An Integrated Approach for Relation Extraction from Wikipedia Texts

Yulan Yan
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan
yulan@mi.ci.i.u-tokyo.ac.jp

Yutaka Matsuo
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan
matsuo@biz-model.t.utokyo.ac.jp

Mitsuru Ishizuka
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan
ishizuka@i.u-tokyo.ac.jp

ABSTRACT

Linguistic-based methods and web mining-based methods are two types of leading methods for semantic relation extraction task. By integrating linguistic analysis with frequent Web information, this paper presents an unsupervised relation extraction approach, for discovering and enhancing relations in which a specified concept participates. We focus on concepts described in Wikipedia articles. By making use of the characteristics of Wikipedia and Web corpus, we define a novel distance function and develop a linear clustering algorithm on the combination of two kinds of patterns: dependency patterns from dependency analysis of texts in Wikipedia, surface patterns generated from high redundant information from the Web corpus. The experiments on two different domains demonstrate the superiority of our approach comparing with previous method. In essence, our approach shows how deep linguistic features contribute complementally with Web surface features to generate a broad variety of relations.

1. INTRODUCTION

The machine learning approaches for relation extraction task require substantial human effort, particularly when applied to the broad range of documents, entities, and relations existing in the Web. Even with semi-supervised approaches which use a large unlabeled corpus, manual construction of a small set of seeds known true instances of the target entity or relation is susceptible to arbitrary human decisions. Hence, there is a need for developing semantic information retrieval algorithms that are as unsupervised as possible.

Currently the leading methods in unsupervised information extraction are based on collecting redundancy information from a local corpus or use the Web as 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-appearances of word pairs with strings between them, then calculate term co-occurrence or generate textual patterns. The method is used widely, however, even when patterns are generated from good-written texts, frequent pattern mining is non-trivial since the number of unique patterns is exponential but many are non-discriminative and correlated. One of the main challenges and research interest for frequent pattern mining is how to abstract away from different surface realizations of semantic relations to discover discriminative patterns efficiently.

Linguistic analysis is another effective technology for semantic relation extraction (see e.g., (Kambhatla, 2004; Bunescu et al., 2005; Harabagiu et al., 2005; Nguyen et l., 2007)). Currently, linguistic approaches for semantic relation extraction are almost exclusively supervised, relying on pre-specification of the desired relationship or hand-coding initial seed words or patterns. The common process is to generate linguistic patterns based on analysis of the syntactic, dependency or shallow semantic structure of text, then train to identify entity pairs which assume a relationship and classify them into pre-defined relationships. The advantage of these methods is using linguistic technologies to learn semantic information from different surface expressions.

In this paper, we consider of integrating linguistic analysis with redundancy Web information to improve the performance of unsupervised relation extraction. As some work (Banko et al., 2007) claimed, "heavy" linguistic technology runs into problems when applied to the heterogeneous text found on the Web. Therefore, we do not plan to parse information from the Web corpus, but from good-written texts. In particular, we focus on natural occurring texts of Wikipedia[1] articles. We are interested in extracting relations from Wikipedia articles. A fundamental type of Wikipedia resource is that of concepts (represented by Wikipedia articles) and relations between concepts. We propose our approach in which concept pairs are clustered into a number of clusters based on the similarity of their contexts. Contexts are collected as two kinds of patterns: dependency patterns from dependency analysis of sentences in Wikipedia, surface patterns generated from Web information through a Web search engine.

The main contributions of this paper are presented as follows:

- By making use of the characteristics of Wikipedia articles and Web corpus respectively, our study suggests an example to bridge the gap between "hard" linguistic technology and redundant Web information for information extraction tasks.
- Our experimental results reveal that relations can be extracted with good precision using linguistic patterns, while surface patterns from Web frequency information contribute greatly to the coverage of relation instances.
- The combination of patterns allows the clustering approach to achieve high precision for bootstrapping a high-recall semi-supervised relation extraction system.

The remainder of the paper is organized as follows. In section 2 we will consider related work. In section 3 we will define more precisely the problem we intend to solve, followed by an analysis of the characteristics of Wikipedia articles. In section 4 we present out

[1] http://en.wikipedia.org/wiki/Main_Page

CAW2.0 2009, April 21, 2009, Madrid, Spain.
.

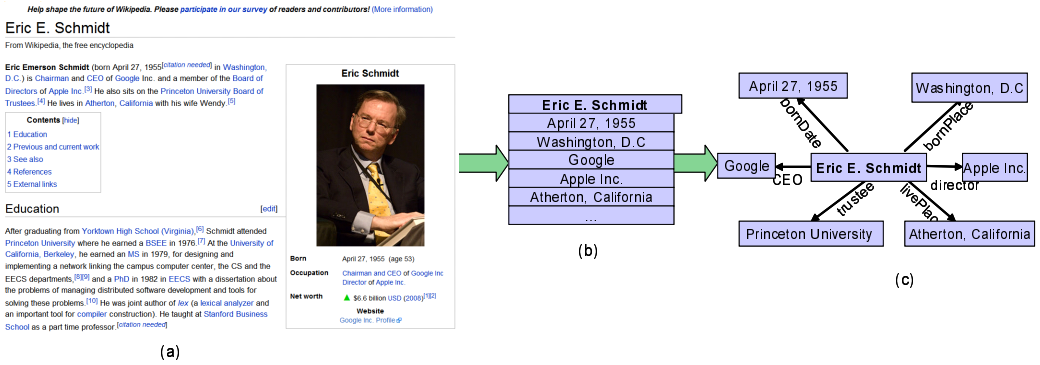

Figure 1: Entitled concept example "Eric E. Schmidt" for problem definition

the overview of our approach and describe it in detail. In section 5 we will report on our exploratory experimental results. Finally, in section 6 we will conclude the paper.

2. RELATED WORK

Discovering relations by clustering pairs of co-occurring entities represented as vectors of context features was introduced by (Hasegawa, et al., 2004). They used a simple representation of contexts - the features were the words that appear in sentences between the entities of the candidate pairs.

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

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

The field of Unsupervised Relation Identification (URI) - the task of automatically discovering interesting relations between entities in a large text corpora is introduced by Hasegawa, Sekine et al. (2006). Discovering relations by clustering pairs of co-occurring entities represented as vectors of context features. In (Rosenfeld and Feldman, 2006) they showed that the clusters discovered by URI can be used for seeding a semi-supervised relation extraction system. To compare different clustering algorithm, feature extraction and selection method, the authors in (Rosenfeld and Feldman, 2007) presented a URI system which used two kinds of surface patterns: patterns that test two entities together and patterns that test only one entity each.

In this study, we propose an unsupervised relation extraction approach upon previous work based on a combination of two types of patterns. On one hand, surface patterns are generated from the Web corpus to provide redundancy information for relation extraction. On the other hand, to obtain semantic information for concept pairs, we build a dependency pattern modeling module to generate dependency patterns, to abstract away from different surface realizations of semantic relations. Dependency patterns have the properties of being more accurate, less spam-prone and much less redundant than surface patterns from the Web corpus. The redundancy information can be used to ease the data sparseness problem. Wikipedia currently widely used as a corpus for information extraction is used as a local corpus and the Web is used as a global corpus.

3. WIKIPEDIA'S ARTICLE CHARACTERISTICS

Wikipedia, unlike the whole Web corpus, as one of the world's most popular Web sites, has several attributes that significantly facilitate information extraction: 1) high quality texts, as some work (Giles, 2005) claim that Wikipedia articles are much cleaner than typical Web pages, and mostly quality as standard written English. It is practical for us to rely on "heavy" linguistic technologies, such as syntactic or dependency parsing; 2) rich link structure, Wikipedia articles are heavily cross-linked, in a way reminiscent of cross-linking of the Web pages. These links are believed (Gabrilovich and Markovitch, 2006) to encode many interesting relations between concepts, and constitute an important source of information in addition to the article texts.

To set the scene for this paper, we start by defining the problem under consideration: major relation extraction from Wikipedia. We make use of the encyclopedic nature of the corpus by focussing on relation extraction between article entitled concept (ec) and one of related concepts (rc), which are described in hyperlinked text in this article. It is based on the common assumption: when investigating the semantics in articles like Wikipedia (e.g. semantic Wikipedia (Volkel et al., 2006)), key information on a concept described on a page p lies within the set of links $l(p)$ on that page; in particular it is likely that there is a salient semantic relation r between p and $p' \in l(p)$. Let us illustrate the scene with an example, shown in Figure 1. For the entitled concept represented in the article in Figure 1(a), related concepts to the entitled concept are extracted, shown in Figure 1(b). A special relational label is assigned to each concept pair which consists of a related concept and the entitled concept, shown in Figure 1(c).

With the scene we stated, the challenges we are facing are as follows: 1) enumerating all potential relation types of interest for extraction is highly problematic for corpora as large and varied as the Wikipedia; 2) training data or seed data is hard to be labeled. Considering (Davidov and Rappoport, 2008), which work to get target word and relationship cluster by given a single ('hook') word, their method mainly depends on frequency information from the Web to get target and clusters. Trying to improve the performance, our solution for these challenges is that besides frequency information from the Web or a local corpus, to make use of the high-quality property of Wikipedia text.

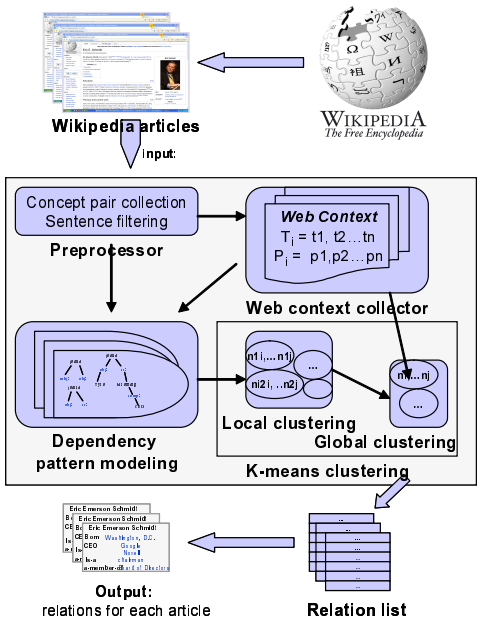

Figure 2: Outline of the unsupervised relation extraction approach

4. PATTERN COMBINATION APPROACH FOR RELATION EXTRACTION

With the scene and challenges stated, we propose a solution in the following way. The tuition idea is the combination of linguistic features and Web features. On one hand we apply linguistic technologies on high-quality text in Wikipedia, on the other hand, we apply web mining technologies on large scaled Web corpus. In this section, we firstly provide the overview of our approach along with the function of the main modules. Secondly, we explain each module in the approach in details.

4.1 Overview of the Approach

Our approach requires a set of Wikipedia articles as input; for each article, outputs a list of concept pairs with relation labels.

As shown in Figure 2, there are four primary modules of our approach:

- **Text Preprocessor and Relation Candidate Generation**, the module which preprocesses Wikipdia articles to split text and filter sentences. For each article, it collects a set of concept pairs as relation candidates.
- **Web Context Collector**, the module which collects context information from the Web corpus. For each relation candidate, it generates a set of ranked relational terms and a set of surface patterns.
- **Dependency Pattern Modeling**, the module which generates dependency patterns for each relation candidates from corresponding sentences in Wikipedia articles.
- **Linear Clustering Algorithm**, the module which clusters relation candidates based on their context. It contains two sub-modules:
  - **Local Clustering**, which merges instances using only dependency patterns generated from Dependency Pattern Modeling;
  - **Global Clustering**, which clusters instances using only textural patterns generated from Web Context Collector, based on the resulted clusters of local clustering. The aim is to merge more instances into existed clusters with surface patterns to improve the coverage of clusters.

The key to our approach lies in the complementary of Web frequency information from the Web and deep linguistic analysis, i.e., use linguistic information to extract relation instances with good precision and use Web frequency information to improve the coverage of relation instances.

4.2 Text Preprocessor and Relation Candidate Generation

This module pre-processes Wikipedia article texts to get relation candidates and corresponding sentences. Given each concept described in a Wikipedia article, our idea of preprocessing goes through firstly considering all hyper-linked concepts in the article as related concepts, which may share a semantic relationship with the entitled concept. The link structure, more particularly, the structure of outgoing links, provides a simple mechanism for identifying relevant articles. Secondly, we use a linguistic parser to split article text into sentences, and finally refine sentences which contain one of the references of entitled concept and one of the linked texts for the dependency pattern modeling module.

4.3 Web Context Collection

Querying with a concept pair through a search engine (we use Google), in this subsection, we try to characterize the semantic relation between them by leveraging the vast size of the Web. Our hypothesis is that there exist some key terms and patterns that provide clues to the relation the concept pair assume. From the resulting snippets of each query, we extract two kinds of relational information: a set of ranked relational terms as keywords and a set of surface patterns. Here surface patterns are generated with support of ranked relational terms.

4.3.1 Relational Term Ranking

Aiming to collect relational terms as indicators for each concept pair, we propose to look for verbs, nouns from qualified sentences in the resulting snippets, instead of simply finding verbs which will result in a loss of other links of importance, such as noun relation "CEO", "founder" between a person and a company.

For each input concept pair cp, our process of relational term collection is to retrieve a set of relation terms from all resulting snippets with all sentences containing this two concepts. Collected terms of all concept pairs are combined, and terms are ranked with an entropy-based algorithm which is described in detail in (Chen et al., 2005). After the ranking, we get a global ranked relational term list T_{all} for the whole data set. For each concept pair, a relational term list T_{cp} is ranked according to term order in T_{all}. From relational term list T_{cp}, a keyword k_{cp} is selected for each concept pair cp as the first term co-appearing in the term list T_{cp} and the corresponding Wikipedia sentence.

4.3.2 Surface Pattern Generation

As simply taking the entire string between two concept words captures an excess of extraneous and incoherent information, we propose to 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 resulting snippet sentence, entitled concept, related concept or the keyword k_{cp} is considered to be a Content Word (CW). Our idea of getting FWs as bag of

Table 1: Surface patterns for a sample 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*

words is to look for verbs, nouns, prepositions, and coordinating conjunctions that can help make explicit the hidden relations between the target nouns.

Our patterns have the general form:

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

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

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

This module is important in two aspects: firstly, it generate a ranked relational term list which assists to get more efficient surface pattern; secondly, for relations represented in different dependency structures, surface patterns collected by this mudule will be used to cluster instances which can not be clustered using dependency patterns.

4.4 Dependency Pattern Modeling

In this section, we will describe how to obtain dependency patterns for relation clustering. After the preprocessing, selected sentences which contains one of entitled concept references and one of the related concepts are parsed into dependency structures. As 1) shortest path dependency kernels outperform dependency tree kernels by offering a very condensed representation of the information needed to assess their relationship (Bunescu and Mooney, 2005); 2) embedded structures of the linguistic representation are important for obtaining a good coverage of the pattern acquisition (see e.g., (Culotta and Sorensen, 2004; Zhang et al., 2006)), we generate dependency patterns as sub-paths from the shortest dependency path for each concept pair. The process of inducing dependency patterns is in two steps:

1. Shortest dependency tree inducing. From the dependency tree structure, we firstly induce the shortest dependency path with the entitled concept and related concept.

2. Dependency pattern generation. Then we use a frequent tree-mining algorithm (Zaki et al., 2002) to generate dependency patterns from the shortest dependency path for relation clustering.

4.5 Linear Clustering Algorithm

In this subsection, we present an unsupervised relation clustering algorithm to merge concept pairs based on two kinds of generated patterns: dependency pattern and surface patterns. We proposed our linear clustering algorithm based on k-means clustering for relation clustering.

The dependency pattern modeling module has the properties of being more accurate, less spam-prone, but Web context has the advantage of containing much more redundant information than the Wikipedia. Our idea of relation instance clustering is: first cluster instance into clusters with good precision using dependency patterns in a local clustering step; then improve the coverage of clusters with more instances by using surface patterns in a global clustering step.

4.5.1 Distance Function and Initial Centroid Selection

The standard k-means algorithm depends on the choice of the seeds and on the number k of clusters, which must be known in advance. However, as we claimed in the introduction section, to extract relations from Wikipedia articles in an unsupervised way, cluster number k is unknown and no good centroids can be predicted. In this paper, we base the selection of centroids on the keyword t_{cp} of each concept pair.

Firstly, all concept pairs are grouped by their keywords t_{cp}. Let $G = \{G_1, G_2, ...G_n\}$ be the resulting groups, where each $G_i = \{cp_{i1}, cp_{i2}, ...\}$ identify a group of concept pairs who share the same keyword t_{cp} (such as "CEO"). With all the groups ranked by their number of instances, we choose the top k groups, and a centroid c_i is selected for each group G_i as follows:

$$s = \arg \max_{1 \le s \le |G_i|} |\{cp_{ij} | (\alpha * dis_1(cp_{ij}, cp_{is}) + \beta * dis_2(cp_{ij}, cp_{is})) <= D_z, 1 \le j \le |G_i|\}| \quad (2)$$

$$c_i = cp_{is} \quad (3)$$

where

$$dis_1(cp_{ij}, cp_{is}) = 1 - \frac{|DP_{cp_{ij}} \cap DP_{cp_{is}}|}{\sqrt{(|DP_{cp_{ij}}| * |DP_{cp_{is}}|)}} \quad (4)$$

To compute distance over surface patterns, we implement the distance function $dis_2(cp_{ij}, cp_{is})$ in Figure 3.

Algorithm 1: distance function $dis_2(cp_{ij}, cp_{is})$

Input: $SP_1 = \{sp_{11}, ..., sp_{1m}\}$(surface patterns of cp_{ij})
$SP_2 = \{sp_{21}, ..., sp_{2n}\}$ (surface patterns of cp_{is})
Output: dis (distance between SP_1 and SP_2)
define a $m \times n$ similarity matrix A: $\{A_{ij} = cost(sp_{1i}, sp_{2j})$ 1≤i≤m; 1≤j≤n};
$dis = 1$
for $\min(m, n)$ *times* **do**
  (x, y) = $\text{argmax}_{0<i<m;0<j<n} A_{ij}$;
  dis = dis - A_{xy};
  A_{x*} = 0; A_{*y} = 0;
return dis

Figure 3: Distance function over surface patterns

We selected the centroid to be the instance which has the most of other instances in the same group that have distance less than D_z with it. D_z is a threshold to avoid noisy instances, we assign it 1/3. dis_1 is the distance function to calculate distance between dependency pattern lists $DP_{cp_{ij}}, DP_{cp_{is}}$ of two concept pairs. The distance is decided by the number of shared dependency patterns. When computing dis_2, for cost function $cost(sp_{1i}, sp_{2j})$ which is used to calculate the similarity of two surface patterns, we use the dynamical programming computing which described in detail in (Rosenfeld and Feldman, 2006). α and β are used to leverage between dependency patterns and surface patterns, in this work we assign them both 1/2.

As for estimating the number of clusters, in this work we apply the stability-based criterions from (Chen, et al, 2005) to decide the number k of optimal clusters.

4.5.2 Local Dependency Pattern Clustering

The purpose of this stage is that given the initial seed instances and cluster number k, to merge relation instances over dependency patterns into k clusters. Each concept pair cp_{ij} has a set of dependency patterns $DP_{cp_{ij}}$, we calculate distances between two pairs cp_{ij} and cp_{is} with above the function $dis_1(cp_{ij}, cp_{is})$. The clustering algorithm is shown in Figure 4.

Algorithm 2: localClustering

Input: $I = \{cp_1, ..., cp_n\}$(all instances)
$C = \{c_1, ..., c_k\}$ (k initial centroids)
Output: $m_{loc} : I \rightarrow C$ (cluster membership)
I_{rest} (rest of instances not clustered)
$C_{loc} = \{c_1, ..., c_k\}$ (recomputed centroids)
for *each* $cp_i \in I$ **do**
  if $\min_{s \in 1..k} dis_1(cp_i, c_s) <= D_l$ **then**
    $m_{loc}(cp_i) = \text{argmin}_{s \in 1..k}\, dis_1(cp_i, c_s)$
  else
    $m_{loc}(cp_i) = 0$; $I_{rest} \leftarrow cp_i$
for *each* $j \in \{1..k\}$ **do**
  recompute c_j as the centroid of
    $\{cp_i | m_{loc}(cp_i) = j\}$

Figure 4: Clustering with dependency patterns

Since there are many concept pairs which are scattered and actually do not belong to any of the top k clusters, we filter concept pairs that with distance larger than D_l with the seed instances, to make sure the precision of the clustering. The rest instances not clustered are stored in I_{rest}. After this step, relation instances with similar dependency patterns are merged into same clusters, see Figure 5 (*ST1*, *ST2*).

4.5.3 Global Surface Pattern Clustering

One major problem faced by the local clustering is the fact that instances of the same semantic relationship which are represented in different dependency structures will not be merged into the same cluster. As shown in Figure 5 (*ST1*, *ST2*, *ST3*, and *ST4*), they are all instances of relationship "CEO", but (*ST3* and *ST4*)) can not be merged with (*ST1*) and (*ST2*) using only dependency patterns because their dependency structure are too different to share enough dependency patterns.

In this step of clustering, we use surface patterns to merge more instance for each cluster to improve the coverage performance using the algorithm shown in Figure 6.

Each concept pair has a set of surface patterns from the Web context collector module, to measure the distance between two instances, we use the distance function dis_2 explained in the above section. Also, we filter concept pairs with distance larger than D_g with the seed instances, to make sure the precision of the clustering.

Finally we have k clusters of instances, each cluster has a centroid instance. To attach a single relationship label each cluster, we use the centroid instance and assign the keyword of it as the relation label.

5. EXPERIMENTS

In this section we wish to consider the variety of relations that can be generated by our approach from Wikipedia, and to measure the quality of these relations in terms of their precision and coverage. To balance between precision and coverage of clustering, our approach uses several parameters: D_l, D_g, k. For purposes of evaluation, we ran our algorithm on two categories from Wikipedia categorization - "American chief executives" and "Companies". This

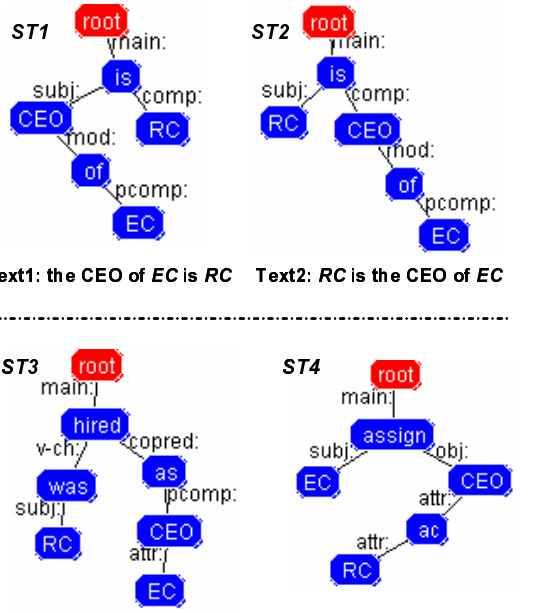

Figure 5: An example showing why we need global clustering

Algorithm 3: globalClustering

Input: I_{rest} (rest of instances)
$C_{loc} = \{c_1, ..., c_k\}$ (initial centroids)
Output: $m_{glo} : I_{rest} \rightarrow C$ (cluster membership)
$C = \{c_1, ..., c_k\}$ (final centroids)
for *each* $cp_i \in I_{rest}$ **do**
  if $\min_{s \in 1..k} dis_2(cp_i, c_s) <= D_g$ **then**
    $m_{glo}(cp_i) = \text{argmin}_{s \in 1..k}\, dis_2(cp_i, c_s)$
  else
    $m_{glo}(cp_i) = 0$
for *each* $j \in 1..k$ **do**
  recompute c_j as the centroid of cluster
    $\{cp_i | m_{loc}(cp_i) = j \vee m_{glo}(cp_i) = j\}$
return clusters C

Figure 6: Clustering with surface patterns

choice of domains allows us to explore different aspects of algorithmic behavior. "American chief executives" and "company" domains are both well defined and closed domains.

English version of Wikipedia dump on 03/12/2008 is our corpus. The performance of the proposed approach is evaluated on different pattern types: dependency patterns, surface patterns and the combination of both. We compare our approach with (Rosenfeld and Feldman, 2007)'s URI method, which showed that their algorithm improved over previous work using two kinds of surface features for unsupervised relation extraction: features that test two entities together and features that test only one slot each. For the purpose of comparison, we use k-means clustering algorithm and choose the same k as our approach when applying their approach.

Note that for our evaluation purposes, since there are many scattered concept pairs which actually do not belong to any of the top k clusters, it is very difficult to measure recall, so we measured coverage (the number of instance clustered over the whole instance set). Two evaluation measures (precision, coverage) are computed manually, the following quantities are considered to compute precision

Table 2: Results on 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

and coverage:

- p = the number of clustered instances.
- p+ = the number of clustered instances which actual belong to that cluster.
- n = the number of the whole set of instances.

$$\text{Precision } (P) = \text{p+/p} \qquad \text{coverage } (R) = \text{p/n}$$

5.1 Wikipedia Category: "American chief executives"

In the first series of experiments we build a development phrase to select appropriate D_l (instance filter in local clustering) and D_g(instance filter in global clustering). To balance between precision and coverage, we assign 1/3 for both D_l and D_g.

526 articles in this category are used for developing. We get 7310 concept pairs from the articles as our data set. Top 18 Groups are chosen to get the centroid instances. Of these, 15 binary relations are clearly identifiable relations which are shown in Table 2, where # Ins. represents the number of instances clustered by each method, and *pre* shows the precision of each cluster.

The proposed approach shows the higher precision and better coverage than URI in Table 2. This demonstrates that adding dependency patterns from linguistic analysis contribute greatly to the precision and coverage of the clustering task than using only surface patterns.

Table 3: Performance of different pattern types

Pattern type	#Instance	Precision	Coverage
dependency	1127	84.29	8.63%
surface	1510	68.27	9.39%
Combined	2314	75.63	15.81%

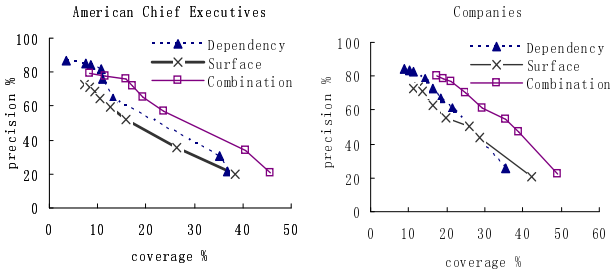

Figure 7: Precision-coverage curves on two categories

For further view of the contribution of dependency pattern, we experiment with dependency pattern separately to compare the results with using only surface patterns or combined patterns. The results are shown in Table 3. The best precision is achieved with dependency patterns, significantly better than surface patterns, though the coverage is the lowest, showing the sparseness of dependency patterns. The coverage is evaluated as the proportion of correctly extracted instances to the whole data set (7310 concept pairs). We use coverage as a relatively evaluation instead of using recall as measure, since for unsupervised task, it is difficult to evaluate with the recall. As we claimed, there are many concept pairs which are scattered and actually do not belong to any of the top k clusters, thus the coverage is low.

5.2 Wikipedia Category: "Companies"

In our second experiments, our approach is used to generate related concepts and clusters for concepts in the category "Companies". Instead of using all the articles, we randomly select 434 articles for developing, 4935 concept pairs from the articles form our data set for this category. We also assign D_l(instance filter in local clustering) and D_g(instance filter in global clustering) to 1/3. 28 groups are used to get the centroid instance and each centroid instance leads to one cluster finally, of these, 25 binary relations are clearly identifiable relations, some of which are shown in Table 4.

As Table 4 shows, two filters D_l and D_g are used to help merge instances with good precision. Similar to the first experiments, the combination of dependency patterns from linguistic analysis and surface patterns contributes greatly to the precision and coverage of the clustering task than using only surface patterns. Table 5 also shows that by using dependency patterns, the precision is the highest (82.58%), though the coverage is the lowest.

We then consider how performance changes for different values of D_l and D_g. The results are given as graphs in Figure 7. It show the precision and coverage over different pattern sets for each categories. The performance is boosted with the combination of patterns in precision and coverage.

All the experimental results support our idea mainly in two aspects: 1) dependency analysis can abstract away from different surface realizations of text and embedded structures of the dependency representation are important for obtaining a good coverage of the pattern acquisition. And the precision is better than string sur-

Table 4: Results on 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

Table 5: Performance of different pattern types

Pattern type	#Instance	Precision	Coverage
dependency	551	82.58	11.17%
surface	685	71.03	13.74%
Combined	1039	76.87	21.06%

face patterns from various kinds of Web pages; 2) surface patterns are used to merge instances with relations represented in different dependency structures with redundancy information from the vast size of Web pages, by using surface patterns, more instances can be clustered, the coverage is improved.

6. CONCLUSIONS

To discover a range of semantic relationships from large-scale corpus, we present an unsupervised relation extraction approach to use deep linguistic information to alleviate surface and noisy surface patterns generated from large corpus, and use Web frequency information to easy the sparseness of linguistic information. In particular, we focus on natural occurring texts from Wikipedia articles. Relations are gathered in an unsupervised way over two types of patterns: dependency patterns by parsing sentences in Wikipedia articles using a linguistic parser, and surface patterns from redundancy information from the Web corpus by using a search engine. We report our experimental results comparing to previous work and evaluating over using different patterns. The results show that the performance is the best with the combination of dependency patterns and surface patterns.

7. REFERENCES

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead and O. Etzioni. 2007. *Open information extraction from the Web.* In proceedings of IJCAI-2007.

D. Bollegala, Y. Matsuo and M. Ishizuka. 2007. *Measuring Semantic Similarity between Words Using Web Search Engines.* In Proceedings of WWW-2007.

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

J. Chen, D. Ji, C.L. Tan, and Z. Niu. 2005. *Unsupervised Feature Selection for Relation Extraction.* In Proceedings of IJCNLP-2005.

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

D. Davidov, A. Rappoport and M. Koppel. 2007. *Fully unsupervised discovery of concept-specific relationships by Web mining.* In Proceedings of ACL-2007.

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

W. Fan, K. Zhang, H. Cheng. and J. Gao. 2008. *Direct Mining of Discriminative and Essential Frequent Patterns via Model-based Search Tree.* In Proceedings of KDD-2008.

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

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

S. Harabagiu, C.A. Bejan and P. Morarescu. 2005. *Shallow semantics for relation extraction.* In Proceedings of IJCAI-2005.

T. Hasegawa, S. Sekine, and R. Grishman. 2004. *Discovering Relations among Named Entities from Large Corpora.* In Proceedings of ACL-2004.

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

D. P. T. Nguyen, Y. Matsuo and M. Ishizuka. 2007. *Relation extraction from wikipedia using subtree mining.* In Proceedings of AAAI-2007.

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

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

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

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

M. Volkel, M. Krotzsch, D. Vrandecic, H. Haller and R. Studer. 2006. *Semantic wikipedia.* In Proceedings of WWW-2006.

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

M. Zhang, J. Zhang, J. Su and G. Zhou. 2006. *A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features.* In Proceedings of ACL-2006.