# Unsupervised Relation Extraction by Mining Wikipedia Texts Using Information from the Web

**Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang and Mitsuru Ishizuka**
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
yulan@mi.ci.i.u-tokyo.ac.jp
okazaki@is.s.u-tokyo.ac.jp
matsuo@biz-model.t.utokyo.ac.jp
yangzl@tkl.iis.u-tokyo.ac.jp
ishizuka@i.u-tokyo.ac.jp

## Abstract

This paper presents an unsupervised relation extraction method for discovering and enhancing relations in which a specified concept in Wikipedia participates. Using respective characteristics of Wikipedia articles and Web corpus, we develop a clustering approach based on combinations of patterns: dependency patterns from dependency analysis of texts in Wikipedia, and surface patterns generated from highly redundant information related to the Web. Evaluations of the proposed approach on two different domains demonstrate the superiority of the pattern combination over existing approaches. Fundamentally, our method demonstrates how deep linguistic patterns contribute complementarily with Web surface patterns to the generation of various relations.

## 1 Introduction

Machine learning approaches for relation extraction tasks require substantial human effort, particularly when applied to the broad range of documents, entities, and relations existing on the Web. Even with semi-supervised approaches, which use a large unlabeled corpus, manual construction of a small set of seeds known as true instances of the target entity or relation is susceptible to arbitrary human decisions. Consequently, a need exists for development of semantic information-retrieval algorithms that can operate in a manner that is as unsupervised as possible.

Currently, the leading methods in unsupervised information extraction collect redundancy information from a local corpus or use the Web as a corpus (Pantel and Pennacchiotti, 2006); (Banko et al., 2007); (Bollegala et al., 2007): (Fan et al., 2008); (Davidov and Rappoport, 2008). The standard process is to scan or search the corpus to collect co-occurrences of word pairs with strings between them, and then to calculate term co-occurrence or generate surface patterns. The method is used widely. However, even when patterns are generated from well-written texts, frequent pattern mining is non-trivial because the number of unique patterns is loose, but many patterns are non-discriminative and correlated. A salient challenge and research interest for frequent pattern mining is abstraction away from different surface realizations of semantic relations to discover discriminative patterns efficiently.

Linguistic analysis is another effective technology for semantic relation extraction, as described in many reports such as (Kambhatla, 2004); (Bunescu and Mooney, 2005); (Harabagiu et al., 2005); (Nguyen et al., 2007). Currently, linguistic approaches for semantic relation extraction are mostly supervised, relying on pre-specification of the desired relation or initial seed words or patterns from hand-coding. The common process is to generate linguistic features based on analyses of the syntactic features, dependency, or shallow semantic structure of text. Then the system is trained to identify entity pairs that assume a relation and to classify them into pre-defined relations. The advantage of these methods is that they use linguistic technologies to learn semantic information from different surface expressions.

As described herein, we consider integrating linguistic analysis with Web frequency information to improve the performance of unsupervised relation extraction. As (Banko et al., 2007) reported, "deep" linguistic technology presents problems when applied to heterogeneous text on the Web. Therefore, we do not parse information from the Web corpus, but from well written texts. Particularly, we specifically examine unsupervised relation extraction from existing texts of Wikipedia articles. Wikipedia resources of a fundamental type are of concepts (e.g., represented by Wikipedia articles as a special case) and their mutual relations. We propose our method, which groups concept pairs into several clusters based on the similarity of their contexts. Contexts are collected as patterns of two kinds: dependency patterns from dependency analysis of sentences in Wikipedia, and surface patterns generated from highly redundant information from the Web.

The main contributions of this paper are as follows:

- Using characteristics of Wikipedia articles and the Web corpus respectively, our study yields an example of bridging the gap separating “deep” linguistic technology and redundant Web information for Information Extraction tasks.
- Our experimental results reveal that relations are extractable with good precision using linguistic patterns, whereas surface patterns from Web frequency information contribute greatly to the coverage of relation extraction.
- The combination of these patterns produces a clustering method to achieve high precision for different Information Extraction applications, especially for bootstrapping a high-recall semi-supervised relation extraction system.

## 2 Related Work

(Hasegawa et al., 2004) introduced a method for discovering a relation by clustering pairs of co-occurring entities represented as vectors of context features. They used a simple representation of contexts; the features were words in sentences between the entities of the candidate pairs.

(Turney, 2006) presented an unsupervised algorithm for mining the Web for patterns expressing implicit semantic relations. Given a word pair, the output list of lexicon-syntactic patterns was ranked by pertinence, which showed how well each pattern expresses the relations between word pairs.

(Davidov et al., 2007) proposed a method for unsupervised discovery of concept specific relations, requiring initial word seeds. That method used pattern clusters to define general relations, specific to a given concept. (Davidov and Rappoport, 2008) presented an approach to discover and represent general relations present in an arbitrary corpus. That approach incorporated a fully unsupervised algorithm for pattern cluster discovery, which searches, clusters, and merges high-frequency patterns around randomly selected concepts.

The field of Unsupervised Relation Identification (URI)—the task of automatically discovering interesting relations between entities in large text corpora—was introduced by (Hasegawa et al., 2004). Relations are discovered by clustering pairs of co-occurring entities represented as vectors of context features. (Rosenfeld and Feldman, 2006) showed that the clusters discovered by URI are useful for seeding a semi-supervised relation extraction system. To compare different clustering algorithms, feature extraction and selection method, (Rosenfeld and Feldman, 2007) presented a URI system that used surface patterns of two kinds: patterns that test two entities together and patterns that test either of two entities.

In this paper, we propose an unsupervised relation extraction method that combines patterns of two types: surface patterns and dependency patterns. Surface patterns are generated from the Web corpus to provide redundancy information for relation extraction. In addition, to obtain semantic information for concept pairs, we generate dependency patterns to abstract away from different surface realizations of semantic relations. Dependency patterns are expected to be more accurate and less spam-prone than surface patterns from the Web corpus. Surface patterns from redundancy Web information are expected to address the data sparseness problem. Wikipedia is currently widely used information extraction as a local corpus; the Web is used as a global corpus.

## 3 Characteristics of Wikipedia articles

Wikipedia, unlike the whole Web corpus, has several characteristics that markedly facilitate information extraction. First, as an earlier report (Giles, 2005) explained, Wikipedia articles are much cleaner than typical Web pages. Because the quality is not so different from standard written English, we can use “deep” linguistic technologies, such as syntactic or dependency parsing. Secondly, Wikipedia articles are heavily cross-linked, in a manner resembling cross-linking of the Web pages. (Gabrilovich and Markovitch, 2006) assumed that these links encode numerous interesting relations among concepts, and that they provide an important source of information in addition to the article texts.

To establish the background for this paper, we start by defining the problem under consideration: relation extraction from Wikipedia. We use the encyclopedic nature of the corpus by specifically examining the relation extraction between the entitled concept (*ec*) and a related concept (*rc*), which are described in anchor text in this article. A common assumption is that, when investigating the semantics in articles such as those in Wikipedia (e.g. semantic Wikipedia (Volkel et al., 2006)), key information related to a concept described on a page $p$ lies within the set of links $l(p)$ on that page; particularly, it is likely that a salient semantic relation $r$ exists between $p$ and a related page $p' \in l(p)$.

Given the scenario we described along with earlier related works, the challenges we face are these: 1) enumerating all potential relation types of interest for extraction is highly problematic for corpora as large and varied as Wikipedia; 2) training data or seed data are difficult to label. Considering (Davidov and Rappoport, 2008), which describes work to get the target word and relation cluster given a single ('hook') word, their method depends mainly on frequency information from the Web to obtain a target and clusters. Attempting to improve the performance, our solution for these challenges is to combine frequency information from the Web and the "high quality" characteristic of Wikipedia text.

## 4 Pattern Combination Method for Relation Extraction

With the scene and challenges stated, we propose a solution in the following way. The intuitive idea is that we integrate linguistic technologies on high-quality text in Wikipedia and Web mining technologies on a large-scale Web corpus. In this section, we first provide an overview of our method along with the function of the main modules. Subsequently, we explain each module in the method in detail.

### 4.1 Overview of the Method

Given a set of Wikipedia articles as input, our method outputs a list of concept pairs for each article with a relation label assigned to each concept pair. Briefly, the proposed approach has four main modules, as depicted in Fig. 1.

- **Text Preprocessor and Concept Pair Collector** preprocesses Wikipedia articles to split text and filter sentences. It outputs concept pairs, each of which has an accompanying sentence.

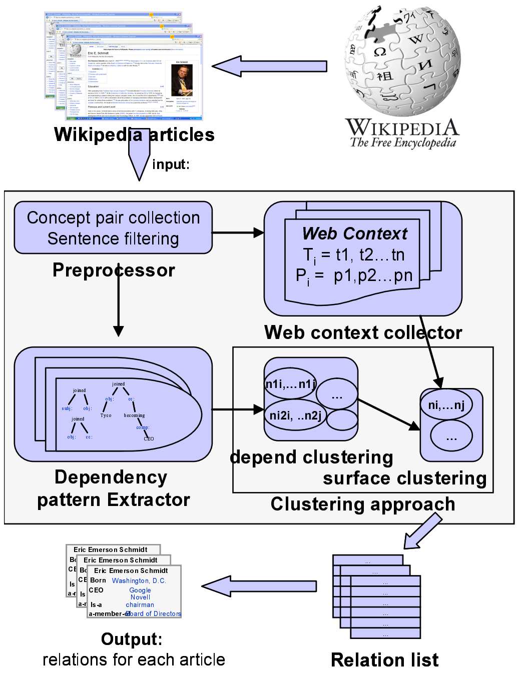

Figure 1: Framework of the proposed approach

- **Web Context Collector** collects context information from the Web and generates ranked relational terms and surface patterns for each concept pair.

- **Dependency Pattern Extractor** generates dependency patterns for each concept pair from corresponding sentences in Wikipedia articles.

- **Clustering Algorithm** clusters concept pairs based on their context. It consists of the two sub-modules described below.
  - **Depend Clustering**, which merges concept pairs using dependency patterns alone, aiming at obtaining clusters of concept pairs with good precision;
  - **Surface Clustering**, which clusters concept pairs using surface patterns based on the resultant clusters of depend clustering. The aim is to merge more concept pairs into existing clusters with surface patterns to improve the coverage of clusters.

## 4.2 Text Preprocessor and Concept Pair Collector

This module pre-processes Wikipedia article texts to collect concept pairs and corresponding sentences. Given a concept described in a Wikipedia article, our idea of preprocessing executes initial consideration of all anchor-text concepts linking to other Wikipedia articles in the article as related concepts that might share a semantic relation with the entitled concept. The link structure, more particularly, the structure of outgoing links, provides a simple mechanism for identifying relevant articles. We split text into sentences and select sentences containing one reference of an entitled concept and one of the linked texts for the dependency pattern extractor module.

## 4.3 Web Context Collector

Querying a concept pair using a search engine (Google), we characterize the semantic relation between the pair by leveraging the vast size of the Web. Our hypothesis is that there exist some key terms and patterns that provide clues to the relations between pairs. From the snippets retrieved by the search engine, we extract relational information of two kinds: ranked relational terms as keywords and surface patterns. Here surface patterns are generated with support of ranked relational terms.

### 4.3.1 Relational Term Ranking

To collect relational terms as indicators for each concept pair, we look for verbs and nouns from qualified sentences in the snippets instead of simply finding verbs. Using only verbs as relational terms might engender the loss of various important relations, e.g. noun relations "CEO", "founder" between a person and a company. Therefore, for each concept pair, a list of relational terms is collected. Then all the collected terms of all concept pairs are combined and ranked using an entropy-based algorithm which is described in (Chen et al., 2005). With their algorithm, the importance of terms can be assessed using the entropy criterion, which is based on the assumption that a term is irrelevant if its presence obscures the separability of the dataset. After the ranking, we obtain a global ranked list of relational terms $T_{all}$ for the whole dataset (all the concept pairs). For each concept pair, a local list of relational terms $T_{cp}$ is sorted according to the terms' order in $T_{all}$. Then from the relational term list $T_{cp}$, a keyword $t_{cp}$ is selected for each concept pair *cp* as the first term appearing in the term list $T_{cp}$. Keyword $t_{cp}$ will be used to initialize the clustering algorithm in Section 4.5.1.

Table 1: Surface patterns for a concept pair

| Pattern | Pattern |
|---|---|
| *ec* ceo *rc* | *rc* found *ec* |
| ceo *rc* found *ec* | *rc* succeed as ceo of *ec* |
| *rc* be ceo of *ec* | *ec* ceo of *rc* |
| *ec* assign *rc* as ceo | *ec* found by ceo *rc* |
| ceo of *ec rc* | *ec* found in by *rc* |

### 4.3.2 Surface Pattern Generation

Because simply taking the entire string between two concept words captures an excess of extraneous and incoherent information, we use $T_{cp}$ of each concept pair as a key for surface pattern generation. We classified words into Content Words (CWs) and Functional Words (FWs). From each snippet sentence, the entitled concept, related concept, or the keyword $k_{cp}$ is considered to be a Content Word (CW). Our idea of obtaining FWs is to look for verbs, nouns, prepositions, and coordinating conjunctions that can help make explicit the hidden relations between the target nouns.

Surface patterns have the following general form.

$$[\textbf{CW1}]\ Infix_1\ [\textbf{CW2}]\ Infix_2\ [\textbf{CW3}] \qquad (1)$$

Therein, $Infix_1$ and $Infix_2$ respectively contain only and any number of FWs. A pattern example is "*ec* assign *rc* as *ceo (keyword)*". All generated patterns are sorted by their frequency, and all occurrences of the entitled concept and related concept are replaced with "*ec*" and "*rc*", respectively for pattern matching of different concept pairs.

Table 1 presents examples of surface patterns for a sample concept pair. Pattern windows are bounded by CWs to obtain patterns more precisely because 1) if we use only the string between two concepts, it may not contain some important relational information, such as "ceo *ec* resign *rc*" in Table 1; 2) if we generate patterns by setting a windows surrounding two concepts, the number of unique patterns is often exponential.

## 4.4 Dependency Pattern Extractor

In this section, we describe how to obtain dependency patterns for relation clustering. After preprocessing, selected sentences that contain at least one mention of an entitled concept or related concept are parsed into dependency structures. We define dependency patterns as sub-paths of the shortest dependency path between a concept pair for two reasons. One is that the shortest path dependency kernels outperform dependency tree kernels by offering a highly condensed representation of the information needed to assess their relation (Bunescu and Mooney, 2005). The other reason is that embedded structures of the linguistic representation are important for obtaining good coverage of the pattern acquisition, as explained in (Culotta and Sorensen, 2005); (Zhang et al., 2006). The process of inducing dependency patterns has two steps.

1. Shortest dependency path inducement. From the original dependency tree structure by parsing the selected sentence for each concept pair, we first induce the shortest dependency path with the entitled concept and related concept.

2. Dependency pattern generation. We use a frequent tree-mining algorithm (Zaki, 2002) to generate sub-paths as dependency patterns from the shortest dependency path for relation clustering.

### 4.5 Clustering Algorithm for Relation Extraction

In this subsection, we present a clustering algorithm that merges concept pairs based on dependency patterns and surface patterns. The algorithm is based on k-means clustering for relation clustering.

The dependency pattern has the properties of being more accurate, but the Web context has the advantage of containing much more redundant information than Wikipedia. Our idea of concept pair clustering is a two-step clustering process: first it clusters concept pairs into clusters with good precision using dependency patterns; then it improves the coverage of the clusters using surface patterns.

#### 4.5.1 Initial Centroid Selection and Distance Function Definition

The standard k-means algorithm is affected by the choice of seeds and the number of clusters $k$. However, as we claimed in the Introduction section, because we aim to extract relations from Wikipedia articles in an unsupervised manner, cluster number $k$ is unknown and no good centroids can be predicted. As described in this paper, we select centroids based on the keyword $t_{cp}$ of each concept pair.

First of all, all concept pairs are grouped by their keywords $t_{cp}$. Let $G = \{G_1, G_2, ...G_n\}$ be the resultant groups, where each $G_i = \{cp_{i1}, cp_{i2}, ...\}$ identify a group of concept pairs sharing the same keyword $t_{cp}$ (such as "CEO"). We rank all the groups by their number of concept pairs and then choose the top $k$ groups. Then a centroid $c_i$ is selected for each group $G_i$ by Eq. 2.

$$c_i = \arg\max_{cp \in G_i} |\{cp_{ij}|(dis_1(cp_{ij}, cp) + \lambda * dis_2(cp_{ij}, cp)) <= D_z, 1 \leq j \leq |G_i|\}| \quad (2)$$

We assume a centroid for each group to be the concept pair which has the most other concept pairs in the same group that have distance less than $D_z$ with it. Also, $D_z$ is a threshold to avoid noisy concept pairs: we assign it 1/3. To balance the contribution between dependency patterns and surface patterns, $\lambda$ is used. The distance function to calculate the distance between dependency pattern sets $DP_i, DP_j$ of two concept pairs $cp_i$ and $cp_j$ is $dis_1$. The distance is decided by the number of overlapped dependency patterns with Eq. 3.

$$dis_1(cp_i, cp_j) = 1 - \frac{|DP_i \cap DP_j|}{\sqrt{(|DP_i| * |DP_j|)}} \quad (3)$$

Actually, $dis_2$ is the distance function to calculate distance between two surface pattern sets of two concept pairs. To compute the distance over surface patterns, we implement the distance function $dis_2(cp_i, cp_j)$ in Fig. 2.

```
Algorithm 1: distance function $dis_2(cp_i, cp_j)$
Input: $SP_1 = \{sp_{11}, ..., sp_{1m}\}$(surface patterns of
        $cp_i$)
$SP_2 = \{sp_{21}, ..., sp_{2n}\}$ (surface patterns of $cp_j$)
Output: $dis$ (distance between $SP_1$ and $SP_2$)
define a $m \times n$ distance matrix A:
$\{A_{ij} = \frac{LD(sp_{1i}, sp_{2j})}{Max(|sp_{1i}|, |sp_{2j}|)}$, 1≤i≤m; 1≤j≤n};
$dis \leftarrow 0$
for $\min(m, n)$ times do
    (x, y) $\leftarrow \mathrm{argmin}_{0<i<m;0<j<n} A_{ij}$;
    $dis \leftarrow dis + A_{xy}/min(m, n)$;
    $A_{x*} \leftarrow 1$; $A_{*y} \leftarrow 1$;
return $dis$
```

Figure 2: Distance function over surface patterns

As shown in Fig. 2, the distance algorithm performs as: firstly it defines a $m \times n$ distance matrix $A$, then repeatedly selects two nearest sequences and sums up their distances. While computing

$dis_2$, we use the Levenshtein distance $LD$ to measure the difference of two surface patterns. The Levenshtein distance is a metric for measuring the amount of difference between two sequences (i.e., the so-called edit distance). Each generated surface pattern is a sequence of words. The distance of two surface patterns is defined as the fraction of the $LD$ value to the length of the longer sequence.

For estimating the number of clusters $k$, we apply the stability-based criteria from (Chen et al., 2005) to decide the number of optimal clusters $k$ automatically.

### 4.5.2 Concept Pair Clustering with Dependency Patterns

Given the initial seed concept pairs and cluster number k, this stage merges concept pairs over dependency patterns into $k$ clusters. Each concept pair $cp_i$ has a set of dependency patterns $DP_i$. We calculate distances between two pairs $cp_i$ and $cp_j$ using above the function $dis_1(cp_i, cp_j)$. The clustering algorithm is portrayed in Fig. 3. The process of depend clustering is to assign each concept pair to the cluster with the closest centroid and then recomputing each centroid based on the current members of its cluster. As shown in Figure 3, this is done iteratively by repeating both two steps until a stopping criterion is met. We apply the termination condition as: centroids do not change between iterations.

**Algorithm 2**: Depend Clustering

```
Input: $I = \{cp_1, ..., cp_n\}$(all concept pairs)
       $C = \{c_1, ..., c_k\}$ (k initial centroids)
Output: $M_d : I \to C$ (cluster membership)
        $I_r$ (rest of concept pairs not clustered)
        $C_d = \{c_1, ..., c_k\}$ (recomputed centroids)
while stopping criterion has not been met do
    for each $cp_i \in I$ do
        if $\min_{s \in 1..k} dis_1(cp_i, c_s) <= D_l$ then
            $M_d(cp_i) \leftarrow \operatorname{argmin}_{s \in 1..k} dis_1(cp_i, c_s)$
        else
            $M_d(cp_i) \leftarrow 0$
    for each $j \in \{1..k\}$ do
        recompute $c_j$ as the centroid of
            $\{cp_i | m_{loc}(cp_i) = j\}$
$I_r \leftarrow C_0$
return $C$ and $C_d$
```

Figure 3: Clustering with dependency patterns

Because many concept pairs are scattered and do not belong to any of the top $k$ clusters, we filter concept pairs with distance larger than $D_l$ with the seed concept pairs. Such concept pairs are stored in $C_0$. We named the cluster of concept pairs $Ir$ which are left to be clustered in the next step of clustering. After this step, concept pairs with similar dependency patterns are merged into same clusters, see Fig. 4 (*ST1*, *ST2*).

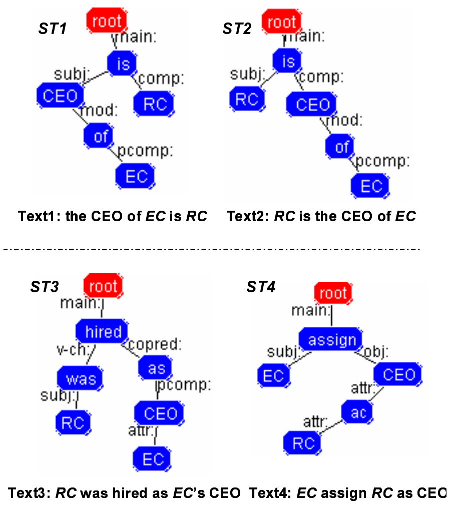

Figure 4: Example showing why surface clustering is needed

### 4.5.3 Concept Pair Clustering with Surface Patterns

A salient difficulty posed by dependency pattern clustering is that concept pairs of the same semantic relation cannot be merged if they are expressed in different dependency structures. Figure 4 presents an example demonstrating why we perform surface pattern clustering. As depicted in Fig. 4, $ST1$, $ST2$, $ST3$, and $ST4$ are dependency structures for four concept pairs that should be classified as the same relation "CEO". However $ST3$ and $ST4$ can not be merged with $ST1$ and $ST2$ using the dependency patterns because their dependency structures are too diverse to share sufficient dependency patterns.

In this step, we use surface patterns to merge more concept pairs for each cluster to improve the coverage. Figure 5 portrays the algorithm. We assume that each concept pair has a set of surface patterns from the Web context collector module. As shown in Figure 5, surface clustering is done iteratively by repeating two steps until a stopping criterion is met: using the distance function $dis_2$ explained in the preceding section, assign each concept pair to the cluster with the closest centroid and recomputing each centroid based on the current members of its cluster. We apply the same termination condition as depend clustering.

Additionally, we filter concept pairs with distance greater than $D_g$ with the centroid concept pairs.

```
Algorithm 3: Surface Clustering
Input: $I_r$ (rest of concept pairs)
       $C_d = \{c_1, ..., c_k\}$ (initial centroids)
Output: $M_s : I_r \rightarrow C$ (cluster membership)
       $C_s = \{c_1, ..., c_k\}$ (final centroids)
while stopping criterion has not been met do
    for each $cp_i \in I_r$ do
        if $\min_{s \in 1..k} dis_2(cp_i, c_s) <= D_g$ then
            $M_s(cp_i) \leftarrow \text{argmin}_{s \in 1..k} dis_2(cp_i, c_s)$
        else
            $M_s(cp_i) \leftarrow 0$
    for each $j \in 1..k$ do
        recompute $c_j$ as the centroid of cluster
            $\{cp_i | M_d(cp_i) = j \vee M_s(cp_i) = j\}$
return clusters C
```

Figure 5: Clustering with surface patterns

Finally we have $k$ clusters of concept pairs, each of which has a centroid concept pair. To attach a single relation label to each cluster, we use the centroid concept pair.

## 5 Experiments

We apply our algorithm to two categories in Wikipedia: "American chief executives" and "Companies". Both categories are well defined and closed. We conduct experiments for extracting various relations and for measuring the quality of these relations in terms of precision and coverage. We use coverage as an evaluation instead of using recall as a measure. The coverage is used to evaluate all correctly extracted concept pairs. It is defined as the fraction of all the correctly extracted concept pairs to the whole set of concept pairs. To balance between precision and coverage of clustering, we integrate two parameters: $D_l$, $D_g$.

We downloaded the Wikipedia dump as of December 3, 2008. The performance of the proposed method is evaluated using different pattern types: dependency patterns, surface patterns, and their combination. We compare our method with (Rosenfeld and Feldman, 2007)'s URI method. Their algorithm outperformed that presented in the earlier work using surface features of two kinds for unsupervised relation extraction: features that test two entities together and features that test only one entity each. For comparison, we use a k-means clustering algorithm using the same cluster number $k$.

Table 2: Results for the category: "American chief executives"

| method | Existing method (Rosenfeld et al.) | | Proposed method (Our method) | |
|---|---|---|---|---|
| Relation (sample) | # Ins. | pre | # Ins. | pre |
| **chairman** (*x be chairman of y*) | 434 | 63.52 | 547 | 68.37 |
| **ceo** (*x be ceo of y*) | 396 | 73.74 | 423 | 77.54 |
| **bear** (*x be bear in y*) | 138 | 83.33 | 276 | 86.96 |
| **attend** (*x attend y*) | 225 | 67.11 | 313 | 70.28 |
| **member** (*x be member of y*) | 14 | 85.71 | 175 | 91.43 |
| **receive** (*x receive y*) | 97 | 67.97 | 117 | 73.53 |
| **graduate** (*x graduate from y*) | 18 | 83.33 | 92 | 88.04 |
| **degree** (*x obtain y degree*) | 5 | 80.00 | 78 | 82.05 |
| **marry** (*x marry y*) | 55 | 41.67 | 74 | 61.25 |
| **earn** (*x earn y*) | 23 | 86.96 | 51 | 88.24 |
| **award** (*x won y award*) | 23 | 43.47 | 46 | 84.78 |
| **hold** (*x hold y degree*) | 5 | 80.00 | 37 | 72.97 |
| **become** (*x become y*) | 35 | 74.29 | 37 | 81.08 |
| **director** (*x be director of y*) | 24 | 67.35 | 29 | 79.31 |
| **die** (*x die in y*) | 18 | 77.78 | 19 | 84.21 |
| all | 1510 | 68.27 | 2314 | 75.63 |

### 5.1 Wikipedia Category: "American chief executives"

We choose appropriate $D_l$(concept pair filter in depend clustering) and $D_g$(concept pair filter in surface clustering) in a development set. To balance precision and coverage, we set 1/3 for both $D_l$ and $D_g$.

The 526 articles in this category are used for evaluation. We obtain 7310 concept pairs from the articles as our dataset. The top 18 groups are chosen to obtain the centroid concept pairs. Of these, 15 binary relations are the clearly identifiable relations shown in Table 2, where # Ins. represents the number of concept pairs clustered using each method, and *pre* denotes the precision of each cluster.

The proposed approach shows higher precision and better coverage than URI in Table 2. This result demonstrates that adding dependency patterns from linguistic analysis contributes more to the precision and coverage of the clustering task than the sole use of surface patterns.

Table 3: Performance of different pattern types

| Pattern type | #Instance | Precision | Coverage |
|---|---|---|---|
| dependency | 1127 | 84.29 | 13.00% |
| surface | 1510 | 68.27 | 14.10% |
| Combined | 2314 | 75.63 | 23.94% |

Table 4: Results for the category: "Companies"

| Method | Existing method (Rosenfeld et al.) | | Proposed method (Our method) | |
|---|---|---|---|---|
| Relation (sample) | # Ins. | pre | # Ins. | pre |
| **found** (*found x in y*) | 82 | 75.61 | 163 | 84.05 |
| **base** (*x be base in y*) | 82 | 76.83 | 122 | 82.79 |
| **headquarter** (*x be headquarter in y*) | 23 | 86.97 | 120 | 89.34 |
| **service** (*x offer y service*) | 37 | 51.35 | 108 | 69.44 |
| **store** (*x open store in y*) | 113 | 77.88 | 88 | 72.72 |
| **acquire** (*x acquire y*) | 59 | 62.71 | 70 | 64.28 |
| **list** (*x list on y*) | 51 | 64.71 | 67 | 70.15 |
| **product** (*x produce y*) | 25 | 76.00 | 57 | 77.19 |
| **CEO** (*ceo x found y*) | 37 | 64.86 | 39 | 66.67 |
| **buy** (*x buy y*) | 53 | 62.26 | 37 | 56.76 |
| **establish** (*x be establish in y*) | 35 | 82.86 | 26 | 80.77 |
| **locate** (*x be locate in y*) | 14 | 50.00 | 24 | 75.00 |
| all | 685 | 71.03 | 1039 | 76.87 |

To examine the contribution of dependency patterns, we compare results obtained with patterns of different kinds. Table 3 shows the precision and coverage scores. The best precision is achieved by dependency patterns. The precision is markedly better than that of surface patterns. However, the coverage is worse than that by surface patterns. As we reported, many concept pairs are scattered and do not belong to any of the top $k$ clusters, the coverage is low.

## 5.2 Wikipedia Category: "Companies"

We also evaluate the performance for the "Companies" category. Instead of using all the articles, we randomly select 434 articles for evaluation and 4073 concept pairs from the articles form our dataset for this category. We also set $D_l$ and $D_g$ to 1/3. Then 28 groups are chosen. For each group, a centroid concept pair is obtained. Finally, of 28 clusters, 25 binary relations are clearly identifiable relations. Table 4 presents some relations.

Table 5: Performance of different pattern types

| Pattern type | #Instance | Precision | Coverage |
|---|---|---|---|
| dependency | 551 | 82.58 | 11.17% |
| surface | 685 | 71.03 | 11.95% |
| Combined | 1039 | 76.87 | 19.61% |

Our clustering algorithms use two filters $D_l$ and $D_g$ to filter scattering concept pairs. In Table 4, we present that concept pairs are clustered with good precision. As in the first experiments, the combination of dependency patterns and surface patterns contribute greatly to the precision and coverage. Table 5 shows that, using dependency patterns, the precision is the highest (82.58%), although the coverage is the lowest.

All experimental results support our idea mainly in two aspects: 1) Dependency analysis can abstract away from different surface realizations of text. In addition, embedded structures of the dependency representation are important for obtaining a good coverage of the pattern acquisition. Furthermore, the precision is better than that of the string surface patterns from Web pages of various kinds. 2) Surface patterns are used to merge concept pairs with relations represented in different dependency structures with redundancy information from the vast size of Web pages. Using surface patterns, more concept pairs are clustered, and the coverage is improved.

## 6 Conclusions

To discover a range of semantic relations from a large corpus, we present an unsupervised relation extraction method using deep linguistic information to alleviate surface and noisy surface patterns generated from a large corpus, and use Web frequency information to ease the sparseness of linguistic information. We specifically examine texts from Wikipedia articles. Relations are gathered in an unsupervised way over patterns of two types: dependency patterns by parsing sentences in Wikipedia articles using a linguistic parser, and surface patterns from redundancy information from the Web corpus using a search engine. We report our experimental results in comparison to those of previous works. The results show that the best performance arises from a combination of dependency patterns and surface patterns.

## References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni. 2007. *Open information extraction from the Web.* In Proceedings of IJCAI-2007.

Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. 2007. *Measuring Semantic Similarity between Words Using Web Search Engines.* In Proceedings of WWW-2007.

Razvan C. Bunescu and Raymond J. Mooney. 2005. *A shortest path dependency kernel for relation extraction.* In Proceedings of HLT/EMLNP-2005.

Jinxiu Chen, Donghong Ji, Chew Lim Tan and Zhengyu Niu. 2005. *Unsupervised Feature Selection for Relation Extraction.* In Proceedings of IJCNLP-2005.

Aron Culotta and Jeffrey Sorensen. 2004. *Dependency tree kernels for relation extraction.* In Proceedings of the ACL-2004.

Dmitry Davidov, Ari Rappoport and Moshe Koppel. 2007. *Fully unsupervised discovery of concept-specific relationships by Web mining.* In Proceedings of ACL-2007.

Dmitry Davidov and Ari Rappoport. 2008. *Classification of Semantic Relationships between Nominals Using Pattern Clusters.* In Proceedings of ACL-2008.

Wei Fan, Kun Zhang, Hong Cheng, Jing Gao, Xifeng Yan, Jiawei Han, Philip S. Yu and Olivier Verscheure. 2008. *Direct Mining of Discriminative and Essential Frequent Patterns via Model-based Search Tree.* In Proceedings of KDD-2008.

Evgeniy Gabrilovich and Shaul Markovitch. 2006. *Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge.* In Proceedings of AAAI-2006.

Jim Giles. 2005. *Internet encyclopaedias go head to head.* Nature 438:900C901.

Sanda Harabagiu, Cosmin Adrian Bejan and Paul Morarescu. 2005. *Shallow semantics for relation extraction.* In Proceedings of IJCAI-2005.

Takaaki Hasegawa, Satoshi Sekine and Ralph Grishman. 2004. *Discovering Relations among Named Entities from Large Corpora.* In Proceedings of ACL-2004.

Nanda Kambhatla. 2004. *Combining lexical, syntactic and semantic features with maximum entropy models.* In Proceedings of ACL-2004.

Dat P.T. Nguyen, Yutaka Matsuo and Mitsuru Ishizuka. 2007. *Relation extraction from Wikipedia using subtree mining.* In Proceedings of AAAI-2007.

Patrick Pantel and Marco Pennacchiotti. 2006. *Espresso: Leveraging generic patterns for automatically harvesting semantic relations.* In Proceedings of ACL-2006.

Benjamin Rosenfeld and Ronen Feldman. 2006. *URES: an Unsupervised Web Relation Extraction System.* In Proceedings of COLING/ACL-2006.

Benjamin Rosenfeld and Ronen Feldman. 2007. *Clustering for Unsupervised Relation Identification.* In Proceedings of CIKM-2007.

Peter D. Turney. 2006. *Expressing implicit semantic relations without supervision.* In Proceedings of ACL-2006.

Max Volkel, Markus Krotzsch, Denny Vrandecic, Heiko Haller and Rudi Studer. 2006. *Semantic wikipedia.* In Proceedings of WWW-2006.

Mohammed J. Zaki. 2002. *Efficiently mining frequent trees in a forest.* In Proceedings of SIGKDD-2002.

Min Zhang, Jie Zhang, Jian Su and Guodong Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features.* In Proceedings of ACL-2006.