Learning Embeddings to lexicalise RDF Properties

Laura Perez-Beltrachini and **Claire Gardent**
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy
F-54500, France
{laura.perez, claire.gardent}@loria.fr

Abstract

A difficult task when generating text from knowledge bases (KB) consists in finding appropriate lexicalisations for KB symbols. We present an approach for lexicalising knowledge base relations and apply it to DBPedia data. Our model learns low-dimensional embeddings of words and RDF resources and uses these representations to score RDF properties against candidate lexicalisations. Training our model using (i) pairs of RDF triples and automatically generated verbalisations of these triples and (ii) pairs of paraphrases extracted from various resources, yields competitive results on DBPedia data.

1 Introduction

In recent years, work on the Semantic Web has led to the publication of large scale datasets in the so-called Linked Data framework such as for instance DBPedia or Yago. However, as shown in (Rector et al., 2004), the basic standards (e.g., RDF, OWL) established by the Semantic Web community for representing data and ontologies are difficult for human beings to use and understand. With the development of the semantic web and the rapid increase of Linked Data, there is consequently a growing need in the semantic web community for technologies that give humans easy access to the machine-oriented Web of data.

Because it maps data to text, Natural Language Generation (NLG) provides a natural means for presenting this data in an organized, coherent and accessible way. It can be used to display the content of linked data or of knowledge bases to lay users; to generate explanations, descriptions and summaries from DBPedia or from knowledge bases; to guide the user in formulating knowledge base queries; and to provide ways for cultural heritage institutions such as museums and libraries to present information about their holdings in multiple textual forms.

In this paper, we focus on an important subtask of generation from RDF data namely lexicalisation of RDF properties. Given a property, our goal is to map this property to a set of possible lexicalisations. For instance, given the property HASWONPRIZE, our goal is to automatically infer lexicalisations such as *was honored with* and *received*.

Our approach is based on learning low-dimensional vector embeddings of words and of KB triples so that representations of triples and their corresponding lexicalisations end up being similar in the embedding space. Using these embeddings, we can then assess the similarity between a property and a set of candidate lexicalisations by simply applying the dot product to their vector embeddings.

One difficulty when lexicalising RDF properties is that, while in some cases, there is a direct and simple relation between the name of a property and its verbalisation (e.g., BIRTHDATE / "was born on"), in other cases, the relation is either indirect (e.g., ROUTEEND / "finishes at") or opaque (e.g., CREW1UP / "is the commander of").

To account for these two possibilities, we therefore explore two main ways of creating candidate lexicalisations based on either *lexical-* or on *extensional-relatedness*. Given some input property p, lexically-related candidate lexicalisations for p are phrases containing synonyms or derivationally related words of the tokens making up the name of the input property. In contrast, extensionally-related candidate lexicalisations are phrases containing named entities which are in its extension. For instance, given the property CREW1UP, if the pair of entities (STS-

*Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*SEM 2016)*, pages 219–228,
Berlin, Germany, August 11-12, 2016.

130, GEORGE_D._ZAMK) is in its extension (i.e., there exists an RDF triple of the form ⟨ STS-130, CREW1UP, GEORGE_D._ZAMK ⟩), all sentences mentioning STS-130, GEORGE_D._ZAMK or both will be retrieved and exploited to build the set of candidate lexicalisations for CREW1UP. Figure 1 shows some example L- and E-candidate lexicalisations phrases.

In summary, the key contribution made in this paper is a novel method for lexicalising RDF properties which differs from previous work in two ways. First, while lexical and extensional relatedness have been used before for lexicalising RDF properties (Walter et al., 2013), ours is the first lexicalisation approach which jointly considers both sources of information. Second, while previous approaches have used discrete representations and similarity metrics based on Wordnet, our method exploits continuous representations of both words and KB symbols that are learned and optimised for the lexicalisation task.

2 Related Work

We situate our work with respect to previous work on ontology lexicons but also to research on relation extraction (extracting verbalisations of knowledge base relations) and to embeddings-based approaches.

Ontology Lexicons (Trevisan, 2010) proposes a simple lexicalisation approach which exploits the tokens included in a property name to build candidate lexicalisations. In brief, this approach consists in tokenizing and part-of-speech tagging relation names with a customized tokenizer and part-of-speech (PoS) tagger. A set of hand-defined mappings is then used to map PoS sequences to lexicalisations. For instance, given the property name HASADRESS, this approach will produce the candidate lexicalisation "the address of S is O" where S and O are place-holders for the lexicalisations of the subject and object entity in the input RDF triple.

(Walter et al., 2013; Walter et al., 2014a; Walter et al., 2014b) describes an approach for inducing a lexicon mapping DBPedia properties to possible lexicalisations. The approach combines a label-based and a pattern-based method. The label-based method extracts lexicalisations from property names using additional information (e.g., synonyms) from external resources. The pattern-based method extract lexicalisations from a text corpus by retrieving sentences containing entities that are related by a DBPedia property and generalising over the dependency paths that connect them using hand-written patterns and frequency counts.

While these approaches can be effective, (Trevisan, 2010)'s approach fails to account for "opaque" property names (i.e., property such as CREW1UP whose lexicalisation is not directly deducible from the tokens making up that property name) and the pattern-based approach of (Walter et al., 2013), because it relies on frequency counts rather than lexical relatedness, allows for lexicalisations that may be semantically unrelated to the input property. In contrast, we learn continuous representations of both KB properties and words and exploit these to rank candidate lexicalisations which are either lexically- or extensionally-related to the properties to be lexicalised. In this way, we consider both types of property names while systematically checking for semantic relatedness.

Relation Extraction Earlier Information Extraction (IE) systems learned an extractor for each target relation from labelled training examples (Riloff, 1996; Soderland, 1999). For instance, (Riloff, 1996) first extract relation mention patterns from the corpus then rank these based on the number of time a relation pattern occurs in a text labelled with the target relation.

More recent work on Open IE has focused on building large scale knowledge bases such as Reverb by extracting arbitrary relations from text (Wu and Weld, 2010; Fader et al., 2011; Mohamed et al., 2011; Nakashole et al., 2012).

While relation extraction can be viewed as the mirror task of relation lexicalisation, there are important differences. Our lexicalisation task differs from domain specific IE in that it is unsupervised (we do not have access to annotated data). It also differs from open IE in that the set of properties to be lexicalised is predefined whereas, by definition, in open IE, the set of relations to be extracted is unrestricted. That is, while we aim to find the possible lexicalisations of a given set of relations (here DBPedia properties), open IE seeks to extract an unrestricted set of relations from text. Nevertheless, (Nakashole et al., 2012) includes a clustering phase which permits grouping relation clusters with a predefined set of properties such as, in particular, DBPedia properties. In Section 6, we therefore compare our results with the lexical-

Property	CROSSES
L-Candidate lexicalisation	"Old Blenheim Bridge spans Schoharie Creek"
Property	CREW1UP
RDF Triple	⟨ STS-130, CREW1UP, GEORGE_D._ZAMKA ⟩
E-Candidate lexicalisation	"Zamka served as the commander of mission STS-130"

Figure 1: Example L- and E-candidate lexicalisation phrases.

isations output by (Nakashole et al., 2012)'s approach.

Embedding-based Approaches The model we propose is inspired by (Bordes et al., 2014). In (Bordes et al., 2014), low dimensional embedding of words and KB symbols are learned so that representations of questions and their corresponding answers end up being similar in the embedding space. The embeddings are learned using automatically generated questions from KB triples and a dataset of questions marked as paraphrases (WikiAnswers, (Fader et al., 2011)). We adapt this model to the lexicalisation task by generating noisy lexicalisations of KB triples using a simple generation approach and by exploiting different paraphrase resources (c.f. Section3). Our approach further differs from (Bordes et al., 2014) in that we combine this embedding based framework with a pre-selection of candidate lexicalisations which reflects knowledge about the property extension and the property name. As mentioned in Section 1, E-related candidate lexicalisation phrases are sentences mentioning subject and/or object of the property being considered for lexicalisation while L-related candidate lexicalisation phrases are phrases containing synonyms or derivationally related words of the token making up the name of that property. In this way, we provide a joint modelling of the impact of lexical and extensional similarity on lexicalisation.

3 Approach

Given a KB property p, our task is to find a set of possible lexicalisations L_p for p. For instance, given the property HASWONPRIZE, our goal is to automatically infer lexicalisations such as *was honoured with* and *received*.

3.1 Lexicalisation Algorithm

Our lexicalisation algorithm is composed of the following steps:

Embeddings Using distant supervision, we learn embeddings of words and KB symbols such that the representations of KB triples, of sentences artificially generated from these triples and of their paraphrases are similar in the embedding space.

Candidate Lexicalisations Using WordNet and the extension of RDF properties (i.e., the set of pairs of entities related by that property), we build sets of candidate lexicalisation phrases. "Subject Relation Object" phrases are extracted from the set of candidate sentences using Reverb (Etzioni et al., 2011). Reverb is a tool for Open IE which extracts relation mentions from text based on frequency counts and regular expression filters.

Ranking Using the dot product on embedding based representations of triples and candidate lexicalisation phrases, we rank candidate lexicalisations of properties.

Extractions We apply some normalisation rules on the relation mention of the ranked lexicalisations to eliminate "duplicates". These rules consist in a small set of basic patterns to detect and remove adverbs, adjectives, determiners, etc. For instance, given the following relation mentions *always led by*, *is also led by* and *is currently led by* only one version will be extracted that is *led by*. From the top ranked lexicalisation phrases according to some threshold (e.g. top 10), we extract the lexicalisation set L_p for property p. Lexicalisations in L_p are relation mentions from the ranked lexicalisation phrases.

3.2 Learning Words and KB symbols Embeddings

Similar to the work of (Bordes et al., 2014), we use distant supervision and multitask training to learn embeddings of words and KB symbols.

Training Set Generation We train on two datasets, one aligning KB triples with automatically generated verbalisations of these triples and

the other, aligning paraphrases. The first dataset ($\mathcal{T}$) is used to learn a similarity function between KB symbols and words, the second ($\mathcal{P}$) to account for the many ways in which a given property may be verbalised.

Triples and Sentences ($\mathcal{T}$) We build a training corpus of KB triples and Natural Language sentences by combining the pattern based lexicalisation approach of (Trevisan, 2010) (c.f. Section 2) with a simple grammar based generation step. We apply this approach to map KB property names to syntactic constructions and then use a simple grammar to generate sentences from KB triples. For instance, the triple in (1a) will yield the sentences in (1b-g):

(1) a. ⟨ DUMBARTON_BRIDGE, LOCATEDINAREA, MENLO_PARK_CALIFORNIA ⟩
b. "The Dumbarton Bridge should be located in menlo park california."
c. "It should be located in menlo park california."
d. "Dumbarton Bridge located in menlo park california."
e. "Dumbarton Bridge which should be located in menlo park california."
f. "Menlo Park California in which dumbarton bridge is located."
g. "The Dumbarton Bridge should be located in menlo park california."

On average, each property is associated with 5.9 sentences. Given a training pair (t, s) such that $t = (s_k, p_k, o_k)$, we generate negative examples by corrupting the triple i.e., by producing pairs of the form (t', s) such that $t' = (s_k, p'_k, o_k)$ and $(s_k, o_k) \notin p'_k$.

Paraphrases ($\mathcal{P}$). To learn embeddings and a similarity function that takes into account the various ways in which a property can be lexicalised, we supplement our training data with pairs of paraphrases contained in the PPDB paraphrase database, in the WikiAnswers dataset and in DBPedia (DBPP). Positive examples (p_i, p_j) are taken from these datasets and negative examples are produced by creating corrupted pairs (p_i, p_l) such that p_i is not in the paraphrase dataset of p_l and vice versa.

The PPDB database was extracted from bilingual parallel corpora following (Bannard and Callison-Burch, 2005)'s bilingual pivoting method[1]. PPDB comes pre-packaged in 6 sizes: S to XXXL. The smaller packages contain only better-scoring, high-precision paraphrases, while the larger ones aim for high coverage. Additionally PPDB is broken down into lexical paraphrases (i.e. one word to one word), phrasal paraphrases (i.e. multi-word phrases), as well as syntactic paraphrases which contain non-terminals. We use PPDB version 2.0 M size lexical and phrasal sets which contain overall 3525057 paraphrase pairs. We choose to use medium size sets to incorporate some variability while still favouring higher quality paraphrases. As for the type of paraphrases, we took only the lexical and phrasal ones given that our goal is geared to acquiring alternative lexicalisations in terms of wording rather than syntactic variation.

Wikianswers is a corpus of 18M question-paraphrase pairs collected by (Fader et al., 2013), with 2.4M distinct questions in the corpus. Because these pairs have been labelled collaboratively, the data is highly noisy ((Fader et al., 2013) estimated that only 55% of the pairs were actual paraphrases).

Finally, the BDPP dataset consists of (entity, class) pairs extracted from the DBPedia ontology. They provide a bridge between the entity names appearing in the DBPedia triples and the more generic common nouns which may be used in text.

Using the resources and tools just described, we create a triple/sentence corpus $\mathcal{T}$ consisting of 317853 triple/sentence pairs obtained from 53384 KB triples of 149 relations. The paraphrase corpus $\mathcal{P}$ contains 3525057 (PPDB), 220998 (WikiAnswers) and 54489 (DBPP) paraphrase pairs. Figure 2 shows some positive and negative training examples drawn from the $\mathcal{T}$ and $\mathcal{P}$ datasets.

Training Using a training corpus created as described in the previous section, we learn a similarity function S between triples and candidate lexicalisations which is defined as:

$$S_{t/s}(t, s) = f(t)^\top . g(s) \quad (1)$$

with

$$f(t) = K^\top . \phi(t) \quad (2)$$

and

$$g(s) = W^\top . \psi(s) \quad (3)$$

[1]Briefly, the intuition underlying the bilingual pivoting method is that expressions sharing the same translation into a target language are paraphrases.

$\mathcal{T}$	
(t, s)	($\langle$ ARISTOTLE, INFLUENCED, CHRISTIAN_PHILOSOPHY $\rangle$, "Christian philosophy who is influenced by Aristotle.")
(t', s)	($\langle$ ARISTOTLE, COMPUTINGMEDIA, CHRISTIAN_PHILOSOPHY $\rangle$, "Christian philosophy who is influenced by Aristotle.")

$\mathcal{P}$ (PPDB)	
(p_i, p_j)	("collaborate", "cooperate")
(p_i, p_l)	("collaborate", "improving")
(p_i, p_j)	("is important to emphasize that", "is notable that")
(p_i, p_l)	("is important to emphasize that", "are using")
$\mathcal{P}$ (Wikianswers)	
(p_i, p_j)	("much coca cola be buy per year", "much do a consumer pay for coca cola")
(p_i, p_l)	("much coca cola be buy per year", "information on neem plant")
$\mathcal{P}$ (DBPP)	
(p_i, p_j)	("Amsterdam", "Place")
(p_i, p_l)	("Amsterdam", "Novels first published in serial form")

Figure 2: Examples of positive examples present in the $\mathcal{T}$ and $\mathcal{P}$ training datasets with their corresponding corrupted negative counterpart.

$K \in \mathbb{R}^{n_k \times d}$ and $W \in \mathbb{R}^{n_w \times d}$ are the embedding matrices for KB symbols and for words respectively with n_k, the number of distinct symbols in the knowledge base and n_w, the number of distinct word forms in the text corpus. Furthermore, $\phi(t)$ and $\psi(s)$ are binary vectors indicating whether a KB symbol/word is present or absent in t/s. Thus, $f(t)$ and $g(s)$ are the embeddings of t and s and $S_{t/s}$ scores their similarity by taking their dot product.

To learn word embeddings which capture the similarity between a triple and a set of paraphrases (rather than just the similarity between a triple and artificially synthesised sentences), we multitask the training of our model with the task of paraphrase detection. That is, the weights of the W matrix for words are learnt with the training of the triple/sentence similarity function $S_{t/s}$ and the training of a similarity function S_p for paraphrases which uses the same embedding matrix W for words and is trained on $\mathcal{P}$, the paraphrase corpus. The phrase similarity function S_p between two natural language phrases p_i and p_j is defined as follows:

$$S_p(p_i, p_j) = f(p_i)^\top . f(p_j) \quad (4)$$

Similarly to (Bordes et al., 2014), we train our model using a margin-based ranking loss function so that scores of positive examples should be larger than those of negative examples by a margin of 1. That is, for $S_{t/s}$, we minimize:

$$\forall i, j, \forall [1 - S_{s/t}(t_i, s_i) + S_{s/t}(t_j, s_i)] \quad (5)$$

where (t_i, s_i) is a positive triple/sentence example and (t_j, s_i) a negative one. Similarly, when training on paraphrase data, the ranking loss function to minimise is:

$$\forall i, j, l, \forall [1 - S_p(p_i, p_j) + S_p(p_i, p_l)] \quad (6)$$

where (p_i, p_j) is a positive example from the paraphrase corpus $\mathcal{P}$ and (p_i, p_l) a negative one.

4 Implementation

The model is implemented in Python using the Keras(Chollet, 2015) library with Theano backend.

We initialise the W matrix with pre-trained vectors which already provide a rich representation for words. We use the publicly available GloVe (Pennington et al., 2014) vectors[2] of length 100. These vectors were trained on 6 billions words from Wikipedia and the English Gigaword. We set the dimension d of the K and W matrices to 100. For K we use uniform initialisation.

The size of the vocabulary for the W matrix, the n_w dimension, is 130970 words. This is considering all words appearing in the $\mathcal{T}$ and $\mathcal{P}$ sets. The size of the K matrix, the n_k dimension, is 43797 counting both KB entities and relations.

The training for both similarity functions $S_{t/s}$ and S_p is performed with Stochastic Gradient Descent. The learning rate is set to 0.1 and the number of epochs to 5. Training run approximately 15 hours[3].

[2] http://nlp.stanford.edu/projects/glove/

[3] A first phase run on a machine with 1 CPU Intel Xeon

5 Experiments

DBPedia[4] is a crowd-sourced knowledge based extracted from Wikipedia and available on the Web in RDF format. Available as Linked Dataon the web, the DBPedia knowledge base defines Linked Data URIs for millions of concepts. It has become a de facto central hub of the web of data and is heavily used by systems that employ structured data for applications such as web-based information retrieval or search engines.

Like many other large knowledge bases (e.g., Freebase or Yago) available on the web, DBPedia lacks lexical information stating how DBPedia properties should be lexicalised. We apply our lexicalisation model to DBPedia object properties. We construct candidate lexicalisation sets in the following way.

Candidate Lexicalisations As mentioned in Section 1, we consider two main ways of building sets of candidate lexicalisations for a given property p.

E-LEX$_p$: Let WKP$_p$ be the set of sentences extracted from Wikipedia which contain at least one mention of two entities that are related in DBPedia by the property p. WKP$_p$ was built using the pre-processing tools[5] of the MATOLL framework (Walter et al., 2013; Walter et al., 2014b). Then E-LEX$_p$ is the corpus of candidate lexicalisations extracted from WKP$_p$ using Reverb.

L-LEX$_p$: Given WKP the corpus of Wikipedia sentences, L-LEX$_p$ is the corpus of relation mentions extracted from WKP using Reverb and filtered to contain only mentions which include words that are lexically related to the tokens making up the property name. Lexically related words include all synonyms and all derivationally related words listed in Wordnet for a given token.

6 Evaluation and Results

We compare the output of our lexicalisation method with the following resources and approaches.

X3440, 4 cores/CPU, 16GB RAM and a second phase on a machine with 2 CPUs Intel Xeon L5420, 4 cores/CPU, 16GB RAM.

[4] http://wiki.dbpedia.org/

[5] https://github.com/ag-sc/matoll

TEAM, COUNTRY, ORDER, DEATHPLACE, OCCUPATION, KINGDOM, NATIONALITY, BATTLE, HOMETOWN, AWARD, PREDECESSOR, PUBLISHER, DISTRIBUTOR, OWNER, RECORDEDIN, ALBUM, PRODUCT, PARENT, AFFILIATION, EDUCATION, ROUTEEND, ORIGIN, NEARESTCITY, ARCHITECT, COMPOSER, MOUNTAINRANGE, FOUNDEDBY, INFLUENCED, GARRISON, LEADER, PROGRAMMINGLANGUAGE

Table 1: Set of DBPedia object properties used in the evaluation.

DBlexipedia$_e$: a lexicon[6] automatically inferred from Wikipedia using the method described in (Walter et al., 2013; Walter et al., 2014a; Walter et al., 2014b) (c.f. section 2). Lexical entries are inferred using either the extension of the properties (by retrieving sentences containing entities that are related by a DBPedia property and generalising over the dependency paths that connect them.) or synonyms of the words contained in the property name.

PATTY: a lexicon automatically inferred from web data using relation extraction and clustering (c.f. (Nakashole et al., 2012)).

QUELO: a lexicon automatically derived using the method described in (Trevisan, 2010) (c.f. section 2). Lexical entries are derived by first, tokenizing and pos tagging property names and second, mapping the resulting pos-tagged sequences to pre-defined mention patterns.

For the quantitative evaluation, we use the lexicon developed manually for DBPedia properties by (McCrae et al., 2011) as a gold standard[7]. We test on a held-out set of 30 properties[8] chosen from DBPedia and which were present in the gold standard lexicon, in the other systems we compare with and in the available E-Lex$_p$ corpus. Table 1 lists the set of properties.

We compute precision (Correct/Found), recall

[6] For this evaluation we use the version available for download at http://dblexipedia.org/download and we use only the English lexical entries.

[7] This lexicon is available at https://github.com/ag-sc/lemon.dbpedia

[8] The selection of these properties was based, on one hand, on the frequency with a third of the selected properties appearing more than 80000 times in DBPedia, a third appearing less than 20000 times and a third appearing between 20000 and 80000 times (min. is 5936 for PROGRAMMINGLANGUAGE and max. is 1825970 for TEAM). On the other hand, we include properties with different name/label patterns imposing differences in verbalisation difficulty, e.g. compound nouns as ROUTEEND or PROGRAMMINGLANGUAGE.

(Correct/GOLD) and F1 measure of each of the above resources. Recall is the proportion of (property, lexicalisation) pairs present in GOLD which are present in the resource being evaluated, precision the proportion in a resource which is also present in GOLD and F1 is the harmonic mean of precision and recall[9].

In our setup though, precision (and therefore F1) values are artificially decreased because the reference lexicon is small (2.4 lexicalisations in average per property) and often fails to include all possible lexicalisations. The number of correct lexicalisations can therefore be under-estimated while the number of found lexicalisations is usually larger than the number of gold lexicalisations and therefore much larger than the number of correct (= GOLD ∩ Found) lexicalisations.

We report results using different sets of lexicalisation candidates (L-LEX, E-LEX, their union and their intersection) and different thresholds or methods for selecting the final set of lexicalisations. These include: retrieving the n-best lexicalisations (k=10) *versus* using an adaptive threshold which varies depending on the size of the set of candidate lexicalisations and on the distributions of its ranking scores. We tried taking all lexicalisations over the median (median), over the mid-range ((min+max)/2) or in the third quartile (Q3). We also tested an alternative ranking technique where the score of each lexicalisation is the product of its similarity score (dot product of the embedding vectors representing the property and the lexicalisation) with the frequency of this particular lexicalisation in the set of candidate lexicalisations[10]. We rerank the lexicalisations using these new scores and consider only the lexicalisations in the third quartile of the distribution (FreqQ3). Further if this results in having either less than 7 or more than 25 lexicalisations, we ignore the Q3 constraint and take the 7 and 25 best respectively (FreqQ3Limit(7,25)).

Table 3 summarises the results.

[9]To determine whether a given property lexicalisation is correct, i.e. present in the GOLD, we use "soft" comparison rather than strict string matching. This consists in checking whether the stemmed gold lexicalisation is contained in a given candidate lexicalisation. For instance, the candidate "main occupation of" and gold "occupation of" are considered as a match.

[10]In the set of candidate lexicalisations, the same lexicalisation may occur with minor variations. We compute the frequency of a given lexicalisation by removing adjectives and adverbs and counting the number of repeated occurrences after removing these.

Recall In terms of recall, our results generally outperform QUELO, PATTY and DBlexipedia$_e$.

The low recall score of QUELO shows that simply using patterns based on the property name does not suffice to find appropriate property lexicalisations. This is true in particular of properties such as ROUTEEND where the correct lexicalisation is difficult to guess from the property name.

DBlexipedia$_e$ at k=10 scores lower (0.29) than the corresponding version of our approach union(k=10), R:0.38). Interestingly, for our approach, better recall values are consistently obtained using L-LEX suggesting that many of the verbalisations found in GOLD can be extracted from text that is unrelated to the extension of DBPedia properties. This is a nice feature as this permits avoiding the data sparsity issue which arises when a DBPedia property has either a restricted extension or a small set WKP$_p$ of candidate lexicalisations. Indeed, we found that out of a set of 149 DBPedia properties, the MATOLL corpus did not provide any sentences for 19 of them. In such cases, an approach based only on extensionally related sentences of the property would have zero recall. This is in line with the results of (Walter et al., 2013; Walter et al., 2014a) who observe that such an approach yields a recall of 0.35 whilst combining it with a lexically based approach (using synonyms of the tokens occurring in the property name) permits increasing recall to 0.5.

Finally, although PATTY has a comparatively high recall value (0.59), its precision is very low (0.0015) and versions of our approach with comparable precision (e.g., E-LEX(All)) have a much higher recall (R: 0.80).

Precision As shown in Table 3, the retrieval approach which gives the best results in terms of both precision and F1 is in fact to take the 10-best. Together with the much lower precision achieved by the random baselines (Random*k=10), this result suggests that the similarity function learned by our model appropriately captures the similarity between DBPedia properties and their lexicalisations.

Unsurprisingly, QUELO has the highest precision as it only guesses lexicalisation based on the tokens making up the property name. For instance, for noun property names like OWNER it produces the following two lexicalisations: "owner" and "owner of"; for verb based property names like RECORDEDIN it produces the lexicalisation

System/goldLemonDBPPatterns	Avg.NB	Recall	Precision	F1
L-LEX(k=10)	9.9	0.3611	0.0875	**0.1409**
L-LEX(median)	343	**0.7500**	0.0052	0.0104
L-LEX((min+max)/2)	216	0.6250	0.0069	0.0137
L-LEX(Q3)	104	0.5000	0.0115	0.0225
L-LEX(FreqQ3)	104	0.5139	0.0118	0.0231
L-LEX(FreqQ3Limit(7,25))	218	0.4583	**0.0505**	0.0909
L-LEX(All)	687.4	**0.8194**	0.0029	0.0057
E-LEX(k=10)	10	0.3333	0.0800	**0.1290**
E-LEX(median)	778.2	**0.7222**	0.0022	0.0044
E-LEX((min+max)/2)	301.8	0.6806	0.0054	0.0107
E-LEX(Q3)	251	0.6250	0.0059	0.0118
E-LEX(FreqQ3)	251	0.6250	0.0059	0.0118
E-LEX(FreqQ3Limit(7,25))	23.3	0.5000	0.0514	0.0933
E-LEX(All)	1557	**0.8056**	0.0012	0.0025
union(k=10)	10	0.3889	0.0933	0.1505
union(median)	543	**0.8194**	0.0036	0.0072
union((min+max)/2)	47.7	0.6389	0.0320	0.0610
union(Q3)	86.7	0.5972	0.0165	0.0320
union(FreqQ3)	85.8	0.6667	0.0185	0.0361
union(FreqQ3Limit(7,25))	10.8	0.4861	**0.1080**	**0.1768**
union(All)	2162.5	0.9444	0.0010	0.0021
intersec(k=10)	0.4	0.0556	0.3636	0.0964
intersec(median)	35.27	**0.4444**	0.0305	0.0571
intersec((min+max)/2)	14.8	0.3333	0.0547	0.0939
intersec(Q3)	8.6	0.2639	0.0748	0.1166
intersec(FreqQ3)	12.3	0.2917	0.0575	0.0961
intersec(FreqQ3Limit(7,25))	2.2	0.2500	**0.2813**	**0.2647**
intersec(All)	81.9	0.5417	0.0159	0.0309
L-LEXRandom(k=10)	9.9	0.2083	0.0505	0.0813
E-LEXRandom(k=10)	10	0.0833	0.0200	0.0323
QUELO	2.13	0.2917	0.3281	0.3088
DBlexipedia$_e$(k=10)	5.4	0.2500	0.1104	0.1532
PATTY	936	0.5694	0.0015	0.0029

Figure 3: Micro-averaged Precision, Recall and F1 with respect to GOLD. The column Avg.NB indicates the averaged number of candidate lexicalisations for each system.

PROGRAMMINGLANGUAGE	*written in*, **uses**, include, **based on**, supports, is a part of, **programming language for** (4/1)
AFFILIATION	member of, associated with, *affiliated with*, **affiliated to**, **affiliate of**, **accredited by**, tied to, founded in, president of, associate member of (4/1)
COUNTRY	village in, **part of**, one of, *located in*, commune in, town in, born in, refer to, county in, country in, city in (2/1)
MOUNTAINRANGE	**mountain in**, **located in**, **include**, range from, **mountain of**, **mountain range in**, *part of*, **lies in**, reach, peak in, **find in**, highest mountain in (8/1)
DISTRIBUTOR	sell, appear in, allocate to, air on, **release**, **make**, star in, appear on (2/2)
LEADER	lead to, **leader of**, **led by**, **is a leader in**, visit, become, **lead**, lead producer of, *president of*, **elected leader of**, left (6/3)

Figure 4: Example Lexicalisations output by our System (Union.FreqQ3Limit7-25). Gold items are in italics. Items in bold indicates a correct lexicalisation absent from the gold. The number N/G in bracket indicates the number N of lexicalisations produced by our system that are not in the gold standard and the number G of items in the gold standard.

"recorded in". On these two properties, QUELO perfectly coincides with the entries defined in GOLD. This explains the high F1 obtained by QUELO. However, as argued in the previous section, QUELO's approach fails to account for cases where the relation name is indirect or opaque. Moreover, it does not support the generation of alternative lexicalisations. For the property EDUCATION, the gold standard defines the the lexical entries "attend", "go to" and "study at" which QUELO fails to produce.

DBlexipedia$_e$ has a precision score (0.11) comparable to the corresponding version of our approach (union(k=10), P:0.09) and PATTY has a very low precision (P:0.0015). A manual examination of the data shows that the relation extraction approach fails to find a sufficiently large number of distinct property lexicalisations. The lexicalisations found often contain many near repetitions (e.g., "has graduated from, graduated from, graduates") but few distinct paraphrases (e.g., "graduate from, study at").

To better assess, the precision of our system we therefore manually examined the results of our system and annotated all outputs lexicalisations which were correct but not in the gold. Based on this updated gold, precision for union.FreqQ3Limit7-25 is in fact, 0.289.

Example Output Table 4 shows some example output of our system (for union.FreqQ3Limit7-25)[11]. These examples show that our system correctly predicts additional lexicalisations that are absent from GOLD.

They also show that our approach can produce both L- and E-related lexicalisations. Thus for instance, for the property PROGRAMMINGLANGUAGE, our model produces the lexicalisation "programming language for" which is clearly an L-lexicalisation that can be directly derived from the property name. However, it also derives more context-sensitive E-lexicalisations such as "written in", "uses" and "based on" which are not lexically related to the property name but can be found by considering E-related candidate lexicalisations i.e., sentences such as "FastTacker Digit was written in Pascal" which contain entities that are arguments of the PROGRAMMINGLANGUAGE property. Similarly, the COUNTRY property whose gold lexicalisation is "located in" (the RDF triple ⟨ Sakhalin_Oblast, country, Russia ⟩ can be verbalised as " Sakhalin Oblast is located in Russia"), is correctly assigned the lexicalisations "located in" and "part of". Interestingly, our approach also yield more specific lexicalisations such as "is a village/commune/town/county in" which may also be correct lexicalisations given the appropriate subject. For instance, "is a town in" is a correct lexicalisation of the COUNTRY property given the triple ⟨ Paris, country, France ⟩.

[11]The complete set of extractions is available at http://www.loria.fr/~perezlla/content/sw_resources/union.FreqQ3Limit.txt .

7 Conclusion

We use an embeddings based framework for identifying plausible lexicalisations of KB properties. While embeddings have been much used in domains such as question answering, semantic parsing and relation extraction, they have not been used so far for the lexicalisation task. Conversely, existing approaches to lexicalisation which exploits the similarity between property name and candidate lexicalisations do so on the basis of discrete representations such as WordNet Synsets. In contrast, we learn embeddings of words and KB symbols using distant supervision. We show that, when applied to DBPedia object properties, our approach yields competitive results with these discrete approaches.

As future work, we plan to conduct a larger scale evaluation. This will include the application of the approach to datatype properties and test on a larger set of properties.

The scoring function used by our approach is based on a bag-of-words representation of natural language phrases. We have observed that tuples and candidate lexicalisation phrases like ⟨ AMERICAN_FILM_INSTITUTE, LOCATION, CALIFORNIA ⟩ and "A new city was built on a nearby location" are scored high as they share some highly related words. We plan to explore whether a more complex representation of natural language phrases could remedy this shortcoming.

Acknowledgements

We thank the French National Research Agency for funding the research presented in this paper in the context of the WebNLG project. We would also like to thank Sebastian Walter for kindly providing us with the MATOLL corpus.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. *CoRR*, abs/1404.4326.

Fran¸cois Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618. Citeseer.

John McCrae, Dennis Spohr, and Philipp Cimiano. 2011. Linking lexical resources and ontologies on the semantic web with lemon. In *The semantic web: research and applications*, pages 245–259. Springer.

Thahir P Mohamed, Estevam R Hruschka Jr, and Tom M Mitchell. 2011. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1447–1455. Association for Computational Linguistics.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Discovering and exploring relations on the web. *Proceedings of the VLDB Endowment*, 5(12):1982–1985.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. 2004. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. *Engineering Knowledge in the Age of the Semantic Web*, pages 63–81.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

Stephen Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine learning*, 34(1-3):233–272.

Marco Trevisan. 2010. A portable menuguided natural language interface to knowledge bases for querytool. Master's thesis, Free University of Bozen-Bolzano (Italy) and University of Groningen (Netherlands).

Sebastian Walter, Christina Unger, and Philipp Cimiano. 2013. A corpus-based approach for the induction of ontology lexica. In *Natural Language Processing and Information Systems*, pages 102–113. Springer.

Sebastian Walter, Christina Unger, and Philipp Cimiano. 2014a. Atolla framework for the automatic induction of ontology lexica. *Data & Knowledge Engineering*, 94:148–162.

Sebastian Walter, Christina Unger, and Philipp Cimiano. 2014b. M-atoll: a framework for the lexicalization of ontologies in multiple languages. In *The Semantic Web–ISWC 2014*, pages 472–486. Springer.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.