evaluation

We assessed the capabilities of our approach in learning expressive OWL axioms from text, on various training-test pair configurations generated starting from the dataset described in Section 5.1.[9] Next we will describe such training-test pairs and the model configuration in terms of hyper-parameters and training settings. Finally, we will discuss the results of our experiments.

Training and Test Sets. From sentence-formula template couples described in the previous section, we can generate the actual training and test examples for our experiments. Our example is a triple $e = (s, t, f)$ made of:

1. a natural language *sentence* s, namely a sequence of words. A sample of this is sentence (1).
2. a *tag sequence* t corresponding to sentence s. Such tag sequence is obtained mapping each word of sentence s to a tag indicating the role of the word in the sentence. A sample of tag sequence is (7).
3. a *formula template* f corresponding to the translation of s in the target logical language. A sample of formula template is the one in (6).

To turn a sentence template into an actual sentence, we have to fill its placeholder with actual words. *Role* placeholders are filled with verbs, randomly selected from a list of 882. *Concept* placeholder are filled combining words from a list of 1522 adjectives, a first list of 2425 nouns and a second list of 192 nouns. Combinations were allowed only according to 66 patterns. A simplified example

[7] More precisely, the constructs considered correspond to the $\mathcal{ALCQ}$ Description Logic, a well-known, expressive extension of $\mathcal{ALC}$ with qualified number restrictions.

[8] The list of all the sentence and formula templates is available here: `https://drive.google.com/file/d/0B_FaCg6LWgw5Z0UxM2N1dTYwYkU`

[9] The several training, evaluation and test sets used in the experiments are available here: `https://drive.google.com/file/d/0B_FaCg6LWgw5ZnBkSEV0NWx2YW8`

Table 2. Patterns for concept name generation

adj.	noun#1	noun#2
cool	sword	sharpener
–	sword	sharpener
cool	–	sharpener
cool	sword	–
–	sword	–
–	–	sharpener

Table 3. From a sentence template to an example.

sent. template	*A* $\mathtt{C}_0$ *is a* $\mathtt{C}_1$ *that* $\mathtt{R}_0$ *exactly* $\mathtt{N}_0$ $\mathtt{C}_2$
sentence	*A bee is a insect that has exactly* $\mathtt{N}_0$ *legs*
tag sequence	$\mathtt{w}\ \mathtt{C}_0\ \mathtt{w}\ \mathtt{w}\ \mathtt{C}_1\ \mathtt{w}\ \mathtt{R}_0\ \mathtt{w}\ \mathtt{N}_0\ \mathtt{C}_2$
form. template	$\mathtt{C}_0 \sqsubseteq \mathtt{C}_1 \sqcap = \mathtt{N}_0\mathtt{R}_0.\mathtt{C}_2$

of such procedure is presented in Table 2. In our actual dataset, concept definitions can be more complex and involve up to 6 different words, being them nouns, adjectives or the preposition *"of"*. We applied some basic preprocessing to the text: using only *"a"* as indeterminate article, using only *"do not"* as the negative form, using only singular words (*"Bee"* for *"Bees"*) and the avoiding the third singular person verbal form (*"take"* for *"takes"*). Numbers placeholders, used only in cardinality restrictions, have not been not filled with actual numbers. This missing substitution can be seen as another text preprocessing phase for number identification, which is a task that can be performed easily with a rule based approach. All the generate sentences are *actual natural language sentences* since they are grammatically correct, even if potentially unrealistic from a human point of view – e.g. *"A smoking hurricane is also something that pump at least 18 orange stampede"*.

Transforming such a sentence into a tag sequence is straightforward: each word in the original sentence template is tagged as `w`, while the words filling a placeholder are tagged with the same symbol of the placeholder. Summing up, Table 3 reports all the elements of a training example, starting from a given sentence template. Given a set of examples $\mathcal{T} = \{(s,t,f)\}$, the set of all the pairs of sentences and formulas $\mathcal{T}_F = \{(s,f)\}$ will be a training/test set for the sentence transduction task, while the set of all the pairs of sentences and tag sequences $\mathcal{T}_T = \{(s,t)\}$ will be used for the sentence tagging task.

We generated different training sets, of different dimensions. Fixed the number of training examples, we randomly generated sentence templates from our grammar, turned each of such sample templates into a sentence, and generated the corresponding tag sequence and formula template, emulating the work of a human annotator. In this way, we could test the actual generalisation capabilities of our model from the syntactic point of view. We also randomly marked some words filling the placeholders for concepts and roles as out-of-vocabulary with the `<UNK>` word; for concepts we used the `<UNK>` symbol with 60% of probability, while for roles we used 20%. Overall, we re-scanned the whole sentence ensuring that the number of placeholder fillers turned to `<UNK>` was between 20% and 40%. In this way we could also test the generalisation capabilities of the model from the lexical point of view. We also generated an evaluation set for each training set, of the same dimension, starting from the same sentence templates, filled with different words. We used such evaluation set to check the network status during the training phase, to be sure that the network was not just memorizing

Table 4. Network parameters.

Networks params.	
# words	~ 5000
# tags	11
# terms	21
word window	5
dim. embedding	100
dim. hidden (tag.)	200
dim. enc/dec (tra.)	1000

Table 5. Training parameters.

Training parameters	
training steps	10000
batch size	50
learning algo.	AdaDelta
learning rate	2.0
ρ	0.95
ϵ	10^{-6}
GPU Card	Tesla K40
time (tag.)	~ 2.5h
time (tra.)	~ 4.5h

Table 6. Accuracy on the test set

dim.	**AT**	**AF**
1000	99.9%	96.2%
2000	99.9%	99.0%
3000	99.9%	99.6%
4000	99.9%	99.8%

the training examples. Finally, for each dataset, we built a larger test set starting from 2 millions of sentence and formula templates generated from our grammar: in this way we ensured that the test sentences are unseen during the training phase. Such templates were turned turned into the actual test set with the very same procedure followed for the training and the evaluation sets, with the slight difference of increasing the overall minimum probability of out-of-vocabulary words to 30%. Although the model used for sentence transduction is made of two stacked recurrent layers, they are jointly trained: the first layer produces an embedding of the input sentence which is then decoded into a formula template by the second. Our gold truth is this final formula template. For the sentence tagging model, the gold truth is the output tag sequence.

Experimental Setting and Results. The goal of our experiments was to assess the accuracy of a trained RNN-based architecture in learning expressive $\mathcal{ALCQ}$ axioms from typical definitory sentences. Therefore, we trained and evaluated the proposed architecture on several datasets produced following the procedure described before. For both tasks, tagging and transduction, we defined the network parameters empirically, according to some experiences in literature. We trained both the networks with AdaDelta (see [21]) for 10000 steps, with batches of 50 examples, evaluating the network against the evaluation set every 100 training steps. The network configuration, together with the training parameters and some indication of the training phase duration, are reported in Table 4 and Table 5. The dimensions of the training set (i.e. the amount of annotated examples), together with the results in terms of accuracy in tagging (AT) and in transduction (AF) are reported in Table 6.

We achieve almost 100% of accuracy for all the datasets in the tagging task and over 96% of accuracy in the transduction task, thus confirming the feasibility of building accurate RNN-based architectures that learn expressive $\mathcal{ALCQ}$ axioms from typical definitory sentences. We want to remark that none of the test sentences is in the training set, so there is no possibility for the networks to memorize the examples. We remark that in the transduction task, even if the number of possible tag sequences and formula templates is limited, our networks do not classify examples but learn how to *generate* new sequences of symbols, namely a formula template, starting from an initial natural language sentence.

6 Discussion

Reviewing the Ontology Learning state of the art presented in Section 2 we can highlight a main intuition: *syntax matters*. Pattern-based approaches, such as LExO, require to manually define rules exploiting the syntactic aspects of the text in order to extract the knowledge it carries. In our work, we pushed such intuition into a totally different direction, with a *learn by examples* approach. We trained a model that *learns* to model encyclopaedic language and its syntactic structures, it *learns* how to parse their occurrences in the text and how to translate them into corresponding logical constructs. Roughly speaking, our tagging model can be seen as a POS tagger, and our transduction model can be seen as a syntactic parser. Both of them extremely *specialized* w.r.t. the type of language of interest. Our model stores in its parameters the embedding of each words in the vocabulary (plus the `<UNK>` word), how to deal with function words, and many other syntactic constraints. Being our model trained in a end-to-end fashion, this *knowledge* —namely the features learnt by the model— remains in the subsymbolic form and is not made explicit.

Our experiments, in which we could achieve extremely high accuracy just annotating 1000 sentences, shows that statistically learning such rules is feasible. Furthermore, our contribution presents several advantages over state-of-the-art pattern-based approaches: (i) it does not require to manually define pattern rules for each possible linguistic natural language variation to be covered, something practically unfeasible; (ii) our model is trained in an end-to-end fashion, from raw text to OWL formulae, without relying on any NLP tool and requiring no feature engineering cost for the input representation; finally, (iii) being our approach purely syntactic, it does not need any domain-specific training: content words of our test set are selected randomly, showing as our model does not rely on their meaning but only on their syntactical features.

Despite being generated starting from sentences in ACE, our system can deal with language variability that goes well beyond controlled English. To confirm this, we generated 4000 random sentence templates and, from them, a set of sentences, filling the various placeholders with a simplified vocabulary compliant with the Attempto Parser Engine (APE)[10]. The APE engine could parse only 13 sentences. A qualitative analysis through the sentences not or incorrectly parsed by the APE service gave us an idea of some of the linguistic phenomena that our system can handle beyond the controlled language. We can roughly split them in two groups:

1. function words that are not parsed by the controlled language but that are actually used in natural language, such as:
   (a) *"anything"* and *"any"* acting as universal quantifier;
   (b) *"at least"*, *"no more than"* or *"exactly"* for cardinality restrictions;
   (c) *"but"* as a conjunction in cardinality restrictions, e.g. *"less than 3 but more than 10"*;

[10] `http://attempto.ifi.uzh.ch/site/resources/`

   (d) *"also"*, in the right hand side of an implication, e.g. *"if something is a mammal, it is also an animal"*;
   (e) *"everything that is"* as a quantifier;
   (f) the use of *"some"* to indicate an intersection, e.g. *"some birds are flightless"*;
   (g) *"do not"* as a negation of a role.
2. constructs that are not parsed by the controlled language but are actually used in natural language, such as:
   (a) ellipsis of the demonstrative pronoun *"that"* in conjunction or disjunctions, e.g. *"everything that has a tail and (that) is a dog, also chases cats"*;
   (b) ellipsis of the demonstrative pronoun, role and range concept in conjunction or disjunction of cardinality restrictions, e.g. *"a bass is an instruments that has at least 4 (strings) or (that has) at most 6 strings"*;
   (c) ellipsis of the adverb in cardinality restriction, instead of *"exactly"*, as in *"a bee is an insect that has (exactly) 6 legs"*;
   (d) ellipsis of the quantifier, as in *"dogs are mammals"*.

7 Conclusion and future work

In this work we presented an approach for ontology learning where an RNN-based model is trained in a end-to-end fashion to translate definitory sentences into OWL formulae. A GRUs based RE-D is used to transduce a definitory sentence into the corresponding formula template, while a GRUs based RNN maps the proper word to the proper role within such formula template. We trained and tested our approach on a newly created dataset of sentence-formula template pairs, sampling from more than 123M distinct sentence templates and more than 260K distinct formula templates. Our system achieved almost 100% of accuracy in the sentence tagging task and over 96% in the sentence transduction task starting from only 1000 training examples. While converting arbitrary natural language text to OWL is still an ambitious, out-of-reach goal, these results give evidence of the capabilities of our approach in translating definition-like sentences to (complex) OWL axioms, while showing good syntactic and lexical generalization generalization capabilities and a reduced tagging effort of 1000 sentences.

The main limitation of our work is that the model has been trained and evaluated on limited amount of data, modeling a *limited* portion of natural language in a sentence-by-sentence (i.e., one sentence is translated to one axiom) fashion. Further validations and evaluation are needed on more realistic data, showing even more language variability. Nonetheless, the encouraging results obtained in the paper pave the way to future extensions and generalizations aiming at tacking some of these limitations. In particular, in our future work we will investigate how to extend our approach (i) to handle portions of knowledge that can be spread across different sentences, overcoming sentence-by-sentence processing, and (ii) to cope with wider language variability, thus covering more realistic encyclopedic text.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. Journal of Machine Learning Resources 3, 1137–1155 (Mar 2003)
3. Bowman, S.R., Potts, C., Manning, C.D.: Recursive neural networks for learning logical semantics. CoRR abs/1406.1827 (2014)
4. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR abs/1406.1078 (2014)
5. Cimiano, P., Mädche, A., Staab, S., Völker, J.: Ontology learning. In: Handbook on Ontologies, pp. 245–267. Springer (2009)
6. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. CoRR abs/1410.5401 (2014)
7. Grefenstette, E., Hermann, K.M., Suleyman, M., Blunsom, P.: Learning to transduce with unbounded memory. CoRR abs/1506.02516 (2015)
8. Kaljurand, K., Fuchs, N.: Verbalizing OWL in Attempto Controlled English. In: OWLED 2007 (2007)
9. Lehmann, J., Voelker, J. (eds.): Perspectives On Ontology Learning. Studies in the Semantic Web, AKA / IOS Press (2014)
10. Ma, Y., Syamsiyah, A.: A hybrid approach to learn description logic based biomedical ontology from texts. In: ISWC 2014 Proceedings (2014)
11. Mesnil, G., He, X., Deng, L., Bengio, Y.: Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In: INTERSPEECH 2013. pp. 3771–3775 (2013)
12. Mikolov, T., Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: NAACL HLT 2013. pp. 746–751 (2013)
13. Peng, B., Lu, Z., Li, H., Wong, K.: Towards neural network-based reasoning. CoRR abs/1508.05508 (2015)
14. Petrucci, G.: Information extraction for learning expressive ontologies. In: The Semantic Web. Latest Advances and New Domains, pp. 740–750. Springer (2015)
15. Petrucci, G., Ghidini, C., Rospocher, M.: Using recurrent neural network for learning expressive ontologies. CoRR abs/1607.04110 (2016)
16. Völker, J., Haase, P., Hitzler, P.: Learning expressive ontologies. In: Ontology Learning and Population: Bridging the Gap between Text and Knowledge, pp. 45–69. IOS Press, Amsterdam, the Netherlands (2008)
17. Völker, J., Hitzler, P., Cimiano, P.: Acquisition of OWL DL axioms from lexical resources. In: ESWC. pp. 670–685 (2007)
18. Weston, J., Bordes, A., Chopra, S., Mikolov, T.: Towards AI-complete question answering: A set of prerequisite toy tasks. CoRR abs/1502.05698 (2015)
19. Weston, J., Chopra, S., Bordes, A.: Memory networks. CoRR abs/1410.3916 (2014)
20. Zaremba, W., Sutskever, I.: Learning to execute. CoRR abs/1410.4615 (2014)
21. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. CoRR abs/1212.5701 (2012)