Measuring the Similarity between Implicit Semantic Relations using Web Search Engines

Danushka Bollegala[*]
The University of Tokyo
Hongo 7-3-1, Tokyo
113-8656, Japan
danushka@mi.ci.i.u-tokyo.ac.jp

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

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

ABSTRACT

Measuring the similarity between implicit semantic relations is an important task in information retrieval and natural language processing. For example, consider the situation where you know an entity-pair (e.g. *Google, YouTube*), between which a particular relation holds (e.g. acquisition), and you are interested in retrieving other entity-pairs for which the same relation holds (e.g. *Yahoo, Inktomi*). Existing keyword-based search engines cannot be directly applied in this case because in keyword-based search, the goal is to retrieve documents that are relevant to the words used in the query – not necessarily to the relations implied by a pair of words. Accurate measurement of relational similarity is an important step in numerous natural language processing tasks such as identification of word analogies, and classification of noun-modifier pairs. We propose a method that uses Web search engines to efficiently compute the relational similarity between two pairs of words. Our method consists of three components: representing the various semantic relations that exist between a pair of words using automatically extracted lexical patterns, clustering the extracted lexical patterns to identify the different semantic relations implied by them, and measuring the similarity between different semantic relations using an inter-cluster correlation matrix. We propose a pattern extraction algorithm to extract a large number of lexical patterns that express numerous semantic relations. We then present an efficient clustering algorithm to cluster the extracted lexical patterns. Finally, we measure the relational similarity between word-pairs using inter-cluster correlation. We evaluate the proposed method in a relation classification task. Experimental results on a dataset covering multiple relation types show a statistically significant improvement over the current state-of-the-art relational similarity measures.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Relational similarity measures, Web mining

1. INTRODUCTION

Similarity measures can be broadly categorized into two types: *attributional* similarity measures and *relational* similarity measures. In attributional similarity measures, the objective is to compute the similarity between two given words by comparing the attributes of each word. If two words show a high degree of attributional similarity they are called *synonyms*. For example, the two words *car* and *automobile* share many attributes (e.g. *has wheels*, *is used for transportation*) between them. Consequently, they are considered as synonyms. On the other hand, relational similarity is the correspondence between semantic relations that exist between two *word-pairs*. Word-pairs that show a high degree of relational similarity are considered as *analogies*. For example, consider the two word-pairs *(ostrich, bird)* and *(lion, cat)*. Ostrich *is a large* bird and lion *is a large* cat. The implicit relation, *is a large*, holds between the two words in each word-pair. We propose a method that uses a Web search engine to compute the similarity between such implicitly stated relations in word-pairs.

Measuring the similarity between semantic relations is an important intermediate step in various tasks in information retrieval and natural language processing. For example, in relation extraction [5, 6, 36], the goal is to retrieve instances of a given relation. For example, given the relation, ACQUIRER-ACQUIREE, a relation extraction system must extract the instance (*Google*, *YouTube*) from the sentence *Google completed the acquisition of YouTube*. Bootstrapping methods [19, 4] that require a small number of seeds (ca. 10 pairs of instances per relation) have been successfully used to extract a large number of candidate instance-pairs from a text corpus. Given a set of candidate instance-pairs, a relational similarity measure can be used to find the similarity between the relations in the seeds and in the candidates. Candidate instance-pairs with high relational similarity with the seed-pairs can then be selected as the correct instances of a relation.

Relational similarity measures have been successfully used to find word analogies [9, 18, 26, 28, 33]. Scholastic Aptitude Test (SAT)'s word-analogy questions have been used to benchmark relational similarity measures. A SAT word-analogy question consists of a stem word-pair that acts as the question and five choice word-pairs, out of which only one is analogous to the stem. A relational similarity measure can be used to compare the stem word-pair with

[*]Research Fellow of the Japan Society for the Promotion of Science (JSPS)

WSDM'09, February 10-12, 2009, Barcelona, Spain.

each choice word-pair, and select the choice word-pair with the highest relational similarity as the correct answer.

An interesting application of relational similarity in information retrieval is to search using implicitly stated analogies [15, 32]. For example, the query "Muslim church" should return "mosque" and the query "Hindu bible" should return "the Vedas". These queries can be formalized as word-pairs: (Christian, Church) vs (Muslim,X), and (Christian, Bible) vs (Hindu,Y). We can then find the words X and Y that maximize the relational similarity in each case.

Despite the wide applications of relational similarity measures, accurately measuring the similarity between implicitly stated relations, remains a challenging task because of several reasons. First, relational similarity is a dynamic phenomenon that varies with time. For example, two companies can be initially competitors and subsequently one company might acquire the other. Second, there can be more than one relation between a word-pair. For example, between the two words in the word-pair *(ostrich, bird)*, besides the relation *is a large*, there is also the relation *is a flightless*. A relational similarity measure must first extract all relations between the two words in each word-pair before it can compute the similarity between the word-pairs. Third, there can be more than one way a particular semantic relation can be expressed in a text. For example, the three patterns: ***X** was acquired by **Y***, ***X** completed the acquisition of **Y***, and ***X** buys **Y***, all indicate an acquisition relation between **X** and **Y**. In addition to the above mentioned problems, measuring relational similarity between pairs in which one or both words are named-entities (e.g. company names, personal names, locations etc.) is even more difficult because such words are not well covered by manually created dictionaries such as WordNet[1] [17]. As we further explain in section 2, most of the previously proposed relational similarity measures either assume the availability of manually created resources or evaluated on common English words, not named entities.

We propose a relational similarity measure that uses a Web search engine to measure the similarity between implicitly stated semantic relations in two word-pairs. Formally, given two word-pairs, *(a,b)* and *(c,d)*, we design a function, *relsim*$((a,b),(c,d))$, that returns a similarity score in the range $[0,1]$. The proposed relational similarity measure first extracts implicitly stated relations that exist between the two words in each word-pair, and then compares the extracted relations between word-pairs.

Our contributions are summarized as follows:

- We propose a shallow, lexical patterns-based approach to represent the various semantic relations that exist between the two words in a given word-pair. The proposed pattern extraction algorithm does not require any language dependent preprocessing steps such as part-of-speech tagging or dependency parsing, which can be time consuming or even infeasible at Web scale. We extract a large number of lexical patterns that describe various semantic relations.

- We present an efficient sequential clustering algorithm to cluster lexical patterns, to identify the different patterns that describe a particular semantic relation. The proposed clustering algorithm requires only one pass through the set of extracted patterns, thus scales linearly with the number of patterns.

- We manually create a dataset of entity-pairs covering five relation types (our dataset is explained in section 4.1). We compare the proposed method against the state-of-the-art Latent Relational Analysis (LRA) [28] and the Vector Space Model-based approach (VSM) [30] on this dataset in a relation classification task. Experimental results show that the proposed method significantly outperforms both LRA and VSM methods.

2. RELATED WORK

Relational similarity has been studied in various fields under different contexts. In this section, we briefly review the previous work on relational similarity.

The Structure-Mapping Theory (SMT) [10] claims that an analogy is a mapping of knowledge from one domain (base) into another (target), which conveys that a system of relations known to hold in the base also holds in the target. The target objects do not have to resemble their corresponding base objects. This structural view of analogy is based on the intuition that analogies are about relations, rather than simple features. Although this approach works best when the base and the target are rich in higher-order causal structures, it can fail when structures are missing or flat [34].

Turney et al. [30] combined 13 independent modules by considering the weighted sum of the outputs of each individual module to solve SAT analogy questions. The best performing individual module was based on the Vector Space Model (VSM). In the VSM approach [29], first a vector is created for a word-pair *(X,Y)* by counting the frequencies of various lexical patterns containing *X* and *Y*. In their experiments they used 128 manually created patterns such as "*X* of *Y*", "*Y* of *X*", "*X* to *Y*" and "*Y* to *X*". These patterns are then used as queries to a search engine and the number of hits for each query is used as elements in a vector to represent the word-pair. Finally, relational similarity is computed as the cosine of the angle between the two vectors that represent the two word-pairs. Turney et al. [30] introduced a dataset containing 374 SAT analogy questions to evaluate relational similarity measures. A SAT analogy question consists of a stem word-pair that acts as the question, and five choice word-pairs. The choice word-pair that has the highest relational similarity with the stem word-pair is selected by the system as the correct answer. The average SAT score reported by high school students for word-analogy questions is 57%. The VSM approach achieves a score of 47% on this dataset.

Turney [26, 28] proposed Latent Relational Analysis (LRA) by extending the VSM approach in three ways: a) lexical patterns are automatically extracted from a corpus, b) the Singular Value Decomposition (SVD) is used to smooth the frequency data, and c) synonyms are used to explore variants of the word-pairs. Likewise in VSM approach, LRA represents a word-pair by a vector of lexical pattern frequencies. First, using a thesaurus, he finds related words for the two words in a word-pair and create additional word-pairs that are related to the original word-pairs in the dataset. Second, n-grams of words are extracted from the contexts in which the two words in a word-pair co-occur. Most frequent n-grams are selected as lexical patterns to represent a word-pair. Then a matrix of word-pairs vs. lexical patterns is created for all the word-pairs in the original dataset and the additional word-pairs. Elements of this matrix correspond to the frequency of a word-pair in a lexical pattern. Singular value decomposition is performed on this matrix to reduce the number of columns (i.e. patterns). Finally, the relational similarity between two word-pairs is computed as the average cosine similarity over the original word-pairs and the additional word-pairs derived from them. LRA achieves a score of 56.4% on SAT analogy questions.

Both VSM and LRA require a large number of search engine queries to create a vector to represent a word-pair. For example, with 128 patterns, VSM approach requires at least 256 queries to create the two pattern-frequency vectors for two word-pairs before

[1] http://wordnet.princeton.edu/

it can compute relational similarity. LRA considers synonymous variants of the given word-pairs, thus requires even more search engine queries. Methods that require a large number of queries impose a heavy load on search engines. Despite efficient implementations, singular value decomposition of large matrices is time consuming. In fact, LRA takes over 8 days to process the 374 SAT analogy questions [28]. This is problematic when computing relational similarity on the scale of the Web. Moreover, in the case of named-entities, related words thesauri are not usually available or not complete, which becomes a problem when creating the additional word-pairs required by LRA.

Veale [33] proposed a relational similarity measure based on the taxonomic similarity in WordNet. He evaluates the quality of a candidate analogy *A:B::C:D* (i.e. *A* to *B* as *C* to *D*), by comparing the paths in the WordNet, joining *A* to *B* and *C* to *D*. Relational similarity is defined as the similarity between the *A:B* paths and *C:D* paths. However, WordNet does not fully cover named-entities such as personal names, organizations and locations, which becomes problematic when using this method to measure relational similarity between named-entities.

Using a relational similarity measure, Turney [27] proposed an unsupervised learning algorithm to extract patterns that express implicit semantic relations from a corpus. His method produces a ranked set of lexical patterns that unambiguously describes the relation between the two words in a given word-pair. Patterns are ranked according to their expected relational similarity (i.e. pertinence), computed using an algorithm similar to LRA. To answer a SAT analogy question, first, ranked lists of patterns are generated for each of the six word pairs (one stem word-pair and five choice word-pairs). Then each choice is evaluated by taking the intersection of its patterns with the stem's patterns. The shared patterns are scored by the average of their rank in the stem's list and the choice's lists. The algorithm picks the choice with the lowest scoring shared pattern as the correct answer. This method reports a SAT score of 54.6%.

Relational similarity measures have been applied in natural language processing tasks such as generating word-analogies [9], and classifying noun-modifier compounds based on the relation between the head and the modifier [28, 18, 8]. Davidov and Rappoport [9] proposed an unsupervised algorithm to discover general semantic relationships that hold between lexical items. They represent a semantic relationship with a cluster of patterns. They use the pattern clusters to generate SAT-like word analogy questions for English and Russian languages. The generated questions are then solved by human subjects. They do not evaluate their method for relational similarity between named-entities.

Relational similarity measures have been used to classify the relationships between the head and the modifier in noun-compounds [28, 18, 8]. For example, in the compound *viral flu*, the *flu* (head) is *caused by* a *virus* (modifier). The *Diverse* dataset of Barker and Szpakowicz [1], which consists of 600 head-modifier pairs (noun-noun, adjective-noun and adverb-noun) is used as a benchmark dataset to evaluate relation classification of noun-compounds. Each noun-modifier pair in this dataset is annotated with one of the following five relations: *causal*, *temporal*, *spatial*, *participant*, and *quality*. Nakov and Hearst [18] proposed a linguistically motivated method that utilizes verbs, prepositions, and coordinate conjunctions that can help make explicit the hidden relations between the target nouns. They report a classification accuracy of 40.5% on the Diverse dataset using a single nearest neighbor classifier.

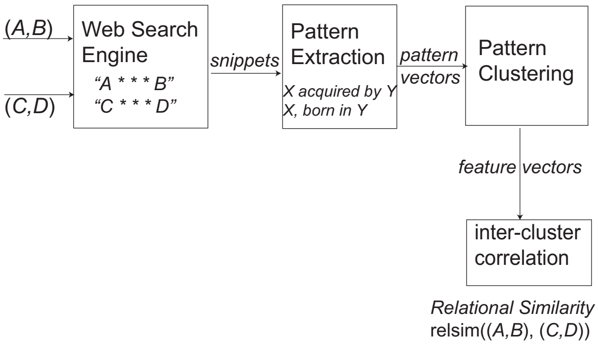

Figure 1: Outline of the proposed relational similarity method.

> Google to acquire YouTube for $1.65 billion in stock. Combination will create new opportunities for users and content owners everywhere...

Figure 2: A snippet returned for the query *"Google * * * YouTube"*.

3. METHOD

3.1 Outline

The proposed method consists of the four components outlined in Figure 1: a *Web search component*, a *pattern extraction component*, a *pattern clustering component*, and a *similarity computation component*. In this section, we give a brief overview of each of those components. The subsequent sections will explain the components in detail.

Let us assume that we are given two pairs of words, *(A,B)* and *(C,D)*, between which we must compute relational similarity. As shown in Figure 1, we first query a Web search engine using various queries to find the contexts in which the two words in each word-pair co-occur. We then download snippets returned by the search engine. This process is described in section 3.2.

Next, we employ a shallow lexical pattern extraction algorithm to extract lexical patterns that express semantic relations between the two words in a word-pair. For example, for the word-pair *(Google, YouTube)*, the proposed pattern extraction algorithm extracts the pattern ***X** acquires **Y***, where **X** and **Y** are placeholders respectively for the first and the second word in the word-pair. The proposed pattern extraction algorithm is further explained in section 3.3.

Using a dataset of entity-pairs, we extract a large number of lexical patterns that express various semantic relations. However, there can be more than one pattern that express the same semantic relation. We cluster the patterns to identify the ones that express a particular semantic relation. For this purpose, we present a sequential pattern clustering algorithm in section 3.4.

We represent a pair of words by a feature vector. Elements of a feature vector correspond to the total frequency of a word-pair in a pattern cluster. In section 3.5, we compute the relational similarity between two word-pairs using their feature vectors. We use inter-cluster correlation to account for the dependence between semantic relations.

3.2 Retrieving Contexts

The first step involved in computing the relational similarity between two word-pairs is to identify the relation (or relations) that hold between the two words in each word-pair. The context in which two words co-occur provides useful clues about the semantic relations that hold between those words. We propose the use of text snippets returned by a Web search engine as an approximation of the context of two words. Snippets (also known as *dynamic teasers*) are brief summaries provided by most of the Web search engines along with the search results. Typically, a snippet is created by a search engine by selecting a window of text including the queried words from a document they occur. Snippets are useful in search because most of the time a user can read the snippet and decide whether a particular search result is relevant, without even opening the url. Using snippets as contexts is also computationally efficient because it obviates the need to download the source documents from the Web, which can be time consuming if a document is large.

A snippet for a query containing two words, captures the local context in which they co-occur. For example, consider the snippet shown in Figure 2, returned by *Yahoo*[2] for the query *"Google * * YouTube"*. Here, the wildcard operator "*" matches one word or none in a document. This snippet is extracted from a newspaper article about the acquisition of YouTube by Google.

To retrieve snippets for a word pair *(A,B)*, we use the following seven types of queries: *"A * B"*, *"B * A"*, *"A * * B"*, *"B * * A"*, *"A * * * B"*, *"B * * * A"*, and *A B*. The queries containing the wildcard operator "*" return snippets in which the two words, *A* and *B* appear within a window of specified length. We call such queries *wildcard* queries. We search for snippets in which the query words co-occur within a maximum of three words (tokens). Moreover, the quotation marks around a query will ensure that the two words appear in the specified order (e.g. *A* before *B* in snippets retrieved for the query *"A * B"*). The usage of wildcards and quotations enables us to mimic the behavior of the *NEAR* operator [3]. In case all wildcard queries fail to return any snippets, we use the query *A B* (without wildcards or quotations) to retrieve snippets.

Once we collect snippets for a word-pair using each of the queries described above, we remove duplicates. We consider two snippets to be duplicates if they contain the exact sequence of all words. Duplicate snippets exist because of two main reasons. First, a web page can be mirrored in more than one location and the de-duplication mechanism of the search engine might fail to filter-out the duplicates (or might not perform de-duplication at all for mirror sites). Second, the queries we construct for a word-pair are not independent. For example, a query with two wildcards might return a snippet that can also be retrieved using a query with one wildcard. However, we observed that the ranking of search results vary with the number of wildcards used. A search engine usually returns only the top ranking results (in the case of Yahoo, only the top 1000 snippets can be downloaded). We use multiple queries per word-pair that induce different rankings, and aggregate search results to circumvent this limitation.

3.3 Extracting Lexical Patterns

Lexical syntactic patterns have been successfully used in various natural language processing tasks such as extracting hypernyms [12, 25], or meronyms [2], question answering [21], and paraphrase extraction [3]. Following these previous work, we present a shallow lexical pattern extraction algorithm to extract the semantic relations between two words from web snippets. The proposed method does not require any language dependent preprocessing such as, part-of-speech tagging or dependency parsing, which can be both time consuming at Web scale, and likely to produce incorrect results due to the fragmented and ill-formed snippets.

The pattern extraction algorithm consists of the following three steps.

Step 1: Given a context S, retrieved for a word-pair (A, B) according to the procedure described in section 3.2, we replace the two words A and B respectively with two variables X and Y. We consider variants of the two words and replace them also with the same variables. For example, we consider the variant *Google Inc.* as an occurrence of the word *Google*. Legal abbreviations such as *Inc., Ltd., Coop.*, and titles such as *Mr., Ms., Prof., Dr., Rev.* are considered as variants of the query terms. We replace all numeric values by D, a marker for digits. However, we do not remove punctuation marks.

Step 2: We generate all subsequences of the context S, that satisfy all the following conditions.

(i). A subsequence must contain exactly one occurrence of each X and Y (i.e. exactly one X and one Y must exist in a subsequence).

(ii). The maximum length of a subsequence is L words.

(iii). A subsequence is allowed to have gaps. However, we do not allow gaps exceeding g words. Moreover, the total length of all gaps in a subsequence should not exceed G words.

(iv). We expand all negation contractions in a context. For example, *didn't* is expanded to *did not*. We do not skip the word *not* when generating subsequences. For example, this condition ensures that from the snippet *X is not a Y*, we do not produce the subsequence *X is a Y*.

Step 3: We count the frequency of all generated subsequences for all word-pairs in the dataset. We select subsequences with frequency greater than N as lexical patterns to represent the semantic relations between words.

Our pattern extraction algorithm has four parameters (ca. *L, g, G* and *N*). We set the values of those parameters experimentally, as explained later in section 4. It is noteworthy that the proposed pattern extraction algorithm considers all the words in a snippet, and is *not* limited to extracting patterns from only the mid-fix (i.e. portion of text in a snippet that appears between the queried words). Moreover, the consideration of gaps enables us to capture relations between distant words in a snippet. We use a modified version of the prefixspan algorithm [20] to to generate subsequences. The conditions in Step 2 are used to prune the search space, thereby reducing the number of generated subsequences. For example, some of the patterns extracted form the snippet shown in Figure 2 are: ***X** to acquire **Y***, ***X** acquire **Y***, and ***X** to acquire **Y** for*.

3.4 Identifying Semantic Relations

A semantic relation can be expressed using more than one pattern. For example, consider the two distinct patterns, ***X** acquired **Y***, and ***X** completed the acquisition of **Y***. Both these patterns indicate that there exist an acquisition relation between **X** and **Y**. When we compute the relational similarity between two word-pairs, it is important to know whether there is any correspondence between the sets of patterns extracted for each word-pair. If there are many related patterns between two word-pairs, we can expect a high relational similarity.

We use distributional hypothesis [11] to find semantically related lexical patterns. Distributional hypothesis claims that words that

[2] http://developer.yahoo.com/search/boss/

[3] YahooBOSS API that we used in our experiments did not have a NEAR operator

occur in the same context have similar meanings. Distributional hypothesis has been used in various related tasks, such as, identifying related words[13], discovering inference rules[14], and extracting paraphrases[3]. If two lexical patterns are similarly distributed over a set of word-pairs (i.e. occurs with the same set of word-pairs), then from the distributional hypothesis it follows that the two patterns must be similar. We can cluster lexical patterns using their distributions over word-pairs, to identify semantically related patterns.

We represent a pattern by a vector of word-pair frequencies. We call this vector the *word-pair frequency* vector of the pattern. It is analogous to the *document frequency* vector of a word, as used in information retrieval. We denote the word-pair frequency vector of a pattern p by $\mathbf{p}$. The value of the element corresponding to a word-pair (a_i, b_i) in a word-pair frequency vector $\mathbf{p}$ of a pattern p, is the frequency, $f(a_i, b_i, p)$, that pattern p occurs with the word-pair (a_i, b_i).

As shown later in the experiments, the pattern extraction algorithm proposed in section 3.3 extracts a large number of lexical patterns (ca. over $400,000$). Clustering algorithms based on pair-wise comparisons between all patterns, are not feasible when the number of patterns is large. Consequently, we propose a sequential clustering algorithm (Algorithm 1) to efficiently cluster the extracted patterns.

Given a set of patterns, P and a clustering similarity threshold, θ, Algorithm 1 returns clusters (of patterns) that express similar semantic relations. First, in Algorithm 1, the function $SORT$, sorts the patterns in the descending order of their total occurrence in all word-pairs. The total occurrence of a pattern p is the sum of frequencies over all word-pairs (i.e. $\sum_i f(a_i, b_i, p)$). Once sorted by the frequency, the most common patterns will appear at the beginning in P, whereas rare patterns (i.e. patterns that occur only with few word-pairs) will get shifted to the end. Next, in line 2, we initialize the set of clusters, C, to empty set. The outer for-loop (starting at line 3), repeatedly takes a pattern $\mathbf{p_i}$ from the ordered set P, and in the inner for-loop (starting at line 6), finds the cluster, c^* ($\in C$) that is most similar to $\mathbf{p_i}$. To compute the similarity between a pattern and a cluster, first we represent a cluster by the vector sum of all word-pair frequency vectors corresponding to the patterns that belong to that cluster. Next, we compute the cosine of the angle between the vector that represents the cluster ($\mathbf{c_j}$), and the word-pair frequency vector of the pattern ($\mathbf{p_i}$). For two n dimensional vectors $\mathbf{x} = [x_1, \ldots, x_n]$, and $\mathbf{y} = [y_1, \ldots, y_n]$, their cosine similarity, $\text{cosine}(\mathbf{x}, \mathbf{y})$, is defined as follows,

$$\text{cosine}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{||\mathbf{x}|| ||\mathbf{y}||}. \quad (1)$$

Here, $||\mathbf{x}||$ denotes the L_2 norm of a vector $\mathbf{x}$ (i.e. $\sqrt[2]{\sum_{i=1}^{n} x_i^2}$) . If the similarity between a pattern $\mathbf{p_i}$, and the most similar cluster to it, $\mathbf{c}^*$, is greater than the value of the threshold θ, we append $\mathbf{p_i}$ to $\mathbf{c}^*$ (line 14). We use the operator $\oplus$ to denote the vector addition between $\mathbf{c}^*$ and $\mathbf{p_i}$. If $\mathbf{p_i}$ is not similar to any of the existing clusters beyond the threshold θ, then we form a new cluster $\{\mathbf{p_i}\}$ and append it to the set of clusters, C.

The only parameter in Algorithm 1 is the similarity threshold, θ, which ranges in $[0, 1]$. It decides the *purity* of the formed clusters. Setting θ to a high value ensures that the patterns in each cluster are highly similar. However, high θ values also result in a large number of clusters (increased model complexity). In section 4, we experimentally investigate the effect of θ on the overall performance of the proposed relational similarity measure.

The computational time complexity of Algorithm 1 is $O(n|C|)$, where n is the number of patterns to be clustered and $|C|$ is the number of clusters. The number of patterns n is usually very large compared to the number of clusters (i.e. $n >> |C|$). Consequently, the overall time complexity of Algorithm 1 linearly scales with the number of patterns. The sequential nature of the algorithm essentially avoids pair-wise comparisons between all patterns. Moreover, sorting the patterns by their total word-pair frequency prior to clustering, ensures that the final set of clusters contains the most common relations in the dataset.

Algorithm 1 Sequential pattern clustering algorithm.

```
Require: patterns $P = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$, threshold $\theta$
Ensure: clusters $C$
 1: SORT($P$)
 2: $C \leftarrow \{\}$
 3: for pattern $\mathbf{p_i} \in P$ do
 4:   $max \leftarrow -\infty$
 5:   $\mathbf{c}^* \leftarrow null$
 6:   for cluster $\mathbf{c_j} \in C$ do
 7:     $sim \leftarrow \text{cosine}(\mathbf{p_i}, \mathbf{c_j})$
 8:     if $sim > max$ then
 9:       $max \leftarrow sim$
10:       $\mathbf{c}^* \leftarrow \mathbf{c_j}$
11:     end if
12:   end for
13:   if $max \geq \theta$ then
14:     $\mathbf{c}^* \leftarrow \mathbf{c}^* \oplus \mathbf{p_i}$
15:   else
16:     $C \leftarrow C \cup \{\mathbf{p_i}\}$
17:   end if
18: end for
19: return $C$
```

3.5 Measuring Relational Similarity

Evidence from psychological experiments suggest that similarity can be context-dependent and even asymmetric [31, 16]. Human subjects have reportedly assigned different similarity ratings to word-pairs when the two words were presented in the reverse order. However, experimental results investigating the effects of asymmetry reports that the average difference in ratings for a word pair is less than 5 percent [16]. Consequently, in this paper we assume relational similarity to be symmetric and limit ourselves to symmetric similarity measures. This assumption is in line with previous work on relational similarity described in section 2.

To define a relational similarity measure over word-pairs, we first represent a word-pair by a feature vector. For explanation purposes, let us denote the feature vector of a word-pair (a, b) by $\mathbf{x}_{ab}$. Elements of the feature vector $\mathbf{x}_{ab}$, are the total frequencies of the word-pair (a, b) in each cluster. For example, the i^{th} element of the feature vector $\mathbf{x}_{ab}$ is given by,

$$\sum_{p \in c_i} f(a, b, p).$$

Here, p is a pattern in the cluster c_i, and $f(a, b, p)$ is the number of times that the word-pair (a, b) appears with the pattern p. We L_2 normalize all feature vectors.

We define the relational similarity, $\text{relsim}((a, b), (c, d))$, between two word-pairs (a, b) and (c, d) as follows,

$$\text{relsim}((a, b), (c, d)) = \mathbf{x}_{ab}^t \Lambda \mathbf{x}_{cd}. \quad (2)$$

Here, $\mathbf{x}_{ab}$ and $\mathbf{x}_{cd}$ respectively are the feature vectors for word-pairs (a, b) and (c, d), and Λ is a correlation matrix. Each cluster represents a set of semantically related patterns. However, in reality, semantic relations are not always mutually independent. The matrix Λ encodes this information about relational dependence. Specifically, the correlation between the relations represented by clusters c_i and c_j is given by the (i, j) element, λ_{ij}, in matrix Λ. It is a $|C| \times |C|$ square matrix, where $|C|$ is the number of clusters. Moreover, for the relational similarity computed using Formula 2 to be symmetric, the correlation matrix Λ must be a symmetric matrix.

It is noteworthy that the number of free parameters in Λ grows quadratically with the number of clusters. Given enough training data, we might be able to use a distance metric learning algorithm [23, 35] to compute Λ. However, in this paper we do not assume the availability of a large training dataset, and approximate Λ using inter-cluster correlation [24, 7]. Specifically, an element λ_{ij} is approximated by the correlation between the clusters c_i and c_j, which is computed as follows,

$$\lambda_{ij} = \frac{2}{|\Gamma|(|\Gamma|-1)} \sum_{(p \neq q),\ p,q \in \Gamma} \mathrm{cosine}(\mathbf{p}, \mathbf{q}). \quad (3)$$

Here, Γ is the merger between the two clusters c_i and c_j (i.e. union of the two clusters), and p and q are two different patterns in Γ. We denote the total number of patterns in Γ by $|\Gamma|$. Because a pattern appears only in one cluster, $|\Gamma| = |c_i| + |c_j|$ holds. Cosine similarity between two vectors is given by Formula 1.

The relational similarity measure defined in Formula 2 corresponds to Mahalanobis distance of the feature vector space. As a special case, if we set Λ to identity matrix we get the Euclidean distance between two points.

4. EXPERIMENTS

4.1 Dataset

We created a dataset [4] of word-pairs to evaluate the proposed relational similarity measure. Our dataset contains 100 instances (word or named-entity pairs) covering the following five relation types.

ACQUIRER-ACQUIREE This relation holds between pairs of company names (A, B), where the company B (acquiree) is acquired by the company A (acquirer). We only consider acquisitions that has already completed.

PERSON-BIRTHPLACE This relation holds between pairs (A, B), where A is the name of a person, and B is the location (place) where A was born. We consider city names and countries as locations.

CEO-COMPANY This relation holds between pairs (A, B), where A is the chief executive officer (CEO) of a company B. We consider both current as well as past CEOs of companies.

COMPANY-HEADQUARTERS This relation holds between pairs (A, B), where company A's headquarters is located in a place B. We select names of cities as B.

PERSON-FIELD This relation holds between pairs (A, B), where a person A is an expert or known for his or her abilities in a field B. Instances of this relation contain scientists and their field of expertise, sportsmen and the sports they are associated with, and artists and the genre that they perform.

Table 2: Distribution of patterns

Frequency (f)	Number of patterns	Percentage
$f \geq 1$	473910	100%
$f \geq 2$	**148655**	**31.36%**
$f \geq 3$	70952	14.97%
$f \geq 4$	46040	9.71%
$f \geq 5$	32596	6.87%
$f \geq 6$	25189	5.31%
$f \geq 7$	20220	4.26%
$f \geq 8$	16807	3.54%
$f \geq 9$	14393	3.03%
$f \geq 10$	12559	2.65%

We selected the above mentioned relation types because previous work on relation detection on the Web have frequently used those relations in evaluations [4]. We manually selected 20 instances for each of the five relation types. Instances were selected from various information sources such as Wikipedia[5], online newspapers, and company reviews[6]. For each instance in the dataset, using the YahooBOSS API[7], we download snippets as described in section 3.2. For each relation type, in Table 1, we show some of the instances and the total number of contexts.

4.2 Lexical Patterns

We run the pattern extraction algorithm described in section 3.3 on the contexts in our dataset to extract lexical patterns. Experimentally, we set the values for the various parameters in the pattern extraction algorithm: $L = 5$, $g = 2$, and $G = 4$. The total number of unique patterns extracted by the algorithm is 473910. However, as shown in Table 2, only 31.36% of those patterns occur more than twice. Patterns that only occur once contain misspellings or badly formatted texts. To filter-out this noise, we only select the 148655 patterns that occur at least twice. The remainder of the experiments in this paper are carried out using those patterns.

4.3 Pattern Clusters

We used the clustering algorithm described in section 3.4 to cluster the extracted patterns. As described in section 3.4, the clustering similarity threshold θ in Algorithm 1 determines the total number of clusters. In Figure 3, we show the ratio of the number of singletons (i.e. clusters with only one pattern) to total number of clusters against the value of threshold θ. It can be seen from Figure 3 that low θ values (e.g. $\theta < 0.35$) do not produce any singletons. However, when θ is increased beyond a certain point, the number of singletons steadily increases. In Algorithm 1, a pattern will be added to an existing cluster only if it has a similarity value greater than the threshold θ. Otherwise, a new cluster (a singleton) will be created with that pattern. Figure 3 confirms this behavior empirically. Some patterns appear only for a particular word pair (or pairs). Consequently, the similarity between those patterns is one. Because of that reason in Figure 3, the ratio does not reach one, when the clustering threshold is increased.

4.4 Relation Classification

We evaluate the proposed relational similarity measure in a relation classification task. Given a word-pair, the goal is to select a relation out of the five relation types in the dataset that describes the

[4] http://www.miv.t.u-tokyo.ac.jp/danushka/reldata.zip

[5] http://wikipedia.org/

[6] http://www.forbes.com/

[7] http://developer.yahoo.com/search/boss/

Table 1: Overview of the relational similarity dataset

Relation Type	Total contexts	Examples (20 in total for each relation type)
ACQUIRER-ACQUIREE	91439	(Google, YouTube), (Adobe Systems, Macromedia), (Yahoo, Inktomi)
PERSON-BIRTHPLACE	72836	(Franz Kafka, Prague), (Charlie Chaplin, London), (Marie Antoinette, Vienna)
CEO-COMPANY,	82682	(Terry Semel, Yahoo), (Eric Scmidt, Google), (Steve Jobs, Apple)
COMPANY-HEADQUARTERS	100887	(Microsoft, Redmond), (Yahoo, Sunnyvale), (Google, Mountain View)
PERSON-FIELD	99660	(Albert Einstein, Physics), (Roger Federer, Tennis), (Shane Warne, Cricket)

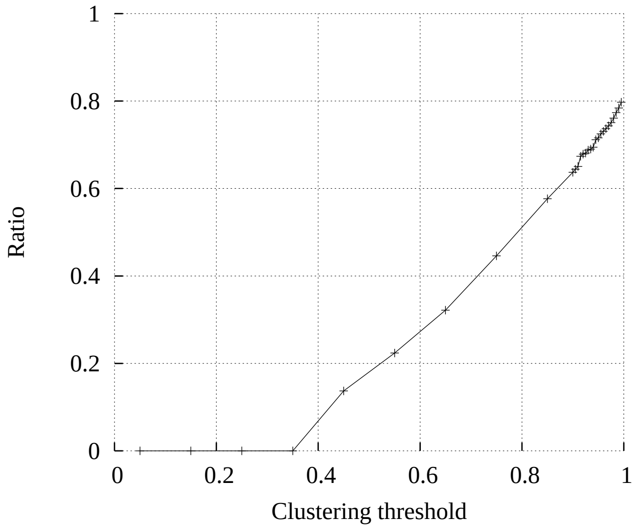

Figure 3: Ratio of singletons to total clusters against the clustering threshold (θ).

relationship between the two words. This is a multi-class classification problem. We use k-nearest neighbor classification to assign a relation to a given word-pair. Specifically, given a word-pair (a, b), for which a relation R holds, we compute the relational similarity between (a, b) and the remaining 99 word-pairs in the dataset. We then sort the word-pairs in the descending order of relational similarity with (a, b), and select the most similar k word-pairs. We then find the relation that is given in the dataset to most number of those k word-pairs and assign this relation to (a, b). If there are more than one majority relation among the top k word-pairs, then we randomly select a relation from those majority relations. We repeat this procedure with each of the word-pairs in the dataset, and compute the classification accuracy as follows,

$$\text{Accuracy} = \frac{\text{No. of correctly classified pairs}}{\text{Total no. of pairs}}. \quad (4)$$

We compute the classification accuracy for each of the five relation types in the dataset individually. Because each relation type has 20 word-pairs in the dataset, the total number of pairs is in Formula 4 is set to 20. Moreover, the classification accuracy for the entire dataset (100 word-pairs) is computed by counting the correctly classified word-pairs for all five relation types.

A good relational similarity measure must assign higher similarity scores to word-pairs with similar implicit relations. However, classification accuracy does not evaluate the relative rankings of similarity scores. We use average precision [22] to evaluate the top most similar k word-pairs to a given word-pair. Average precision integrates the precision at different ranks and is frequently used as an evaluation measure in ranking tasks. Average precision for a particular relation type R is defined as follows,

$$\text{AveragePrecision} = \frac{\sum_{r=1}^{k} \text{Pre}(r) \times \text{Rel(r)}}{\text{No of relevant word-pairs}}. \quad (5)$$

Here, $\text{Rel}(r)$ is a binary valued function that returns 1 if the word-pair at rank r has the same relation (i.e. R) as in (a, b). Otherwise it returns zero. $\text{Pre}(r)$ is the precision at rank r, and is given by,

$$\text{Pre}(r) = \frac{\text{no. of word-pairs with relation R in top r pairs}}{r}. \quad (6)$$

The number of relevant word-pairs is 20 for all five relation types in our dataset.

We consider the 10 most similar word-pairs (i.e. $k = 10$) for nearest neighbor classification. Average precision is computed for those top 10 word-pairs. Figure 4 shows the performance (classification accuracy and average precision) against the clustering threshold θ. From Figure 4, we can see that low θ values result in poor performance. Performance increases when we increase the value of θ. This behavior can be understood from the fact that higher values of θ result in highly similar pattern clusters that represent specific semantic relations. Best performance is reported for $\theta = 0.955$. At this value of θ we obtain 2629 non-singleton clusters (i.e. containing more than one pattern), and 6930 singletons (i.e. containing only one pattern).

In Table 3, we show the top 10 clusters with the largest number of lexical patterns. The number of patterns in each cluster is shown within brackets in the first column. For each cluster in Table 3, we show the top four patterns that occur in most number of word-pairs. For explanation purposes, we label the clusters with the five relation types as: clusters 1 and 4 (acquirer-acquiree); clusters 2, 3, 6 and 7 (person-field); cluster 5 (ceo-company); cluster 8 and 10 (company-headquarters); cluster 9 (person-birthplace). From Table 3, it is clear that patterns representing various semantic relations are extracted by the proposed pattern extraction algorithm. Moreover, we see that each cluster contains different lexical patterns that express a specific semantic relation. We can also see that there are multiple clusters even among the top few clusters shown in Table 3 that represent a particular relation type. For example, cluster 1 and 4 both represent *acquirer-acquiree* relation, although the patterns in cluster 1 are derived from the verb *acquire*, whereas the patterns in cluster 4 are derived from the verbs *buy* and *purchase*. We can expect a certain level of correlation between such clusters, which justifies the relational similarity measure defined in Formula 2.

We compare the proposed relational similarity measure (**CORR**) against cluster inner-product baseline (**IP**), vector space model-based relational similarity [30] (**VSM**), and the state-of -the-art Latent Relational Analysis [28] (**LRA**). Next, we explain each of those relational similarity measures in detail.

VSM: This is the vector space model-based approach proposed by Turney et al. [30]. First, each word-pair is represented by a

Table 3: Most frequent patterns in the largest clusters

cluster 1 (2868)	**X** acquires **Y**	**X** has acquired **Y**	**X**'s **Y** acquisition	**X** compra **Y** ,	**Y** team to **X**
cluster 2 (2711)	**Y** legend **X** was	brazilian **Y** legend **X**	**Y** legend **X** was held	**X** 's championship **Y**	**Y** star **X** robbed
cluster 3 (2615)	**Y** champion **X**	world **Y** champion **X**	**X** teaches **Y**	**X**'s greatest **Y**	**Y** players like **X**
cluster 4 (2008)	**X** to buy **Y**	**X** and **Y** confirmed	**X** buy **Y** is	**Y** purchase to boost **X**	**Y** grab shows **X**
cluster 5 (2002)	**Y** founder **X**	**Y** founder and ceo **X**	**X**, founder of **Y**	**X** says **Y** connect	**X** says **Y** connect is
cluster 6 (1364)	**X** revolutionized **Y**	**X** professor of **Y**	in **Y** since **X**	ago, **X** revolutionized **Y**	**X**'s contribution to **Y**
cluster 7 (845)	**X** and modern **Y**	genius: **X** and modern **Y**	**Y** in DDDD, **X** was	on **Y** by **X**	**X**'s lectures on **Y**
cluster 8 (280)	**X** headquarters in **Y** .	**X** offices in **Y**	past **X** offices in **Y**	the **X** conference in **Y**	**X** headquarters in **Y** on
cluster 9 (144)	**X**'s childhood in **Y**	**X**'s birth in **Y**	**Y** born **X**	**Y** born **X** introduced the	sobbing **X** left **Y** to
cluster 10 (49)	**X** headquarters in **Y**	**X**'s **Y** headquarters	**Y** - based **X**	**X** works with the **Y**	**Y** office of **X**

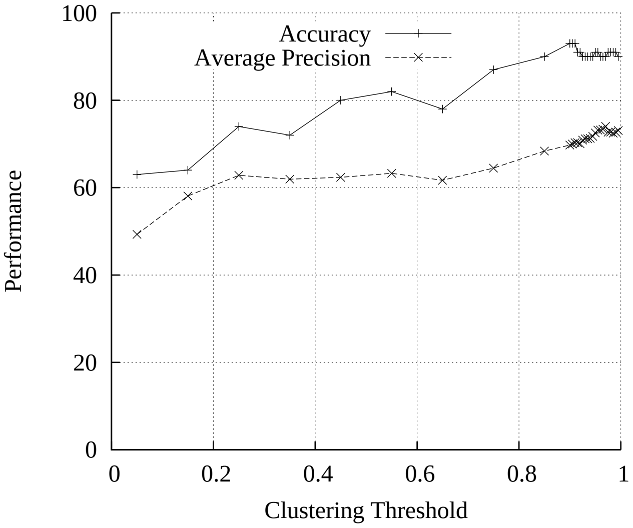

Figure 4: Performance of the proposed method against the clustering threshold (θ).

vector of pattern frequencies. Then the relational similarity between two word-pairs is computed as the cosine of the angle between the two vectors representing the two word-pairs. This approach is equivalent to computing relational similarity using Formula 2, if we define feature vectors as pattern frequency vectors, and set the correlation matrix Λ to identity matrix.

LRA: This is the Latent Relational Analysis (LRA) proposed by Turney [28]. First, a matrix is created, in which the rows represent word-pairs and the columns represent lexical patterns. An element of the matrix corresponds to the frequency of occurrence of a word-pair in a particular lexical pattern. Next, singular value decomposition (SVD) is performed on this matrix to reduce the number of columns. Finally, the relational similarity between two word-pairs is computed as the cosine of the angle between the corresponding row vectors. We re-implemented LRA as described in the original paper. However, we do not use related word thesauri to find additional word-pairs, because such resources are not available for named-entities. Following, Turney's proposal, we used the most frequent 4000 lexical patterns in the matrix and reduced the number of columns to 300 via SVD (i.e. eigenvectors corresponding to the largest 300 eigenvalues are used to approximate the matrix). We used scientific python's SVD library[8] for the computation of SVD. LRA is the current state-of-the-art relational similarity measure.

IP: We set Λ in Formula 2 to the identity matrix and compute relation similarity using pattern clusters. This is equivalent to computing relational similarity between two word-pairs as the inner-product (IP) between the two feature vectors created using pattern clusters. This baseline is a cut-down version of the proposed method, where we assume all clusters are independent (i.e. correlation coefficients $\lambda_{ij} = 0, \forall_{i \neq j}$). **IP** baseline is expected to show the effect on the performance when the correlation between clusters is ignored.

CORR: This is the proposed relational similarity measure. It is defined in Formula 2. For both **IP** and **CORR**, we used the same set of clusters. Therefore, any difference in performance can be directly attributable to using cluster correlation when computing relational similarity. We used the 9559 clusters (including both the 2629 non-singleton clusters and the 6930 singleton clusters) derived by setting the clustering threshold θ to the value 0.955.

We compare the above mentioned four methods in Table 4 using average precision. The proposed method (**CORR**) reports the highest overall average precision (73.18) in Table 4. Moreover, **CORR** has the best average precision scores for four out of five relation types. Analysis of variance (ANOVA) reveals that the average precision scores in Table 4 are statistically significant. Moreover, paired t-tests conducted between the proposed method (**CORR**) and each of the remaining three methods in Table 4, reveal that the improvement shown by the proposed method over **VSM**, **IP**, and **LRA** is statistically significant ($\alpha = 0.01$).

Table 5 compares the four methods using classification accuracy. The proposed method (**CORR**) has the highest classification accuracy (93%), followed by **IP**, **LRA**, and **VSM**. It is interesting to note that the **IP** baseline that does not consider inter-cluster correlation performs better than **VSM** method. This result shows that clustering similar patterns before computing relational similarity indeed improves performance. Among the five relation types compared in Table 5, all four methods report a perfect (100%) classification accuracy for the three relation types: acquirer-acquiree, company-headquarters, and ceo-company. Lowest performance is reported for the person-birthplace relation. A closer look into the snippets extracted for the person-birthplace pairs revealed that there were many snippets that convey information related to places that people associate with other than the place of birth. For example, in the case of actors, the locations where they gave their first performance are incorrectly extracted as a contexts for the person-birthplace relation.

[8] www.scipy.org

Table 4: Average precision

Relation	VSM	LRA	IP	CORR
acquirer-acquiree	92.27	92.24	91.47	**93.78**
comp.-headquarters	84.55	82.54	79.86	**85.19**
person-field	44.70	43.96	51.95	**56.09**
ceo-comp.	95.82	**96.12**	90.58	95.31
person-birthplace	27.47	27.95	33.43	**35.52**
overall	68.96	68.56	69.46	**73.18**

Table 5: Relation classification accuracy

Relation	VSM	LRA	IP	CORR
acquirer-acquiree	100	100	100	100
comp.-headquarters	100	100	100	100
person-field	80	80	95	95
ceo-comp.	100	100	100	100
person-birthplace	50	60	55	70
overall	86	88	90	93

5. DISCUSSION

The proposed method requires only a few queries to compute the relational similarity between two word-pairs. For each pair of words it uses the seven types of queries described in section 3.2, to retrieve snippets, and then matches a large number of lexical patterns within those snippets. Moreover, multiple snippets (e.g. 50 or 100) can be retrieved using only one query. Contrastingly, previously proposed relational similarity measures such as LRA [28] require a query per each lexical pattern. For example, LRA, which uses approximately 4000 lexical patterns, requires at least 8000 (2 word-pairs× 4000 patterns) search engine queries. Moreover, the number of search engine queries required by LRA increases with the number of lexical patterns. This is problematic not only because it increases the load on search engines, but also makes it impossible to use a large number of lexical patterns to express various semantic relations. The low number of search engine queries required by the proposed method makes it attractive for measuring relational similarity on the Web.

LRA is based on singular value decomposition (SVD), which is computationally expensive. Performing SVD on large matrices can be time consuming. In fact, it has been shown [28] that LRA takes over 9 days to process a dataset of 374 word-analogy questions. Comparatively, the proposed method requires under 6 hours in total (retrieve contexts, extract patterns, cluster, and compute similarity) to process 100 word-pairs. Moreover, in LRA, to compute the similarity for new (unseen) word-pairs, their pattern frequency vectors must be first appended to the matrix and then the SVD process must be repeated. On the other hand, in the proposed method we can simply use the existing clusters to create feature vectors for new word-pairs. Therefore, the proposed method is suitable in an online setting (e.g. web search), where we must quickly compute relational similarity for unseen word-pairs.

The definition of relational similarity as given in Formula 2 can be viewed as a general framework into which all of the existing relational similarity measures can be integrated. The existing approaches differ in their definition of matrix Λ. For example, in **VSM**, Λ is the identity matrix and in **LRA** it is computed via SVD. The task of designing relational similarity measures can be modeled as searching for a matrix Λ that best reflects the notion of relational similarity possessed by humans. If we consider the transformation, $\mathbf{y} = \mathbf{x_{ab}} - \mathbf{x_{cd}}$, then we can use Formula 2 to define a *relational distance*, $\mathbf{y}^t\Delta\mathbf{y}$, where Δ is a distance matrix. Given a training dataset, we can learn Δ using a distance metric learning technique [23, 35].

6. CONCLUSION

We proposed a method to compute the similarity between implicit semantic relations in two word-pairs. Given two word-pairs, the proposed relational similarity measure first finds contexts in which the two words in each word-pair co-occur on the Web. We use text snippets returned by a Web search engine as contexts, and proposed a shallow lexical pattern extraction algorithm to represent the various semantic relations that exist between two words. The proposed pattern extraction algorithm does not require language specific preprocessing techniques such as part-of-speech taggers or dependency parsers. We then cluster the extracted patterns to identify the different lexical patterns that convey a particular semantic relation. We proposed a sequential clustering algorithm that scales linearly with the number of patterns, to efficiently cluster a larger number of patterns. We create a feature vector using the formed pattern clusters, and compute the relational similarity between two word-pairs as the Mahalanobis similarity between the two feature vectors. We account for the dependence between semantic relations by using an inter-cluster correlation matrix. We compared the proposed method against a baseline relational similarity measure and previously proposed relational similarity measures, including the state-of-the-art latent relational analysis. Experimental results on a dataset that contains five common relation types show that the proposed method significantly outperforms the state-of-the-art relational similarity measure in a relation classification task. In future, we intend to employ the proposed relational similarity measure to retrieve a set of word-pairs for a given implicit relation from the Web.

7. REFERENCES

[1] K. Barker and S. Szpakowicz. Semi-automatic recognition of noun modifier relationships. In *Proc. of COLING'98*, pages 96–102, 1998.

[2] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proc. of ACL'99*, pages 57–64, 1999.

[3] R. Bhagat and D. Ravichandran. Large scale acquisition of paraphrases for learning surface patterns. In *Proc. of ACL'08: HLT*, pages 674–682, 2008.

[4] R. C. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proc. of ACL'07*, pages 576–583, 2007.

[5] P. Cimiano and J. Wenderoth. Automatic acquisition of ranked qualia structures from the web. In *Proc. of ACL'07*, pages 888–895, 2007.

[6] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proc. of ACL'04*, pages 423–429, 2004.

[7] Cutting, R. Douglas, D. R. Karger, and J. O. Pedersen. Constant interaction-time scatter/gather browsing of very large documents collections. In *Proceedings of SIGIR'93*, 1993.

[8] D. Davidov and A. Rappoport. Classification of semantic relationships between nominals using pattern clusters. In *Proc. of the ACL'08*, 2008.

[9] D. Davidov and A. Rappoport. Unsupervised discovery of generic relationships using pattern clusters and its evaluation

by automatically generated sat analogy questions. In *Proc. of ACL'08-HLT*, pages 692–700, 2008.

[10] B. Falkenhainer, K. Forbus, and D. Gentner. Structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

[11] Z. Harris. Distributional structure. *Word*, 10:146–162, 1954.

[12] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of 14th COLING*, pages 539–545, 1992.

[13] D. Lin. Automatic retrieval and clustering of similar words. In *Proc. of COLING-ACL'98*, pages 768–774, 1998.

[14] D. Lin and P. Pantel. Dirt: Discovery of inference rules from text. In *Proc. of ACM SIGKDD'01*, pages 323–328, 2001.

[15] Z. Marx, D. Ido, B. Joachim, and S. Eli. Coupled clustering: A method for detecting structural correspondance. *Journal of Machine Learning Research*, 3:747–780, 2002.

[16] D. Medin, R. Goldstone, and D. Gentner. Respects for similarity. *Psychological Review*, 6(1):1–28, 1991.

[17] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introducton to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:238–244, 1990.

[18] P. Nakov and M. Hearst. Solving relational similarity problems using the web as a corpus. In *Proc. of ACL'08-HLT*, pages 452–460, 2008.

[19] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proc. of AAAI'06*, pages 1400–1405, 2006.

[20] J. Pei, J. Han, B. Mortazavi-Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004.

[21] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proc. of ACL '02*, pages 41–47, 2001.

[22] G. Salton and C. Buckley. *Introduction to Modern Information Retreival*. McGraw-Hill Book Company, 1983.

[23] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Proc. of NIPS'03*, 2003.

[24] H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[25] R. Snow, D. Jurafsky, and A. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proc. of Advances in Neural Information Processing Systems (NIPS) 17*, pages 1297–1304, 2005.

[26] P. Turney. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141, 2005.

[27] P. Turney. Expressing implicit semantic relations without supervision. In *Proc. of Coling/ACL'06*, pages 313–320, 2006.

[28] P. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

[29] P. Turney and M. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60:251–278, 2005.

[30] P. Turney, M. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proc. of RANLP'03*, pages 482–486, 2003.

[31] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1997.

[32] T. Veale. The analogical thesaurus. In *Proc. of 15th Innovative Applications of Artificial Intelligence Conference (IAAI'03)*, pages 137–142, 2003.

[33] T. Veale. Wordnet sits the sat: A knowledge-based approach to lexical analogy. In *Proc. of ECAI'04*, pages 606–612, 2004.

[34] T. Veale and M. T. Keane. The competence of structure mapping on hard analogies. In *Proc. of IJCAI'03*, 2003.

[35] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proc. of NIPS'05*, pages 1473–1480, 2005.

[36] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.