A Lexicalized Tree Kernel for Open Information Extraction

Ying Xu[†], **Christoph Ringlstetter**[‡], **Mi-Young Kim**[†], **Randy Goebel**[†], **Grzegorz Kondrak**[†], **Yusuke Miyao**[§]
[†]Department of Computing Science, University of Alberta
[‡]Gini, Muenchen
[§]National Institute of Informatics / JST, PRESTO
[†]{yx2,miyoung2,rgoebel,gkondrak}@ualberta.ca
[‡]c.ringlstetter@gini.net
[§]yusuke@nii.ac.jp

Abstract

In contrast with traditional relation extraction, which only considers a fixed set of relations, Open Information Extraction (Open IE) aims at extracting all types of relations from text. Because of data sparseness, Open IE systems typically ignore lexical information, and instead employ parse trees and Part-of-Speech (POS) tags. However, the same syntactic structure may correspond to different relations. In this paper, we propose to use a lexicalized tree kernel based on the word embeddings created by a neural network model. We show that the lexicalized tree kernel model surpasses the unlexicalized model. Experiments on three datasets indicate that our Open IE system performs better on the task of relation extraction than the state-of-the-art Open IE systems of Xu et al. (2013) and Mesquita et al. (2013).

1 Introduction

Relation Extraction (RE) is the task of recognizing relationships between entities mentioned in text. In contrast with traditional relation extraction, for which a target set of relations is fixed a priori, Open Information Extraction (Open IE) is a generalization of RE that attempts to extract all relations (Banko et al., 2007). Although Open IE models that extract N-ary relations have been proposed, here we concentrate on binary relations.

Most Open IE systems employ syntactic information such as parse trees and part of speech (POS) tags, but ignore lexical information. However, previous work suggests that Open IE would benefit from lexical information because the same syntactic structure may correspond to different relations. For instance, the relation <Annacone, coach of, Federer> is correct for the sentence "Federer hired Annacone as a coach", but not for the sentence "Federer considered Annacone as a coach," even though they have the same dependency path structure (Mausam et al., 2012). Lexical information is required to distinguish the two cases.

Here we propose a lexicalized tree kernel model that combines both syntactic and lexical information. In order to avoid lexical sparsity issues, we investigate two smoothing methods that use word vector representations: Brown clustering (Brown et al., 1992) and word embeddings created by a neural network model (Collobert and Weston, 2008). To our knowledge, we are the first to apply word embeddings and to use lexicalized tree kernel models for Open IE.

Experiments on three datasets demonstrate that our Open IE system achieves absolute improvements in F-measure of up to 16% over the current state-of-the-art systems of Xu et al. (2013) and Mesquita et al. (2013). In addition, we examine alternative approaches for including lexical information, and find that excluding named entities from the lexical information results in an improved F-score.

2 System Architecture

The goal of the Open IE task is to extract from text a set of triples $\{< E_1, R, E_2 >\}$, where E_1 and E_2 are two named entities, and R is a textual fragment that indicates the semantic relation between the two entities. We concentrate on binary, single-word relations between named entities. The candidate relation words are extracted from dependency structures, and then filtered by a supervised tree kernel model.

Our system consists of three modules: entity extraction, relation candidate extraction, and tree

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 279–284, Beijing, China, July 26-31, 2015.

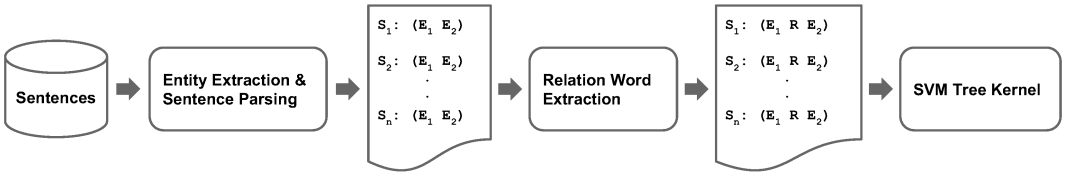


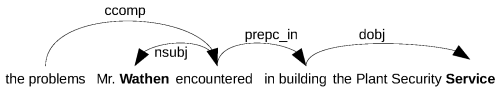
Figure 1: Our Open IE system structure.

kernel filtering. The system structure is outlined in Figure 1. We identify named entities, parse sentences, and convert constituency trees into dependency structures using the Stanford tools (Manning et al., 2014). Entities within a fixed token distance (set to 20 according to development results) are extracted as pairs $\{< E_1, E_2 >\}$. We then identify relation candidates R for each entity pair in a sentence, using dependency paths. Finally, the candidate triples $\{< E_1, R, E_2 >\}$ are paired with their corresponding tree structures, and provided as input to the SVM tree kernel. Our Open IE system outputs the triples that are classified as positive. In the following sections, we describe the components of the system in more detail.

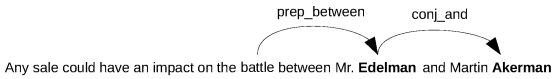
3 Relation Candidates

Relation candidates are words that may represent a relation between two entities. We consider only lemmatized nouns, verbs and adjectives that are within two dependency links from either of the entities. Following Wu and Weld (2010) and Mausam et al. (2012), we use dependency patterns rather than POS patterns, which allows us to identify relation candidates which are farther away from entities in terms of token distance.

We extract the first two content words along the dependency path between E_1 and E_2. In the following example, the path is $E_1 \rightarrow$ encounter $\rightarrow$ build $\rightarrow E_2$, and the two relation word candidates between "Mr. Wathen" and "Plant Security Service" are *encounter* and *build*, of which the latter is the correct one.



If there are no content words on the dependency path between the two entities, we instead consider words that are directly linked to either of them. In the following example, the only relation candidate is the word *battle*, which is directly linked to "Edelman."



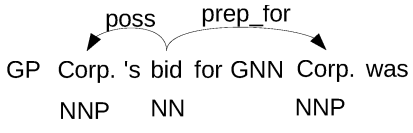
The relation candidates are manually annotated as correct/incorrect in the training data for the tree kernel models described in the following section.

4 Lexicalized Tree Kernel

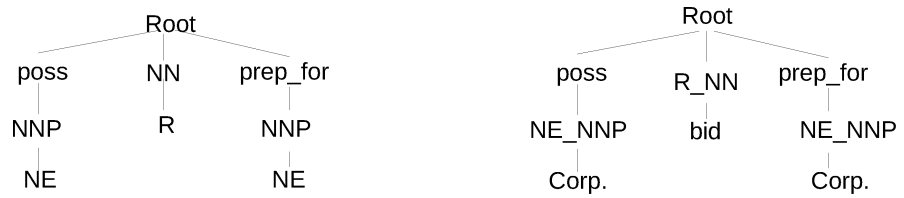
We use a supervised lexicalized tree kernel to filter negative relation candidates from the results of the candidate extraction module. For semantic tasks, the design of input structures to tree kernels is as important as the design of the tree kernels themselves. In this section, we introduce our tree structure, describe the prior basic tree kernel, and finally present our lexicalized tree kernel function.

4.1 Tree Structure

In order to formulate the input for tree kernel models, we need to convert the dependency path to a tree-like structure with unlabelled edges. The target dependency path is the shortest path that includes the triple and other content words along the path. Consider the following example, which is a simplified representation of the sentence *"Georgia-Pacific Corp.'s unsolicited $3.9 billion bid for Great Northern Nekoosa Corp. was hailed by Wall Street."* The candidate triple identified by the relation candidate extraction module is *<Georgia-Pacific Corp., bid, Great Northern Nekoosa Corp.>*.



Our unlexicalized tree representation model is similar to the unlexicalized representations of Xu et al. (2013), except that instead of using the POS tag of the path's head word as the root, we create an abstract *Root* node. We preserve the dependency labels, POS tags, and entity information as tree nodes: (a) the top dependency labels are in-



(a) An un-lexicalized dependency tree.

(b) A lexicalized dependency tree.

Figure 2: An unlexicalized tree and the corresponding lexicalized tree.

cluded as children of the abstract *Root* node, other labels are attached to the corresponding parent labels; (b) the POS tag of the head word of the dependence path is a child of the *Root*; (c) other POS tags are attached as children of the dependency labels; and (d) the relation tag 'R' and the entity tags 'NE' are the terminal nodes attached to their respective POS tags. Figure 2(a) shows the unlexicalized dependency tree for our example sentence.

Our lexicalized tree representation is derived from the unlexicalized representation by attaching words as terminal nodes. In order to reduce the number of nodes, we collapse the relation and entity tags with their corresponding POS tags. Figure 2(b) shows the resulting tree for the example sentence.

4.2 Tree Kernels

Tree kernel models extract features from parse trees by comparing pairs of tree structures. The essential distinction between different tree kernel functions is the Δ function that calculates similarity of subtrees. Our modified kernel is based on the SubSet Tree (SST) Kernel proposed by Collins and Duffy (2002). What follows is a simplified description of the kernel; a more detailed description can be found in the original paper.

The general function for a tree kernel model over trees T_1 and T_2 is:

$$K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \Delta(n_1, n_2), \quad (1)$$

where n_1 and n_2 are tree nodes. The Δ function of SST kernel is defined recursively:

1. $\Delta(n_1, n_2) = 0$ if the productions (context-free rules) of n_1 and n_2 are different.
2. Otherwise, $\Delta(n_1, n_2) = 1$ if n_1 and n_2 are matching pre-terminals (POS tags).
3. Otherwise, $\Delta(n_1, n_2) = \prod_j (1 + \Delta(c(n_1, j), c(n_2, j))$,

where $c(n, j)$ is the jth child of n.

4.3 Lexicalized Tree Kernel

Since simply adding words to lexicalize a tree kernel leads to sparsity problems, a type of smoothing must be applied. Bloehdorn and Moschitti (2007) measure the similarity of words using WordNet. Croce et al. (2011) employ word vectors created by Singular Value Decomposition (Golub and Kahan., 1965) from a word co-occurrence matrix. Plank and Moschitti (2013) use word vectors created by Brown clustering algorithm (Brown et al., 1992), which is another smoothed word representation that represents words as binary vectors. Srivastava et al. (2013) use word embeddings of Collobert and Weston (2008), but their tree kernel does not incorporate POS tags or dependency labels.

We propose using word embeddings created by a neural network model (Collobert and Weston, 2008), in which words are represented by n-dimensional real valued vectors. Each dimension represents a latent feature of the word that reflects its semantic and syntactic properties. Next, we describe how we embed these vectors into tree kernels.

Our lexicalized tree kernel model is the same as SST, except in the following case: if n_1 and n_2 are matching pre-terminals (POS tags), then

$$\Delta(n_1, n_2) = 1 + G(c(n_1), c(n_2)), \quad (2)$$

where $c(n)$ denotes the word w that is the unique child of n, and $G(w_1, w_2) = exp(-\gamma \|w_1 - w_2\|^2)$ is a Gaussian function for two word vectors, which is a valid kernel.

We examine the contribution of different types of words by comparing three methods of including lexical information: (1) relation words only; (2) all words (relation words, named entities, and other words along the dependency path fragment); and (3) all words, except named entities. The words that are excluded are assumed to be different; for example, in the third method, $G(E_1, E_2)$ is always zero, even if the entities, E_1 and E_2, are the same.

5 Experiments

Here we evaluate alternative tree kernel configurations, and compare our Open IE system to previous work.

We perform experiments on three datasets (Table 1): the Penn Treebank set (Xu et al., 2013), the New York Times set (Mesquita et al., 2013), and the ClueWeb set which we created for this project from a large collection of web pages.[1] The models are trained on the Penn Treebank training set and tested on the three test sets, of which the Penn Treebank set is in-domain, and the other two sets are out-of-domain. For word embedding and Brown clustering representations, we use the data provided by Turian et al. (2010). The SVM parameters, as well as the Brown cluster size and code length, are tuned on the development set.

Set	train	dev	test
Penn Treebank	750	100	100
New York Times	—	300	500
ClueWeb	—	450	250

Table 1: Data sets and their size (number of sentences).

Table 2 shows the effect of different smoothing and lexicalization techniques on the tree kernels. In order to focus on tree kernel functions, we use the relation candidate extraction (Section 3) and tree structure (Section 4.1) proposed in this paper. The results in the first two rows indicate that adding unsmoothed lexical information to the method of Xu et al. (2013) is not helpful, which we attribute to data sparsity. On the other hand, smoothed word representations do improve F-measure. Surprisingly, a neural network approach of creating word embeddings actually achieves a lower recall than the method of Plank and Moschitti (2013) that uses Brown clustering; the difference in F-measure is not statistically significant according to compute-intensive randomization test (Padó, 2006).

With regards to lexicalization, the inclusion of relation words is important. However, unlike Plank and Moschitti (2013), we found that it is better to exclude the lexical information of entities themselves, which confirms the findings of Riedel et al. (2013). We hypothesize that the correctness of a relation triple in Open IE is not closely related to entities. Consider the example mentioned in (Riedel et al., 2013): for relations like "X visits Y", X could be a person or organization, and Y could be a location, organization, or person.

[1]The Treebank set of (Xu et al., 2013), with minor corrections, and the ClueWeb set are appended to this publication.

Smoothing	Lexical info	P	R	F_1
none (Xu13)	none	85.7	72.7	78.7
none	all words	89.8	66.7	76.5
Brown (PM13)	relation only	88.7	71.2	79.0
Brown (PM13)	all words	84.5	74.2	79.0
Brown (PM13)	excl. entities	86.2	75.8	80.7
embedding	relation only	93.9	69.7	80.0
embedding	all words	93.8	68.2	79.0
embedding	excl. entities	95.9	71.2	**81.7**

Table 2: The results of relation extraction with alternative smoothing and lexicalization techniques on the Penn Treebank set (with our relation candidate extraction and tree structure).

Our final set of experiments evaluates the best-performing version of our system (the last row in Table 2) against two state-of-the-art Open IE systems: Mesquita et al. (2013), which is based on several hand-crafted dependency patterns; and Xu et al. (2013), which uses POS-based relation candidate extraction and an unlexicalized tree kernel. Tree kernel systems are all trained on the Penn Treebank training set, and tuned on the corresponding development sets.

The results in Table 3 show that our system consistently outperforms the other two systems, with absolute gains in F-score between 4 and 16%. We include the reported results of (Xu et al., 2013) on the Penn Treebank set, and of (Mesquita et al., 2013) on the New York Times set. The ClueWeb results were obtained by running the respective systems on the test set, except that we used our relation candidate extraction method for the tree kernel of (Xu et al., 2013). We conclude that the substantial improvement on the Penn Treebank set can be partly attributed to a superior tree kernel, and not only to a better relation candidate extraction method. We also note that word embeddings statistically outperform Brown clustering on the ClueWeb set, but not on the other two sets.

The ClueWeb set is quite challenging because it contains web pages which can be quite noisy. As a result we've found that a number of Open IE errors are caused by parsing. Conjunction structures are especially difficult for both parsing and relation extraction. For example, our system extracts the relation triple <Scotland, base, Scott> from the sentence "Set in 17th century Scotland

	P	R	F_1
	Penn Treebank set		
Xu et al. (2013)*	66.1	50.7	57.4
Brown (PM13)	82.8	65.8	73.3
Ours (embedding)	91.8	61.6	**73.8**
	New York Times set		
Mesquita et al. (2013)*	72.8	39.3	51.1
Brown (PM13)	83.5	44.0	**57.6**
Ours (embedding)	85.9	40.7	55.2
	ClueWeb set		
Xu et al. (2013)	54.3	35.8	43.2
Mesquita et al. (2013)	63.3	29.2	40.0
Brown (PM13)	54.1	31.1	39.5
Ours (embedding)	45.8	51.9	**48.7**

Table 3: Comparison of complete Open IE systems. The asterisks denote results reported in previous work.

and based on a novel by Sir Walter Scott, its high drama..." with the wrong dependency path *Scotland* $\overset{conj\_and}{\rightarrow}$ *based* $\overset{prep\_by}{\rightarrow}$ *Scott*. In the future, we will investigate whether adding information from context words that are not on the dependency path between two entities may alleviate this problem.

6 Conclusion

We have proposed a lexicalized tree kernel model for Open IE, which incorporates word embeddings learned from a neural network model. Our system combines a dependency-based relation candidate extraction method with a lexicalized tree kernel, and achieves state-of-the-art results on three datasets. Our experiments on different configurations of the smoothing and lexicalization techniques show that excluding named entity information is a better strategy for Open IE.

In the future, we plan to mitigate the performance drop on the ClueWeb set by adding information about context words around relation words. We will also investigate other ways of collapsing different types of tags in the lexicalized tree representation.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions. This work was funded in part by Alberta Innovates Center for Machine Learning (AICML), Natural Sciences and Engineering Research Council of Canada (NSERC), Alberta Innovates Technology Futures (AITF), and National Institute of Informatics (NII) International Internship Program.

References

Michele Banko, Michael J Cafarella, Stephen Soderl, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *International Joint Conference on Artificial Intelligence*, pages 2670–2676.

Stephan Bloehdorn and Alessandro Moschitti. 2007. Structure and semantics for expressive text kernels. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 861–864, New York, NY, USA. ACM.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1034–1046, Stroudsburg, PA, USA. Association for Computational Linguistics.

G. Golub and W. Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, page 205224.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open

relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457. Association for Computational Linguistics.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '13)*, June.

Shashank Srivastava, Dirk Hovy, and Eduard Hovy. 2013. A walk-based semantically enriched tree kernel over distributed word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1411–1416, Seattle, Washington, USA, October. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*, ACL '10, pages 118–127, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 868–877, Atlanta, Georgia, June. Association for Computational Linguistics.