Relation Extraction from Wikipedia Using Subtree Mining

Dat P.T. Nguyen
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan

Yutaka Matsuo
AIST
1-18-13 Sotokanda,
Tokyo 101-0021, Japan

Mitsuru Ishizuka
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan

Abstract

The exponential growth and reliability of Wikipedia have made it a promising data source for intelligent systems. The first challenge of Wikipedia is to make the encyclopedia machine-processable. In this study, we address the problem of extracting relations among entities from Wikipedia's English articles, which in turn can serve for intelligent systems to satisfy users' information needs. Our proposed method first anchors the appearance of entities in Wikipedia articles using some heuristic rules that are supported by their encyclopedic style. Therefore, it uses neither the Named Entity Recognizer (NER) nor the Coreference Resolution tool, which are sources of errors for relation extraction. It then classifies the relationships among entity pairs using SVM with features extracted from the web structure and subtrees mined from the syntactic structure of text. The innovations behind our work are the following: a) our method makes use of Wikipedia characteristics for entity allocation and entity classification, which are essential for relation extraction; b) our algorithm extracts a core tree, which accurately reflects a relationship between a given entity pair, and subsequently identifies key features with respect to the relationship from the core tree. We demonstrate the effectiveness of our approach through evaluation of manually annotated data from actual Wikipedia articles.

Introduction

The exponential growth and the reliability of Wikipedia (www.wikipedia.org) have recently attracted the attention of numerous users and researchers (Strube & Ponzetto 2006; Gabrilovich & Markovitch 2006). However, Wikipedia usage is currently limited to human readers (Völkel *et al.* 2006) because the Wikipedia data are written in natural language. This study is intended to deal with the problem of converting Wikipedia's English version (http://en.wikipedia.org) into a different structure using *relation extraction* technique, with the goal of locating interesting entities and identifying relations among them (Culotta & Sorensen 2004).

Relations used to structure Wikipedia in this task are defined in the form of a triple (e_p, *rel*, e_s) in which e_p and e_s are entities and where *rel* indicates the directed relationship between e_p and e_s. The current experiment limits entities and relations to a reasonable size in that an entity is classifiable as one of seven types: *person*, *organization*, *location*, *artifact*, *year*, *month* or *date*. A relation can be one of 13 types (e.g., Person-Founder-Company).

Unlike the web, Wikipedia records only one article for a real world entity, then its articles contain few duplicated pieces of text that provide cues for relations between an entity pair. In other words, Wikipedia contents are not so abundant, which requires that all the texts be analyzed even if they have complex structure. Furthermore, because Wikipedia articles are edited continuously by numerous collaborators[1], their content is believed to have high *grammatical correctness* compared to that of the web overall. Those assumptions enable us to use deep analysis techniques, which is usually infeasible for ordinary web pages. We propose a supervised learning method based on analyses of the syntactic structure of text, in which long dependencies between constituents (words or group of words which form a single unit in a sentence) in text are handled to improve recall. Put differently, analyzing the text at a syntactic level allows reduction of the variation of superficial text, which subsequently enables machines to recognize entity relations more accurately.

Some previous works (Bunescu & Mooney 2006; Cui *et al.* 2005) proposed matching techniques on dependency paths (sequences of dependency relations when tracing from one entity to the other in dependency structures) to estimate similarity of relations between entity pairs. Instead of analyzing dependency paths, our method analyzes dependency subtrees extended from the paths with the guidance of some keywords. Such subtrees are inferred to contain more evidence of the entities' inter-relation than the paths in some cases. Then, we propose a new feature obtained using a subtree-mining technique.

In addition to analysis of the Wikipedia text, we also make use of the characteristics of Wikipedia articles to narrow down the list of relations that might pertain between an entity pair. Specifically, the category hierarchy of Wikipedia is the main feature in our method to classify the entity type.

Our contributions can be summarized as follows:

- We propose various techniques targeting Wikipedia,

[1] http://en.wikipedia.org/wiki/Wikipedia:Modelling_Wikipedia's_growth

which are useful on Wikipedia for various other applications such as thesaurus construction and summarization.

- We show the feasibility of using Wikipedia for Semantic Web development in that the output of our system can be transformed straightforwardly into Semantic Web's metadata. Despite the limitation of the scope of our experiments, the results suggest the potential applicability of our approach.
- Our study suggests an example to bridge the gap between online social media (popular social media including blogs, wikis, message boards, and so on) and Semantic Web technology.

The remainder of this paper is organized as follows. The next section describes some related works, followed by an analysis of the characteristics of Wikipedia articles that are useful for this research. We then explain our proposed methods for relation extraction and entity classification in detail. Finally, we report our experimental results and conclude this paper with suggestions for some future work.

Related Works

Some earlier works on relation extraction using learning surface text have been introduced into the literature (Brin 1998; Agichtein & Gravano 2000; Ravichandran & Hovy 2002; Pasca *et al.* 2006). The authors conducted experiments on web data that are so abundant that they enable their systems to obtain easy patterns. The systems then learn such patterns based mostly on lexical information. Consequently, they cannot cope with distant dependencies between words in sentences. As a result, the methods might fail in this problem because the Wikipedia source data are more formal and complex, but not abundant.

Bunescu & Mooney (2006) present a kernel method to classify relationships of entity pairs by estimating similarity between dependency paths. Their method relies on an assumption that paths with different lengths tend to express different relationships. They estimate the similarity by hard matching, which depends on the lengths of the paths. Their method might be more efficient if the assumption and matching condition were relaxed. Our work is an attempt to overcome the problem by matching the decomposed subtrees, independently of their size.

To our knowledge, only one recent work has attempted relation extraction on Wikipedia: Culotta, McCallum, & Betz (2006) present a probabilistic model to integrate extraction and mining tasks performed on biographical text from Wikipedia. They formulate the relation extraction problem into a sequence labeling problem, which is then solved using Conditional Random Field to avoid errors of the traditional pipeline, including the Named Entity Recognizer (NER). Their supervised method uses both contextual and relational information to enable the two tasks to be mutually supportive and thereby improve the entire system. Our work resembles that effort, but the present work is motivated more by the Semantic Web vision: we intend to convert Wikipedia texts into machine-processable knowledge. Therefore, we use some special characteristics that are intrinsic to Wikipedia.

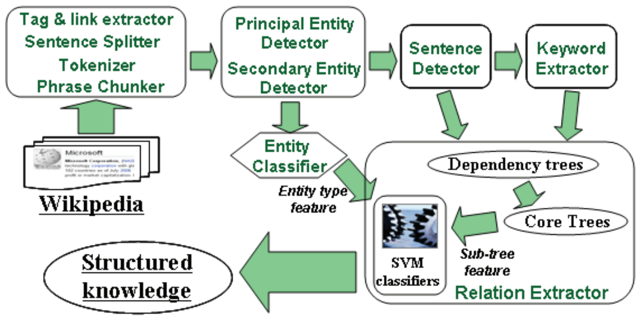

Figure 1: System framework

Wikipedia's Article Characteristics

Wikipedia has an *encyclopedic style*. Therefore, it mainly contains entries or articles, each of which provides information for a specific entity and further mentions other entities related to it. Culotta, McCallum, & Betz (2006) define the entities as *principal entity* and *secondary entity* respectively. In this research, we predict only relationships between the principal entity and each described secondary entity that contains a link to its descriptive article.

We use the following assumptions: A relationship can be expressed completely in one sentence. Furthermore, a relationship between an entity pair might be expressed with the implication of the principal entity in some sentences. Thus, for an article, only those sentences that contain at least one secondary entity are necessarily analyzed.

An interesting characteristic of Wikipedia is the existing category hierarchy that is used to group articles according to their content. Additionally, those articles for famous entities provide summary section on their right side, which are created by human editors. Such information describes the interactions of the principal entity to the others in table format, in which some target relations can be extracted directly. Finally, the first sentence of an article often defines the principal entity. We exploit such characteristics in this research.

Proposed Method

In this section, we first provide the overview of the method along with the framework of our systems. Then, we explain the details for each module in the framework.

Figure 1 depicts our framework for relation extraction. First, articles are processed to remove HTML tags and to extract hyperlinks that point to other Wikipedia articles. Text is then submitted to a pipeline including a *Sentence Splitter*, a *Tokenizer*, and a *Phrase Chunker* (an NLP module to split a sentence into difference phrases such as noun phrase, verb phrase and so on) supplied by the OpenNLP [2] tool set. The instances of the principal entity and secondary entities are then anchored in the articles. The *Secondary Entity Detector* simply labels the appropriate surface texts of the hyperlinks to other Wikipedia articles, which are proper nouns as secondary entities. The *Principal Entity Detector* will be explained in the following subsection.

[2]http://opennlp.sourceforge.net/

Table 1: Sample extracted referring expressions

Article	Referring expressions	Step
Bill Gates	[NP Bill/NNP Gates/NNP]	(ii)
	[NP William/NNP H./NNP Gates/NNP]	(ii)
	[NP Gates/NNP]	(iii)
	[NP The/DT Gates/NNP]	(iii)
	[NP he/PRP]	(iv)
	[NP him/PRP]	(iv)
Microsoft	[NP Microsoft/NNP]	(ii)
	[NP The/DT Microsoft/NNP Corporation/NNP]	(ii)
	[NP that/DT Microsoft/NNP]	(iii)
	[NP It/PRP]	(iv)
	[NP the/DT company/NN]	(v)
Microsoft Windows	[NP Microsoft/NNP Windows/NNP]	(ii)
	[NP Microsoft/NNP]	(iii)
	[NP Windows/NNP]	(iii)
	[NP the/DT Windows/NNP]	(iii)
	[NP it/PRP]	(iv)

After the entities are anchored, sentences that include at least one mention of secondary entities will be selected by a *Sentence Detector*. Each mention of the secondary entities should be analyzed to identify the relation between the underlying entity and the secondary entity. Secondary entities are always explicit, although the principal entity is sometimes implicit in sentences containing no mention.

Keywords that provide clues for each relation label are identified by a *Keyword Extractor*. An *Entity Classifier* module will classify the entities into types to limit the available relations for entity pairs. The *Relation Extractor* will extract *subtree feature* from a pair of the principal entity and a mention of secondary entity. It then incorporates the *subtree feature* together with *entity type feature* into a feature vector and classifies relations of the entity pairs using *SVM-based classifiers*.

Principal Entity Detector

This module detects all mentions of the principal entity in an article. The function of this module is actually classifiable as *Coreference Resolution* for noun phrases (Soon, Lim, & Ng 2001; Morton 2000), in which *referring expressions*, noun phrases that refer to the principal entity, are identified. All occurrences of identified referring expressions are labeled as mentions of the principal entity.

Ideally, we can use a Coreference Resolution tool to detect all the expressions referring to a topic of interest. However, our investigation on the coreference tool in the LingPipe library[3] and coreference package in OpenNLP tool set shows that directly applying the tools to Wikipedia's articles yields poor results. Supported by the aforementioned nature of Wikipedia, we propose a simple but efficient technique to identify a set of referring expressions, denoted as F, which provides better results than those produced by the above coreference tools. We adopt (Morton 2000) to classify the expressions in F into three types: (1) personal pronoun (2) proper noun (3) common nouns. Based on chunking information, the technique is as follows:

(i) Start with $F = \{\}$.

(ii) Select the first two chunks for F: the proper chunk (chunk with at least a proper noun) of the *article title* and the first proper chunk in the *first sentence* of the article, if any. These are the first two names of the principal entity. If F is still empty, stop.

(iii) For each remaining proper chunk p in the article, if p is derived from any expressions selected in (ii), then $F \leftarrow p$. Proper chunk p_1 is derived from proper chunk p_2 if all its proper nouns appear in p_2. These proper chunks are various identifiers of the principal entity.

(iv) In the article, select c as the most frequent *subjective pronouns*, find c' as its equivalent *objective pronoun* and add them to F.

(v) For each chunk p with the pattern [DT $N_1 \ldots N_k$] where DT is a *determiner* and N_i's are *common nouns*, if p appears more frequently than all the selected pronouns in (iv), then $F \leftarrow p$.

Table 1 shows some sample referring expressions extracted by the above technique. The third column indicates in which step the expressions are selected.

Entity Classifier

The entity type is very useful for relation extraction. For instance, the relation label between a *person* and an *organization* should be *founder*, *chairman*, and so on, but cannot be *spouse*, *product*, and so on. We first identify *year*, *month* and *date* entities by directly examining their surface text. Types of other entities, including principal entities and secondary entities, are identified by classifying their corresponding articles. We develop one SVM-based classifier for each remaining type using a *one-against-all* strategy (Hsu & Lin 2002). Using this strategy, the entities of one class serve as positive samples to train the corresponding classifier: all other entities serve as negatives. When testing, a novel entity is passed through all classifiers and receives a label from the corresponding classifier that gives the highest score. We represent an article in the form of a feature vector and use the following features: *category feature* (categories collected when tracing from the article up to k levels of its category structure), *pronoun feature* (the most frequent subjective pronoun in the article) and *singular noun feature* (singular nouns of the first sentence of the article).

Keyword Extractor

Our hypothesis in this research is that there exist some keywords that provide clues to the relationship between a pair. For example, to express the *founder* relation, a sentence should contain one keyword such as: *found*, *founder*, *founded*, *co-founders*, or *establish*, and so on. We identify such keywords using a semi-automatic method. First, we automatically extract some true relations from summary sections of Wikipedia articles. Then, we map entities in such relations to those in sentences to collect sample sentences for each relationship. The *Tf-idf* model is exploited to measure

[3]http://www.alias-i.com/lingpipe/index.html

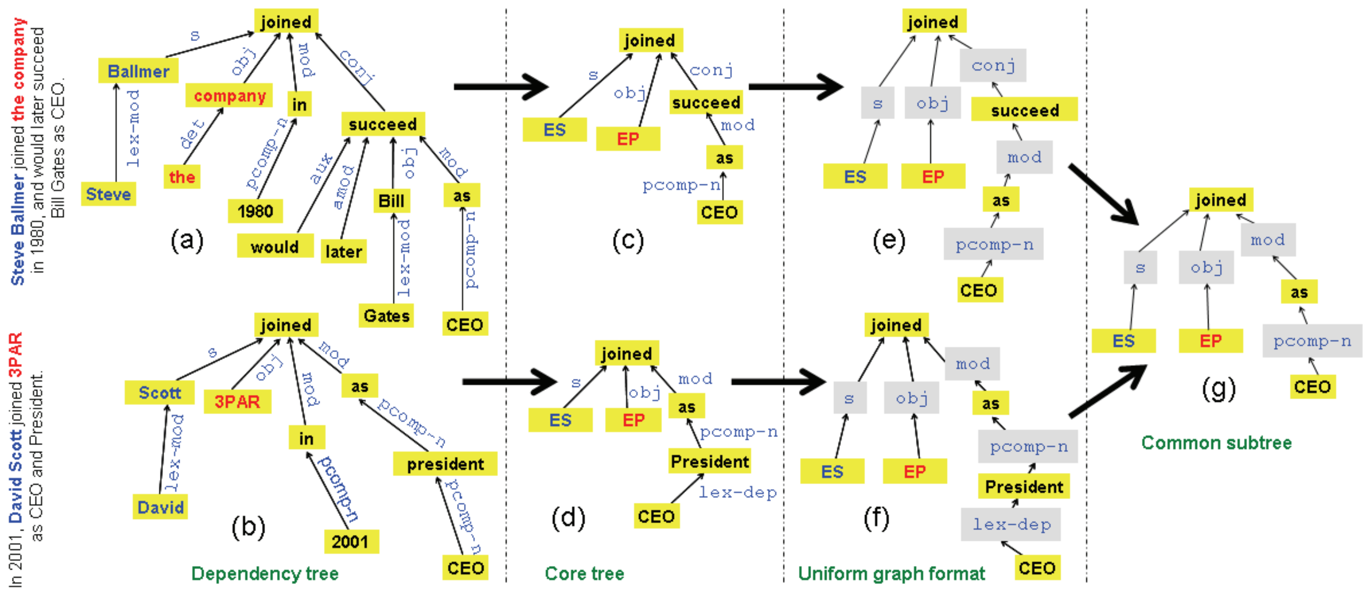

Figure 2: Dependency trees in (a) & (b); core trees with respect to *CEO* relationship in (c) & (d); new representation of the core trees in (e) & (f); common subtree in (g). The red letters *EP* denote the principal entity; the blue letters *ES* denote the secondary entity.

Table 2: List of relations and their keywords

Relation	Keywords
CEO	CEO, chief, executive, officer
Chairmans	chairman
COO	coo, chief, operating, officer
Director	director
Founder	found, founder, founded, establish, form, foundation, open
President	president
Vice chairman	vice, chairman
Birth date	born, bear, birth, birthday
Birth place	born, bear
Foundation	found, establish, form, founded, open, create, formed, established, foundation, founding, cofounder, founder
Location	headquartered, based, locate, headquarter, base, location, situate, located
Product	product, include, release, produce, service, operate, provide, market, manage, development, focus, manufacture, provider, launch, make, sell, introduce, producer, supplier, possess, retailer, design, involve, production, offering, serve, sale, supply
Spouse	marry, wife, married, husband, marriage

the relevance of words to each relationship for those on the dependency path between the entity pair. Finally, we choose the keywords manually from lists of candidates ranked by relevance score with respect to each relation. Table 2 shows the list of our target relations and our result selected from ranked lists of 35,820 total keyword candidates using only one hour of human labor.

Subtree Feature from the Dependency Path

In this section, we will describe how to obtain efficient features for extracting relations using subtree mining. One challenge for this problem is posed by the wide variation of surface text styles. However, because of syntactic analyses, we can explore relations between words in a sentence even if they are separated by many words. In other words, the syntactic analysis of sentences enables us to reduce the sentence variation. In this study, we extract relations between entities by analyzing a dependency graph of sentences. Bunescu & Mooney (2006) investigated the sentences in an Automated Content Extraction[4] (ACE) newspaper corpus and suggested that clues for the relationship between two entities in a sentence be placed on the shortest dependency path between the entities.

Some analyses suggest that Wikipedia sentences in which one entity of the pair is implied might counteract the hypothesis. For example, although the sentence shown in Fig. 2a shows the *CEO* relationship between *Steve Ballmer* and *the company*, the dependency path *"[the company]N* $\rightarrow^{obj}$ *[joined]V* $\leftarrow^{s}$ *[Steve Ballmer]N"* between the entities reveals no clue to the relationship.

To investigate whether a relationship r is held between the entities or not, our novel idea is to expand such a dependency path to a tree that contains as many clues for r as possible and then analyze the tree. We expand the path by integrating more paths between the secondary entity and the keywords of r, as described in the previous section. The expanded tree is defined as *core tree* of r because it attempts to capture the clues for r. Steps to extract the core tree C of a relationship r from a sentence s are described as follows.

(i) Initialize the core tree C as blank.

(ii) Derive the dependency tree D from s.

(iii) Label the group of nodes corresponding to words of secondary entity by an ES node in D.

[4]http://www.nist.gov/speech/tests/ace/

(iv) Apply (iii) to replace the principal entity with EP if the principal entity appears in s. Then obtain the shortest path from ES to EP in D denoted as P_0 and augment C with nodes and edges from P_0.

(v) For each keyword w of r, obtain the shortest path from ES to node of w, denoted as P_w and augment C with nodes and edges from P_w.

Figures 2c & 2d present exemplary core trees of the *CEO* relationship derived from the dependency trees in Figs. 2a & 2b. To analyze both words and relations of a core tree uniformly, we transform it into a *uniform graph format* (Figs. 2e & 2f) in which the core tree's words and relations are also represented as graph nodes.

We define a basic element of a relationship r as a key subtree that commonly appears in various core trees of r. As an example, the core trees in Figs. 2e & 2f share a common subtree in Fig. 2g. Intuitively, this subtree is shared by the core trees of sentences, that express the idea of *"joined the company as CEO"* or *"joined the company and did something as CEO"*.

We formally define the subtree as

- Assume that we have $T = (V, E)$, a directed tree, in which V is a set of nodes and E is a set of directed edges.
- Node y is called an ancestor of node x, denoted by $x \prec y$, if $(x, y) \in E$ or $\exists i_1, ..., i_k$ ($k \in \mathbb{N}$ and $k \geq 1$) such that $(x, i_1), (i_1, i_2), ..., (i_{k-1}, i_k), (i_k, y) \in E$.
- We define that a tree $S = (V_S, E_S)$ is a subtree of T if and only if: (i) $V_S \subset V$, and (ii) $\forall(x, y) \in E_S$, we have $x \prec y$ in T.

We use a subtree as a feature for relation extraction. From a set of training sentences with respect to a relationship r, we derive the core trees and transform them to the uniform graph formats. Therefore, it is necessary to generate all subtrees from the set of core trees to form the feature space. A frequent tree-mining algorithm (Zaki 2002) is used to generate subtrees from a set of core trees. A minimum support parameter is used in this algorithm to allow filtering: the filter yields only those subtrees that appear more frequently than the minimum support. Assuming that a relation has an appropriate core tree given a relationship, each mined subtree corresponds to a subtree feature value of the relation instance with respect to the relationship.

Supervised Learning for Relation Extraction

Figure 2b shows that there might be more than one relation that pertains between an entity pair. Therefore, we formulate our problem of relation classification into a multiclass and multi-label problem in which one SVM-based classifier is dedicated for a relation.

The *Sentence Selector* is run on the sentences of given training articles to select sentences containing at least one mention of a secondary entity as relation candidates to build training data. For a relation r, we manually select a set of positive instances, Pos_r, from relation candidates that actually express the relationship r between the entity pair. We also choose a separate set of negative instances in which no target relation is expressed. During training for the classifier of relation r, only instances in Pos_r serve as positive samples, all the others are used as negative samples.

We represent each mention of a secondary entity in a sentence with respect to a relation r as a feature vector receiving values 0 and 1 . Feature vectors are created from the type of principal entity (first eight slots), type of secondary entity (next eight slots), and the mined subtree of the sentence (the remaining slots). The number of slots for a subtree feature depends on the relation. The principal entity might be absent in a sentence and its type is unchanged for all sentences in an article.

During testing, a relation candidate is passed through all the classifiers. This time, we accumulate all the labels corresponding to the classifiers that return positive scores. Then, the accumulated labels are assigned to the relation candidate. No relation exists for the relation candidate if no classifier returns positive scores. It is noteworthy that each classifier receives a different feature vector because the subtree feature depends on which relation is being determined.

Table 3: Compare our proposed system and baselines (B_0 & B_1: baselines; $Deptree_0$: use only Entity type feature; $Deptree_1$: use only Subtree feature; $Deptree_2$: use both Subtree feature and Entity type feature)

	Precision(%)	**Recall(%)**	**F1(%)**
B_0	8.70	22.26	12.51
B_1	9.88	25.31	14.21
$DepTree_0$	16.73	41.79	23.89
$DepTree_1$	24.17	25.57	24.85
$DepTree_2$	**29.07**	**53.86**	**37.76**

Experiments and Evaluations

For experimentation, 5,975 articles are selected, of which 45 articles are for testing and 5,930 articles are used for training. We apply the framework in Fig. 1 on the training articles to extract keywords and select relation candidates. Subsequently, 3,833 positive instances and 805 negative instances from the candidates are annotated to train the *Relation Extractor*. Among 39,467 entities collected from all principal and secondary entities, we randomly select 3,300 entities and manually annotate their types to develop the *Entity Classifier*. Finally, 3,100 entities are used for training and 200 entities are used for testing.

We develop two baseline systems to evaluate our method, which uses a bag-of-words model. The second system (B_1 in Table 3) works like the *Keyword Extractor* on training instances in that it calculates *tf-idf* scores for words on the dependency path between the entities with respect to each relation. During testing, it accumulates the *tf-idf* scores of the words on the path and chooses the relation label that gives the highest score for the entity pair. The only difference between the two baseline systems is that the first one (B_0 in Table 3) does not use a dependency parse tree; instead, it specifically examines all the words between the entities in sentence text. In other words, we attempt to evaluate the contribution of syntactic information in this problem.

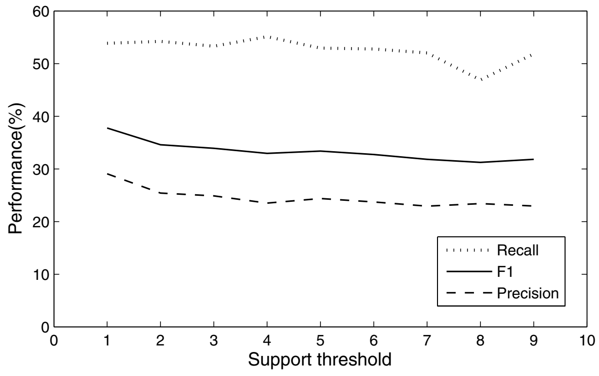

Figure 3: Performance of the system with various support thresholds

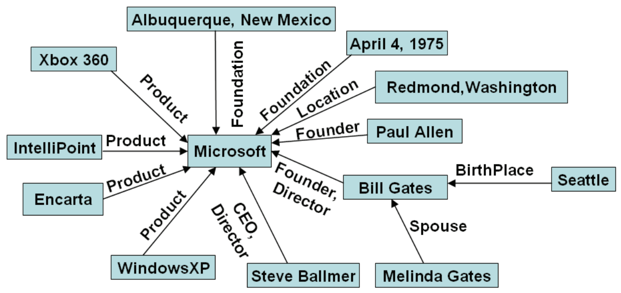

Figure 4: Some relations extracted by our system

In our experiments, dependency graphs of sentences are obtained using the Minipar parser (Lin 1998), all the classifiers are trained using SVM Light (Joachims 1999) with second-order polynomial kernel function and the frequent tree miner FREQT[5] is used to mine dependency subtrees.

Based on preliminary experiments, we can report the performance of our system compared to those of baseline systems in Table 3. Results show that our proposed method in $Deptree_2$ gives a substantial improvement over the baselines B_0 and B_1. In addition, using some information from the dependency tree in B_1 improves the performance slightly in B_0. The system $Deptree_0$ and $Deptree_1$ individually evaluates the performance of the entity type feature and subtree feature for relation classification. The best result is produced when both features are combined. Clearly, both features are important for this task. For the result shown in $Deptree_2$, although the recall is quite adequate, its precision is low. Data analysis reveals that our negative training set lacks useful negative samples to determine appropriate boundaries for classifiers. One more reason for the limited result is that using all the generated subtrees produces many irrelevant features. The method can be improved by performing feature selection on the generated subtrees.

We also report our system's performance when varying the minimum support parameter of the frequent tree-mining algorithm in Fig. 3. Although the high values of minimum support might remove the subtree features that occur infrequently in the training set, they also remove some useful features. Therefore, the best system is obtained when the minimum support is set to 1, meaning that all the subtrees that are mined from training data are used as features.

[5]http://chasen.org/~taku/software/freqt/

Table 4: Result of *Entity Classifier* to evaluate each feature and various values of K parameters (levels of exploited category structure)

	Depth *K*	Accuracy(%)
Without *pronoun feature*		80.5
Without *singular noun feature*		80.5
Without *category feature*		63.0
All features	1	64.0
	2	69.5
	3	81.0
	4	**81.5**
	5	79.5
	6	77.5
	7	77.0
	8	78.0
	9	75.0
	10	74.5

Table 4 shows the importance of each feature used in *Entity Classifier*. The performance is slightly degraded when we discard either *pronoun feature* or *singular noun feature*. However, removing *category feature* reduces the performance considerably. The classifier works best when we incorporate all the features and trace four levels on category hierarchy. Using more levels of category degrades the performance because it may collect many common abstract category items for each article and then leads to the existence of redundant features.

The interesting fact is that the high performance of this module allows Wikipedia to be used as an external knowledge source for Named Entity Recognition. Particularly, Wikipedia currently supports a search service that returns some of the most appropriate articles for a given name or phrase. The search result can be passed to our module to return the entity type for the input.

Finally, Figure 4 shows some relations extracted by the system. We can see various entities and relations extracted on *Microsoft* and *Bill Gates*.

Conclusions and Future Works

We have presented a method to extract relations among entities from Wikipedia articles by incorporating information from the Wikipedia structure and through analysis of Wikipedia text. The key features of our method include: (1) an algorithm to build a core syntactic tree that more accurately reflects the relation between a given entity pair; (2) the use of a tree-mining algorithm to identify the basic elements of syntactic structure of sentences for relationships; and (3) a method to make use of Wikipedia characteristics for entity allocation and entity classification.

Most of assumptions in this research are put on Wikipedia structure to enable effective entity allocation and entity classification. Those may constrain the applicability of our approach to other types of text but the main part of our algorithm, subtree feature for relation classification, can be applied to other texts. In case we apply the algorithm to other text-based content, we could use NER and Coreference Resolution tool to recognize entities.

As a subject for future work, we plan to conduct feature selection on subtree feature values for each classifier to remove irrelevant mined subtrees. Furthermore, we also intend to incorporate information from multiple articles to predict a single relation. For instance, clues from both the *Microsoft* and *Bill Gates* articles give stronger evidence supporting the *founder* relationship between the entities.

Additionally, we intend to make use of more Wikipedia features, such as the link structure or various list-like articles of Wikipedia. Aside from the summary sections, some articles provide information in the form of a list. Although they cannot be processed directly by machines, they are well structured, in contrast to free-form text.

References

Agichtein, E., and Gravano, L. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*.

Brin, S. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of the 1998 International Workshop on the Web and Databases*, 172–183.

Bunescu, R., and Mooney, R. 2006. Extracting relations from text: From word sequences to dependency paths. In Kao, A., and Poteet, S., eds., *Text Mining and Natural Language Processing*.

Cui, H.; Sun, R.; Li, K.; Kan, M.-Y.; and Chua, T.-S. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR-2005*.

Culotta, A., and Sorensen, J. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL-2004*.

Culotta, A.; McCallum, A.; and Betz, J. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of HLT-NAACL-2006*.

Gabrilovich, E., and Markovitch, S. 2006. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of AAAI-06*, 1301–1306.

Hsu, C.-W., and Lin, C.-J. 2002. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks* 13.

Joachims, T. 1999. Making large-scale svm learning practical. In Schölkopf, B.; Burges, C.; and Smola, A., eds., *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

Lin, D. 1998. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, the First International Conference on Language Resources and Evaluation*.

Morton, T. 2000. Coreference for nlp applications. In *Proceedings of ACL-2000*.

Pasca, M.; Lin, D.; Bigham, J.; Lifchits, A.; and Jain, A. 2006. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proceedings of AAAI-06*.

Ravichandran, D., and Hovy, E. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL-2002*, 41–47.

Soon, W.; Lim, D.; and Ng, H. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics* 27.

Strube, M., and Ponzetto, S. 2006. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of AAAI-06*, 1419–1424.

Völkel, M.; Krötzsch, M.; Vrandecic, D.; Haller, H.; and Studer, R. 2006. Semantic wikipedia. In *Proceedings of WWW2006*, 585–594.

Zaki, M. 2002. Efficiently mining frequent trees in a forest. In *Proceedings of KDD-2002*.