2G2-2

Semantic Relation Extraction Using Penalty Tree Similarity

Jie Yang[*1] Yutaka Matsuo[*2] Mitsuru Ishizuka[*1]

[*1]Graduate School of Information Science and Technology, The University of Tokyo
[*2]School of Engineering, The University of Tokyo

In the past decades, kernel methods are enthusiastically explored for relation extraction. This paper proposes a penalty tree similarity algorithm by extending the dependency tree kernel. Dependency tree kernel computes the similarity of two parse trees by enumerating their matched sub-trees. The penalty tree similarity, however, not only consider the similar structures of the parse trees, but also count in their influences by exploring relative position information between the sub-trees to the target (the entity pairs that generate the tree). An experiment is conducted and the comparison between the dependency tree kernel and the penalty tree similarity is also done. The results show that the former achieves a better performance.

1. Introduction

Information Extraction (IE) is an important task for natural language processing (NLP). It is further divided several subtasks: entity recognition, relation recognition, event recognition and so on ([**?**]). This paper focuses on the relation recognition task that extracts named relationship between entities in text. Take the sentence "Officials in California are warning residents." as an example, *Officials* is an instance of the entity type *government*; *California* is an instance of the entity type *location*; the two entities are connected by the relation *located-in*.

Many studies have been done over the past decades, which can be divided into three classes: the supervised approach, the unsupervised approach and the semi-supervised approach. The supervised approach [MFRW00, ZAR03] achieves the best performance, but is known to suffer from poor adaptivity. It is noticed that it is time consuming to construct an adequate size of training data for a new application or domain. The unsupervised approach [HSG] does not need the annotated corpus. However, they usually fail to give the relation name connecting two entity mentions; the performance is also low. Semi-supervised approach [YGTH00, BM07] draws much attention recently, yet the effectiveness of the method depends on the initial seeds and it still lacks systematic methods for seed selection. Because the extraction performance is our first concern, this work focuses on the supervised approach.

Kernel methods [CST00] show their effectiveness on the relation extraction task and have gained popularity in recent years. Typically, the supervised approach models the relation extraction task into a classification problem on tree structures of sentences. Kernels are used in the diagram to measure the tree similarity. [Hau99] introduces the convolution kernel over discrete structures like strings and trees. [ZAR03] proposes a kernel between two parse trees, which recursively matches nodes from roots to leaves in a top-down manner. [CS04] extends the parse tree kernel and makes two kernels over the dependency trees. Various extensions over the dependency tree kernel are proposed since then. Composite kernels[ZG] explore different levels of syntactic processing and achieve better performance than single kernel. [BM] observed that the shortest path between entity mentions usually captures the relation between entities and proposed a shortest path kernel. A context-sensitive convolution tree kernel is proposed by [ZZJZ07] that enumerates both context-free and context-sensitive information. Kernel based similarity methods for relation extraction are also utilized by [ZSW+05]. These studies treat tree nodes equally, we will show in this paper that improved performance can be achieved by treating nodes differently according to their positions.

Contact: Jie Yang
Graduate School of Information Science and Technology, The University of Tokyo
Room 111C1, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, JAPAN
TEL: 03-5841-6774, FAX: 03-5841-8570
Email: yangj@mi.ci.i.u-toko.ac.jp

2. Penalty Tree Similarity

In this section we first introduce the dependency tree kernel briefly; then present our penalty tree similarity algorithm which adjust the kernel result by weighting tree nodes differently.

2.1 Dependency Tree Kernel

Formally, a kernel function K is a mapping $K : X \times X \longrightarrow [0, \infty]$ from instance space X to a similarity score. It is a symmetric, positive semi-definite function and can be expressed as a dot product in a high-dimensional space: $K(x, y) = \Sigma_i \phi_i(x)\phi_i(y) = \phi(x) \cdot \phi(y)$. Here $\phi_i(x)$ is some feature function over the instance x.

We describe the tree kernel following the notations of [CS04]. A parse tree T is a set of node $t_0, t_1, ...t_n$. A node t_i includes a set of features given by $\phi(t_i) = v_1, ..., v_d$. We denote by $t_i[j]$ the jth child of node t_i, and by $t_i[\mathbf{c}]$ the set of all children of node t_i. We reference a subset **j** of children of t_i by $t_i[\mathbf{j}] \subseteq t_i[\mathbf{c}]$. The parent of node t_i is referred to as $t_i.p$. Then the tree similarity function $K(T_1, T_2)$ over two trees T_1 and T_2 with the root nodes r_1 and r_2 ($r_1, r_2 \in T$) is defined recursively as follows:

$$K(T_1, T_2) = \begin{cases} 0, & \text{if } m(r_1, r_2) = 0 \\ s(r_1 + r_2) + & \\ \quad K_c(r_1[\mathbf{c}], r_2[\mathbf{c}]), & \text{otherwise} \end{cases} \quad (1)$$

where $m(t_i, t_j) \in \{0, 1\}$ is a matching function and $s(t_i, t_j) \in$

$[0, \infty)$ is a similarity function. Denote two possibly overlapping subsets of $\phi(t_i)$ as $\phi_m(t_i) \subseteq \phi(t_i)$ and $\phi_s(t_i) \subseteq \phi(t_i)$, we define:

$$m(t_i, t_j) = \begin{cases} 1, & \text{if } \phi_m(t_i) = \phi_m(t_j) \\ 0, & \text{otherwise} \end{cases}$$

and

$$s(t_i, t_j) = \sum_{v_q \in \phi_s(t_i)} \sum_{v_r \in \phi_s(t_j)} C(v_q, v_r)$$

where $C(v_q, v_r)$ equals to one if the feature of the two words equals, otherwise zero.

The function K_c is a kernel function over children. In our work we use contiguous kernel rather than sparse kernel to save processing time. Denote $\sum_{s=1,...l(\mathbf{i})} K(P_1[i_s], P_2[j_s])$ by $K(P_1(\mathbf{i}), P_2[\mathbf{j}])$ then,

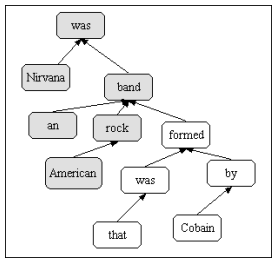
$$K_c(r_1[\mathbf{c}], r_2[\mathbf{c}]) = \sum_{\mathbf{a},\mathbf{b},l(\mathbf{a})=l(\mathbf{b})} \lambda^{l(\mathbf{a})} K(t_i[\mathbf{a}], t_j[\mathbf{b}])$$

2.2 Penalty Tree Similarity Algorithm

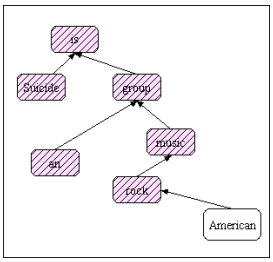
Tree kernel presented above measures the structure similarity of two dependency trees by enumerating and comparing all the possible sub-structures of the two trees. However there are cases where considering the tree structure only is not enough. Figure 1 shows an example. Figure 1(a) and Figure 1(c) are subtrees of the sentence "Nirvana was an American rock band that was formed by Kurt Cobain and Krist Novoselic in Aberdeen, Wahsington". Figure 1(a) is the minimal tree T_1 of nodes *Nirvana* and *Cobain* and Figure 1(c) the minimal tree T_1' of *Nirvana* and *American*. Figure 1(b) and 1(d) show the same minimal tree T_2 of nodes *Suicide* and *American* of the sentence "Suicide is an American rock music group since 1971". When computing their similarities with the tree kernel, $K(T_1, T_2)$ and $K(T_1', T_2)$ return the same result. It is because the matched structures of the two pairs of trees are same, as shown by the colored nodes in Figure 1. However, in fact T_1' and T_2 should be assigned a greater similarity value than T_1 and T_2. This problem occurs because the tree kernel assumes that tree nodes contribute equally to the similarity measurement, but we suggest that their contributions should be calculated differently according to the node pair that generates the minimal tree. In essential, its the similarity between node pairs, not that between tree structures that we are seeking for. For example, in T_1' the matched nodes mostly convey the information about the band, such as its genre (*rock*) or the origination place (*American*), but are less about its membership information (*Cobain*).

With above observation, we propose an algorithm that calculates the contribution of different nodes differently.

We first find out the shortest path that connects the target node pair, which can be thought as the trunk of the minimal tree. For example, in tree T_1 of Figure 1(a), the trunk connecting nodes *Nirvana* and *Cobain* is "Nirvana was band formed by Cobain". Other nodes belong to the branch of the tree which, according to the feature of dependency tree, grammatically modify the trunk. For example, the words *an American rock* modifies the word *band* and *that was* modifies *formed*. Our hypothesis is that the farther a node is to the trunk, the less important it is to the node pair, the nearer the more important. We consider the nodes that did't get matched after the tree-kernel function and calculate the penalty they bring to the tree-kernel similarity. The more unmatched nodes

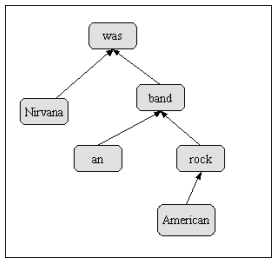


(a) T_1

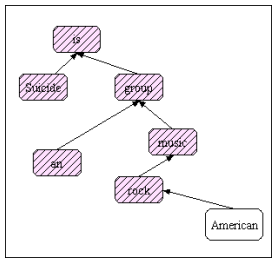


(b) T_2

T_1 is the sub minimal tree generated by "Nirvana" and "Cobain"; T_2 is the minimal tree generated by "Suicide" and "American".



(c) T_1'



(d) T_2

T_1' is the minimal tree generated by "Nirvana" and "'American"; T_2 is the minimal tree generated by "Suicide" and "American".

Figure 1: $K(T_1, T_2)$ and $K(T_1', T_2)$ return the same value because their matched nodes (matched nodes are filed with colors) are the same.

are in a tree, the nearer those nodes are to the trunk, the more penalty is given.

Above statements are formulated as below. Given two nodes t_i and t_j in a dependency tree T, the trunk is defined as the shortest path $p(t_i, t_j)$ (written as $\mathbf{p}$ for short when necessary) connecting them.

$$p(t_i, t_j) = t_i, t_{i_1}, t_{i_2} ... t_{i_n}, t_p, t_{j_m}, ... t_{j_2}, t_{j_1}, t_j$$

where $t_i.p = t_{i_1}$, $t_{i_1}.p = t_{i_2}$, ..., $t_{i_n}.p = t_p$ ... and $t_j.p = t_{j_1}$, $t_{j_1}.p = t_{j_2}$, ..., $t_{j_m}.p = t_p$. For any node $t \in T$, its distance to the trunk $p(t_i, t_j)$ is defined to be:

$$d(t, \mathbf{p}) = min_{t_k \in \mathbf{p}} \{|p(t, t_k)|\}$$

For example, in Figure 1(a), $p(Nirvana, Cobain)$ is "Nirvana was band formed by Cobain" and the distance from the node *American* to the trunk is three.

Denote the set of unmatched nodes as $\mathbf{u}$, the penalty function is:

$$P(\mathbf{u}) = \frac{1}{(1 + \sum_{x \in \mathbf{u}} \frac{1}{\sqrt{x}})^2} \in (0, 1] \quad (2)$$

and the penalty similarity algorithm of two dependency trees is defined as:

$$S(T_1, T_2) = P(u_1) * P(u_2) * K(T_1, T_2) \quad (3)$$

where $P(u_1)$ and $P(u_2)$ calculate the penalty function of T_1 and T_2 respectively.

3. Relation Extraction with Penalty Tree Similarity

The relation extraction approach proposed in this paper consists of 1) pre-processing, 2) word pair detection and 3) relation labeling and raking using KNN.

3.1 Pre-processing and Word Pair Detection

First a sentence is parsed into a dependency tree using the Connexor*1 parser. Then the entity tagger Annie [CMBT02] is used to detect entity information. After pre-processing, a dependency tree is constructed for the input sentence. Nodes of the tree correspond to words of the sentence and edges correspond to dependency relations. Each node is described by eight features as shown in Table 1.

	Feature	Example
f_1	lemma word	move (stem of moving)
f_2	dependency function (37 values)	subj, agt
f_3	syntactic function (28 values)	ADVL, APP
f_4	surface syntactic tag (12 values)	N, A, VP
f_5, f_6, f_7	morphological tag	ABBR, NUM, PRON
f_8	Annie tag (eight values)	PERSON, DATE

Table 1: Features of each node in the dependency tree.

Then we can evaluate the similarity between two words using the eight features. Knowledge encoded in WordNet is also used. The similarity between word w_1 and word w_2 is calculated as below:

$$Sim_w(w_1, w_2) = \frac{\sum_{i=1...8} \varphi(f_{1,i}, f_{2,i}) + Sim_{wn}(w_1, w_2)}{9}$$

where $\varphi(f_{1,i}, f_{2,i})$ equals to *one* if the ith feature of the two words equals, otherwise *zero*. Function Sim_{wn} is the WordNet based word similarity method developed by [SVH04]. Based on the function Sim_{wn}, for a input sentence, we can detect we detect word pairs that are more likely to be embodied in a relation instance. The aim of word pair detection is to reduce noise and to save processing time by eliminating useless word pairs.

The word pair detection step is described as follows. Given an input sentence s_t and a sentence corpus C; each sentence $s \in C$ is connected with a set of relation instances R_s. Our task is to detect the valuable word pairs of s_t. Suppose words w_1 and w_2 belong to s_t. Each relation $r_{s,i} \in R_s$ takes the form of (w_1', rel', w_2'). The word pair is estimated by function $Est(w_1, w_2)$:

$$\sum_{s \in R} \sum_{r_{s,i} \in R_s} \frac{\sum_{w \in \{w_1, w_1'\}, w' \in \{w_2, w_2'\}} Sim_w(w, w')}{sum_{s \in C}|R_s|} \quad (4)$$

Every word pairs in sentence s is tested against the Est function. Word pairs with the estimating value above the threshold θ_{wp} is reserved to the next step for relation labeling.

3.2 Relation Labeling and Ranking with KNN

For each candidate word pair we get from the previous step, we find its K most nearest neighbors (relation instances) in the training corpus using Equation (3). The neighbors vote for the relation name of the word pair then construct a candidate relation triple. The average similarity over the K neighbors are assigned to the triple as rank. A threshold θ_{rel} is used to filter out triples ranked lower. Note that the word pair detection step considers word features independently whereas the relation labeling step emphasizes more on the structure feature of a sentence in addition to word features.

*1 http://www.connexor.eu/

4. Experiments

4.1 Experiment Setup

We invited twelve college volunteers from different departments to create a lightweight annotated corpus. They are asked to annotate sentences of Wikipedia web pages about the music band. The annotation takes the form of a triple (e_1, rel, e_2). Triple elements e_1 and e_2 are words (also named as entity mentions) in the sentence and rel is the relation name connecting e_1 and e_2. We call it a lightweight annotation because annotators are not obliged to any predefined knowledge (e.g. predefined entity types or relation classes). As a result, a corpus containing 90 Wikipedia articles is created with 363 sentences being annotated by 774 relation triples. After an analysis on the relation triples, we found that they form 20 major classes as is shown in Table 2. More than half (415 our of 774) of the triple relations are covered by the top 20 relation classes.

Classes from Wikipedia nformation table			
genre (70)	members (57)	origin (48)	member role (48)
years active (44)	a music group (39)	member occupations (26)	album (26)
lyrics (13)	born in	alias	instruments
former members	die in		
associate act	web site	single	birth name
label	awards		

Table 2: 20 convergent relation classes.

The entire corpus is fed to our extraction process as a training corpus, although the evaluation is conducted over on the top 20 relation class only.

We created a test corpus containing 10 abstracts randomly selected from Wikipedia music band pages. The corpus does not overlap with the training corpus. The experiment is carried out along the process describe in Section 3.. The threshold for word pair detection θ_{wp} and candidate triple θ_{rel} are set to be 0.2. K is set to 3 based on our observation of the training data.

Because of the specularity of our training corpus, we evaluate the performance of our method as follows. One of the volunteers is invited to classify the extracted relations as right or wrong. Since we did not use name entity recognition at present, we do not require the whole entity mention be recognized. For example, "(Dream, isa, band)" is thought to be correct, although the complete relation should be the "(Dream Theater, isa, band)".

4.2 Results

From the test corpus, there are 158 relations extracted, among which 68 are considered to be correct. Under this criterion, we achieve a precision of 43.0%. To show the novelty of the penalty tree similarity function, we made another experiment with the same settings, only changing the penalty function (Equation (3)) with the typical dependency tree kernel (Equation (1)). It reports a precision of 15.1%. The penalty tree similarity achieves a significant improvement over the dependency tree kernel.

Table 3 summarized the correctly extracted relations by classifying them into corresponding relation classes. We can see that most of the extracted relations (shown in the upper part of Table 3) fall into the relation classes listed in Table 2. We then use the

Relation Class	relation number	relation name
genre	27	style,genre,isa
a music group	15	wasan,isa,is
years active	6	formed,startfrom,time,founded
origin	4	locatedin,origin,foundedin
members	6	credit,is
member role	2	as
is	2	is
is a kind of	4	isa,is
others	2	was

Table 3: 68 out of 158 (43.0%) relations are correct, the 68 relations refelct domain interests.

convergent relation classes as the baseline to test the coverage of our method. We identify in each test sentence the convergent relation classes (shown in Table 2) it is about. If a relation class is identified in a sentence, the "expected number" of the relation class increases by one. If our method extracts at least one relation of the labeled class, the "extracted number" increases by one. The result is shown in Table 4 and the recall over the relation classes is 56.6%. The tree kernel reports the coverage of 26.9%.

Relation Class	expected	extracted	Relation Class	expected	extracted
genre	13	9	a music group	12	12
origin	11	4	members	8	1
member role	3	0	years active	6	4

Table 4: Correctly extracted relations appear in 30 sentences, cover 56.6% over the 50 expectations.

5. Conclusion

This paper proposes an extension of the tree-kernel based relation extraction method. A penalty tree similarity function is presented which adjust the kernel methods by weighting nodes differently according their positions. Comparative experiments on a lightweight corpus show our method improves extraction performance.

References

[BM] Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP'2005*,.

[BM07] Razvan Bunescu and Raymond Mooney. Learning to extract relations from the web using minimal supervision. In *ACL'2007*, pages 576–583, Prague, Czech Republic, June 2007.

[CMBT02] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL'2002*, pages 168–175, Philadelphia, July 2002.

[CS04] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *ACL'2004*, pages 423–429, July 2004.

[CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[Hau99] D. Haussler. Convolution kernels on discrete structures. Technical report, 1999.

[HSG] T. Hasegawa, S. Sekine, and R. Grishman. Discovering relations among named entities from large corpora. In *ACL'2004*, pages 415–422.

[MFRW00] Scott Miller, Heidi Fox, Lance A. Ramshaw, and Ralph M. Weischedel. A novel use of statistical parsing to extract information from text. In *ANLP'2000*, pages 226–233, Seattle, Washington, USA, April 2000.

[SVH04] Nuno Seco, Tony Veale, and Jer Hayes. An intrinsic information content metric for semantic similarity in wordnet. In *ECAI'2004*, pages 1089–1090, Valencia, Spain, August 2004.

[Wan05] Pei Wang. Experience-grounded semantics: a theory for intelligent systems. *Cognitive Systems Research*, 6(4):282–302, 2005.

[YGTH00] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *COLING'2000*, pages 940–946, August 2000.

[ZAR03] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106, 2003.

[ZG] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *ACL'2005*, June.

[ZSW+05] Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *IJCNLP'2005*, pages 378–389, October 2005.

[ZZJZ07] Guodong Zhou, Min Zhang, Donghong Ji, and Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *EMNLP-CoNLL'2007*, pages 728–736, 2007.