A Web Search Engine-based Approach to Measure Semantic Similarity between Words

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka, *Member, IEEE*

Abstract—Measuring the semantic similarity between words is an important component in various tasks on the web such as relation extraction, community mining, document clustering, and automatic meta data extraction. Despite the usefulness of semantic similarity measures in these applications, accurately measuring semantic similarity between two words (or entities) remains a challenging task. We propose an empirical method to estimate semantic similarity using page counts and text snippets retrieved from a Web search engine for two words. Specifically, we define various word co-occurrence measures using page counts and integrate those with lexical patterns extracted from text snippets. To identify the numerous semantic relations that exist between two given words, we propose a novel pattern extraction algorithm and a pattern clustering algorithm. The optimal combination of page counts-based co-occurrence measures and lexical pattern clusters is learned using support vector machines. The proposed method outperforms various baselines and previously proposed web-based semantic similarity measures on three benchmark datasets showing a high correlation with human ratings. Moreover, the proposed method significantly improves the accuracy in a community mining task.

Index Terms—Web Mining, Information Extraction, Web Text Analysis

1 Introduction

Accurately measuring the semantic similarity between words is an important problem in web mining, information retrieval, and natural language processing. Web mining applications such as, community extraction, relation detection, and entity disambiguation, require the ability to accurately measure the semantic similarity between concepts or entities. In information retrieval, one of the main problems is to retrieve a set of documents that is semantically related to a given user query. Efficient estimation of semantic similarity between words is critical for various natural language processing tasks such as word sense disambiguation (WSD), textual entailment, and automatic text summarization.

Semantically related words of a particular word are listed in manually created general-purpose lexical ontologies such as WordNet[1], In WordNet, a *synset* contains a set of synonymous words for a particular sense of a word. However, semantic similarity between entities changes over time and across domains. For example, *apple* is frequently associated with *computers* on the Web. However, this sense of apple is not listed in most general-purpose thesauri or dictionaries. A user who searches for *apple* on the Web, might be interested in this sense of apple and not apple as a fruit. New words are constantly being created as well as new senses are assigned to existing words. Manually maintaining ontologies to capture these new words and senses is costly if not impossible.

• *The University of Tokyo, Japan.*
danushka@mi.ci.i.u-tokyo.ac.jp

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

We propose an automatic method to estimate the semantic similarity between words or entities using Web search engines. Because of the vastly numerous documents and the high growth rate of the Web, it is time consuming to analyze each document separately. Web search engines provide an efficient interface to this vast information. Page counts and snippets are two useful information sources provided by most Web search engines. Page count of a query is an estimate of the number of pages that contain the query words. In general, page count may not necessarily be equal to the word frequency because the queried word might appear many times on one page. Page count for the query *P AND Q* can be considered as a global measure of co-occurrence of words *P* and *Q*. For example, the page count of the query *"apple"* AND *"computer"* in *Google* is $288,000,000$, whereas the same for *"banana"* AND *"computer"* is only $3,590,000$. The more than 80 times more numerous page counts for *"apple"* AND *"computer"* indicate that *apple* is more semantically similar to *computer* than is *banana*.

Despite its simplicity, using page counts alone as a measure of co-occurrence of two words presents several drawbacks. First, page count analysis ignores the position of a word in a page. Therefore, even though two words appear in a page, they might not be actually related. Secondly, page count of a polysemous word (a word with multiple senses) might contain a combination of all its senses. For an example, page counts for *apple* contains page counts for *apple* as a fruit and *apple* as a company. Moreover, given the scale and noise on the Web, some words might co-occur on some pages without being actually related [1]. For those reasons, page counts alone are unreliable when measuring semantic similarity.

Snippets, a brief window of text extracted by a search engine around the query term in a document, provide

useful information regarding the local context of the query term. Semantic similarity measures defined over snippets, have been used in query expansion [2], personal name disambiguation [3], and community mining [4]. Processing snippets is also efficient because it obviates the trouble of downloading web pages, which might be time consuming depending on the size of the pages. However, a widely acknowledged drawback of using snippets is that, because of the huge scale of the web and the large number of documents in the result set, only those snippets for the top-ranking results for a query can be processed efficiently. Ranking of search results, hence snippets, is determined by a complex combination of various factors unique to the underlying search engine. Therefore, no guarantee exists that all the information we need to measure semantic similarity between a given pair of words is contained in the top-ranking snippets.

We propose a method that considers both page counts and lexical syntactic patterns extracted from snippets that we show experimentally to overcome the above mentioned problems. For example, let us consider the following snippet from Google for the query ***Jaguar*** *AND* ***cat***.

> *"The **Jaguar** is the largest **cat** in Western Hemisphere and can subdue larger prey than can the puma"*

Fig. 1. A snippet retrieved for the query ***Jaguar*** *AND* ***cat***.

Here, the phrase *is the largest* indicates a hypernymic relationship between Jaguar and cat. Phrases such as *also known as, is a, part of, is an example of* all indicate various semantic relations. Such indicative phrases have been applied to numerous tasks with good results, such as hypernym extraction [5] and fact extraction [6]. From the previous example, we form the pattern ***X*** *is the largest* ***Y***, where we replace the two words *Jaguar* and *cat* by two variables **X** and **Y**.

Our contributions are summarized as follows.

- We present an automatically extracted lexical syntactic patterns-based approach to compute the semantic similarity between words or entities using text snippets retrieved from a web search engine. We propose a lexical pattern extraction algorithm that considers word subsequences in text snippets. Moreover, the extracted set of patterns are clustered to identify the different patterns that describe the same semantic relation.
- We integrate different web-based similarity measures using a machine learning approach. We extract synonymous word-pairs from WordNet synsets as positive training instances and automatically generate negative training instances. We then train a two-class support vector machine to classify synonymous and non-synonymous word-pairs. The integrated measure outperforms all existing Web-based semantic similarity measures on a benchmark dataset.
- We apply the proposed semantic similarity measure to identify relations between entities, in particular people, in a community extraction task. In this experiment, the proposed method outperforms the baselines with statistically significant precision and recall values. The results of the community mining task show the ability of the proposed method to measure the semantic similarity between not only words, but also between named entities, for which manually created lexical ontologies do not exist or incomplete.

2 Related Work

Given a taxonomy of words, a straightforward method to calculate similarity between two words is to find the length of the shortest path connecting the two words in the taxonomy [7]. If a word is polysemous then multiple paths might exist between the two words. In such cases, only the shortest path between any two senses of the words is considered for calculating similarity. A problem that is frequently acknowledged with this approach is that it relies on the notion that all links in the taxonomy represent a uniform distance.

Resnik [8] proposed a similarity measure using information content. He defined the similarity between two concepts C_1 and C_2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C_1 and C_2. Then the similarity between two words is defined as the maximum of the similarity between any concepts that the words belong to. He used WordNet as the taxonomy; information content is calculated using the Brown corpus.

Li et al., [9] combined structural semantic information from a lexical taxonomy and information content from a corpus in a nonlinear model. They proposed a similarity measure that uses shortest path length, depth and local density in a taxonomy. Their experiments reported a high Pearson correlation coefficient of 0.8914 on the Miller and Charles [10] benchmark dataset. They did not evaluate their method in terms of similarities among named entities. Lin [11] defined the similarity between two concepts as the information that is in common to both concepts and the information contained in each individual concept.

Cilibrasi and Vitanyi [12] proposed a distance metric between words using only page-counts retrieved from a web search engine. The proposed metric is named *Normalized Google Distance* (NGD) and is given by,

$$NGD(P,Q) = \frac{\max\{\log H(P), \log H(Q)\} - \log H(P,Q)}{\log N - \min\{\log H(P), \log H(Q)\}}.$$

Here, P and Q are the two words between which distance $NGD(P,Q)$ is to be computed, $H(P)$ denotes the page-counts for the word P, and $H(P,Q)$ is the page-counts for the query *P AND Q*. NGD is based on normalized information distance [13], which is defined using

Kolmogorov complexity. Because NGD does not take into account the context in which the words co-occur, it suffers from the drawbacks described in the previous section that are characteristic to similarity measures that consider only page-counts.

Sahami et al., [2] measured semantic similarity between two queries using snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF-weighted term vector. Each vector is L_2 normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors. They did not compare their similarity measure with taxonomy-based similarity measures.

Chen et al., [4] proposed a double-checking model using text snippets returned by a Web search engine to compute semantic similarity between words. For two words P and Q, they collect snippets for each word from a Web search engine. Then they count the occurrences of word P in the snippets for word Q and the occurrences of word Q in the snippets for word P. These values are combined nonlinearly to compute the similarity between P and Q. The *Co-occurrence Double-Checking* (CODC) measure is defined as,

$$\mathrm{CODC}(P,Q) = \begin{cases} 0 & \text{if } f(P@Q) = 0, \\ \exp\left(\log\left[\frac{f(P@Q}{H(P)} \times \frac{f(Q@P)}{H(Q)}\right]^{\alpha}\right) & \text{otherwise.} \end{cases}$$

Here, $f(P@Q)$ denotes the number of occurrences of P in the top-ranking snippets for the query Q in Google, $H(P)$ is the page count for query P, and α is a constant in this model, which is experimentally set to the value 0.15. This method depends heavily on the search engine's ranking algorithm. Although two words P and Q might be very similar, we cannot assume that one can find Q in the snippets for P, or vice versa, because a search engine considers many other factors besides semantic similarity, such as publication date (novelty) and link structure (authority) when ranking the result set for a query. This observation is confirmed by the experimental results in their paper which reports zero similarity scores for many pairs of words in the Miller and Charles [10] benchmark dataset.

Semantic similarity measures have been used in various applications in natural language processing such as word-sense disambiguation [14], language modeling [15], synonym extraction [16], and automatic thesauri extraction [17]. Semantic similarity measures are important in many Web-related tasks. In query expansion [18] a user query is modified using synonymous words to improve the relevancy of the search. One method to find appropriate words to include in a query is to compare the previous user queries using semantic similarity measures. If there exist a previous query that is semantically related to the current query, then it can be either suggested to the user, or internally used by the search engine to modify the original query.

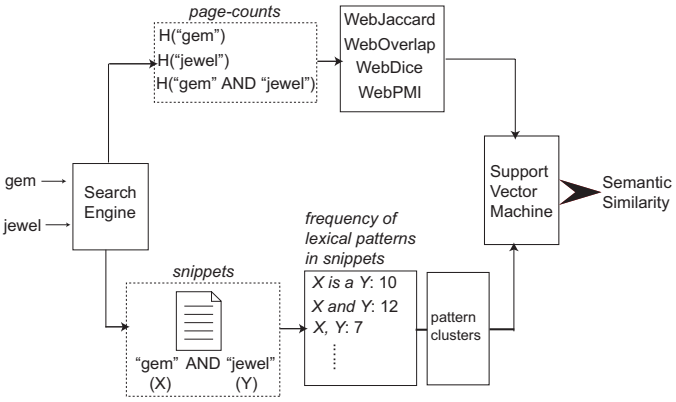

Fig. 2. Outline of the proposed method.

3 METHOD

3.1 Outline

Given two words P and Q, we model the problem of measuring the semantic similarity between P and Q, as a one of constructing a function $\mathrm{sim}(P,Q)$ that returns a value in range $[0,1]$. If P and Q are highly similar (e.g. synonyms), we expect $\mathrm{sim}(P,Q)$ to be closer to 1. On the other hand, if P and Q are not semantically similar, then we expect $\mathrm{sim}(P,Q)$ to be closer to 0. We define numerous features that express the similarity between P and Q using page counts and snippets retrieved from a web search engine for the two words. Using this feature representation of words, we train a two-class support vector machine (SVM) to classify synonymous and non-synonymous word pairs. The function $\mathrm{sim}(P,Q)$ is then approximated by the confidence score of the trained SVM.

Fig. 2 illustrates an example of using the proposed method to compute the semantic similarity between two words, *gem* and *jewel*. First, we query a Web search engine and retrieve page counts for the two words and for their conjunctive (i.e. *"gem"*, *"jewel"*, and *"gem AND jewel"*). In section 3.2, we define four similarity scores using page counts. Page counts-based similarity scores consider the global co-occurrences of two words on the Web. However, they do not consider the local context in which two words co-occur. On the other hand, snippets returned by a search engine represent the local context in which two words co-occur on the web. Consequently, we find the frequency of numerous lexical-syntactic patterns in snippets returned for the conjunctive query of the two words. The lexical patterns we utilize are extracted automatically using the method described in Section 3.3. However, it is noteworthy that a semantic relation can be expressed using more than one lexical pattern. Grouping the different lexical patterns that convey the same semantic relation, enables us to represent a semantic relation between two words accurately. For this purpose

we propose a sequential pattern clustering algorithm in Section 3.4. Both page counts-based similarity scores and lexical pattern clusters are used to define various features that represent the relation between two words. Using this feature representation of word pairs, we train a two-class support vector machine (SVM) [19] in Section 3.5.

3.2 Page-count-based Co-occurrence Measures

Page counts for the query P *AND* Q can be considered as an approximation of co-occurrence of two words (or multi-word phrases) P and Q on the Web. However, page counts for the query P *AND* Q alone do not accurately express semantic similarity. For example, Google returns $11,300,000$ as the page count for *"car" AND "automobile"*, whereas the same is $49,000,000$ for *"car" AND "apple"*. Although, *automobile* is more semantically similar to *car* than *apple* is, page counts for the query *"car" AND "apple"* are more than four times greater than those for the query *"car" AND "automobile"*. One must consider the page counts not just for the query P *AND* Q, but also for the individual words P and Q to assess semantic similarity between P and Q.

We compute four popular co-occurrence measures; Jaccard, Overlap (Simpson), Dice, and Pointwise mutual information (PMI), to compute semantic similarity using page counts. For the remainder of this paper we use the notation $H(P)$ to denote the page counts for the query P in a search engine. The *WebJaccard* coefficient between words (or multi-word phrases) P and Q, $\text{WebJaccard}(P,Q)$, is defined as,

$$\text{WebJaccard}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c, \\ \frac{H(P\cap Q)}{H(P)+H(Q)-H(P\cap Q)} & \text{otherwise.} \end{cases} \quad (1)$$

Therein, $P \cap Q$ denotes the conjunction query P *AND* Q. Given the scale and noise in Web data, it is possible that two words may appear on some pages even though they are not related. In order to reduce the adverse effects attributable to such co-occurrences, we set the WebJaccard coefficient to zero if the page count for the query $P \cap Q$ is less than a threshold c[2]. Similarly, we define *WebOverlap*, $\text{WebOverlap}(P,Q)$, as,

$$\text{WebOverlap}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c, \\ \frac{H(P\cap Q)}{\min(H(P),H(Q))} & \text{otherwise.} \end{cases} \quad (2)$$

WebOverlap is a natural modification to the Overlap (Simpson) coefficient. We define the *WebDice* coefficient as a variant of the Dice coefficient. $\text{WebDice}(P,Q)$ is defined as,

$$\text{WebDice}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c, \\ \frac{2H(P\cap Q)}{H(P)+H(Q)} & \text{otherwise.} \end{cases} \quad (3)$$

2. we set $c = 5$ in our experiments

Pointwise mutual information (PMI) [20] is a measure that is motivated by information theory; it is intended to reflect the dependence between two probabilistic events. We define *WebPMI* as a variant form of pointwise mutual information using page counts as,

$$\text{WebPMI}(P,Q) = \begin{cases} 0 & \text{if } H(P\cap Q) \leq c, \\ \log_2\left(\frac{\frac{H(P\cap Q)}{N}}{\frac{H(P)}{N}\frac{H(Q)}{N}}\right) & \text{otherwise.} \end{cases} \quad (4)$$

Here, N is the number of documents indexed by the search engine. Probabilities in (4) are estimated according to the maximum likelihood principle. To calculate PMI accurately using (4), we must know N, the number of documents indexed by the search engine. Although estimating the number of documents indexed by a search engine [21] is an interesting task itself, it is beyond the scope of this work. In the present work, we set $N = 10^{10}$ according to the number of indexed pages reported by Google. As previously discussed, page counts are mere approximations to actual word co-occurrences in the Web. However, it has been shown empirically that there exist a high correlation between word counts obtained from a Web search engine (e.g. Google and Altavista) and that from a corpus (e.g. British National corpus) [22]. Moreover, the approximated page counts have been sucessfully used to improve a variety of language modelling tasks [23].

3.3 Lexical Pattern Extraction

Page counts-based co-occurrence measures described in Section 3.2 do not consider the local context in which those words co-occur. This can be problematic if one or both words are polysemous, or when page counts are unreliable. On the other hand, the snippets returned by a search engine for the conjunctive query of two words provide useful clues related to the semantic relations that exist between two words. A snippet contains a window of text selected from a document that includes the queried words. Snippets are useful for 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. For example, consider the snippet in Figure 3.3. Here,

> *"**Cricket** is a **sport** played between two teams, each with eleven players."*

Fig. 3. A snippet retrieved for the query *"cricket" AND "sport"*.

the phrase *is a* indicates a semantic relationship between **cricket** and **sport**. Many such phrases indicate semantic relationships. For example, *also known as, is a, part of, is*

Ostrich, a large, flightless bird that lives in the dry grasslands of Africa.

Fig. 4. A snippet retrieved for the query *"ostrich * * * * * bird"*.

an example of all indicate semantic relations of different types. In the example given above, words indicating the semantic relation between *cricket* and *sport* appear between the query words. Replacing the query words by variables X and Y we can form the pattern X *is a* Y from the example given above.

Despite the efficiency of using snippets, they pose two main challenges: first, a snippet can be a fragmented sentence, second a search engine might produce a snippet by selecting multiple text fragments from different portions in a document. Because most syntactic or dependency parsers assume complete sentences as the input, deep parsing of snippets produces incorrect results. Consequently, we propose a shallow lexical pattern extraction algorithm using web snippets, to recognize the semantic relations that exist between two words. Lexical syntactic patterns have been used in various natural language processing tasks such as extracting hypernyms [5], [24], or meronyms [25], question answering [26], and paraphrase extraction [27]. Although a search engine might produce a snippet by selecting multiple text fragments from different portions in a document, a pre-defined delimiter is used to separate the different fragments. For example, in Google, the delimiter "..." is used to separate different fragments in a snippet. We use such delimiters to split a snippet before we run the proposed lexical pattern extraction algorithm on each fargment.

Given two words P and Q, we query a web search engine using the wildcard query *"P * * * * * Q"* and download snippets. The "*" operator matches one word or none in a web page. Therefore, our wildcard query retrieves snippets in which P and Q appear within a window of seven words. Because a search engine snippet contains ca. 20 words on average, and includes two fragments of texts selected from a document, we assume that the seven word window is sufficient to cover most relations between two words in snippets. In fact, over 95% of the lexical patterns extracted by the proposed method contain less than five words. We attempt to approximate the local context of two words using wildcard queries. For example, Fig. 4 shows a snippet retrieved for the query *"ostrich * * * * * bird"*.

For a snippet δ, retrieved for a word pair (P, Q), first, we replace the two words P and Q, respectively, with two variables X and Y. We replace all numeric values by D, a marker for digits. Next, we generate all subsequences of words from δ that satisfy all of the following conditions.

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

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

(iii). A subsequence is allowed to skip one or more words. However, we do not skip more than g number of words consecutively. Moreover, the total number of words skipped in a subsequence should not exceed G.

(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.

Finally, we count the frequency of all generated subsequences and only use subsequences that occur more than T times as lexical patterns.

The parameters L, g, G and T are set experimentally, as explained later in Section 3.6. It is noteworthy that the proposed pattern extraction algorithm considers all the words in a snippet, and is *not* limited to extracting patterns only from the mid-fix (i.e., the 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 [28] to generate subsequences from a text snippet. Specifically, we use the constraints (ii)-(iv) to prune the search space of candidate subsequences. For example, if a subsequence has reached the maximum length L, or the number of skipped words is G, then we will not extend it further. By pruning the search space, we can speed up the pattern generation process. However, none of these modifications affect the accuracy of the proposed semantic similarity measure because the modified version of the prefixspan algorithm still generates the exact set of patterns that we would obtain if we used the original prefixspan algorithm (i.e. without pruning) and subsequently remove patterns that violate the above mentioned constraints. For example, some patterns extracted form the snippet shown in Fig. 4 are: $\boldsymbol{X}$, *a large* $\boldsymbol{Y}$, $\boldsymbol{X}$ *a flightless* $\boldsymbol{Y}$, and $\boldsymbol{X}$, *large* $\boldsymbol{Y}$ *lives*.

3.4 Lexical Pattern Clustering

Typically, a semantic relation can be expressed using more than one pattern. For example, consider the two distinct patterns, $\boldsymbol{X}$ *is a* $\boldsymbol{Y}$, and $\boldsymbol{X}$ *is a large* $\boldsymbol{Y}$. Both these patterns indicate that there exists an *is-a* relation between $\boldsymbol{X}$ and $\boldsymbol{Y}$. Identifying the different patterns that express the same semantic relation enables us to represent the relation between two words accurately. According to the distributional hypothesis [29], words that occur in the same context have similar meanings. The distributional hypothesis has been used in various related tasks, such as identifying related words [16], and extracting paraphrases [27]. If we consider the word pairs that satisfy (i.e. co-occur with) a particular lexical pattern as the *context* of that lexical pair, then from the distributional hypothesis it follows that the lexical

Algorithm 1 Sequential pattern clustering algorithm.

Input: patterns $\Lambda = \{\mathbf{a_1}, \ldots, \mathbf{a_n}\}$, threshold θ
Output: clusters C

```
1: SORT(Λ)
2: C ← {}
3: for pattern a_i ∈ Λ do
4:    max ← −∞
5:    c* ← null
6:    for cluster c_j ∈ C do
7:       sim ← cosine(a_i, c_j)
8:       if sim > max then
9:          max ← sim
10:         c* ← c_j
11:      end if
12:   end for
13:   if max > θ then
14:      c* ← c* ⊕ a_i
15:   else
16:      C ← C ∪ {a_i}
17:   end if
18: end for
19: return C
```

patterns which are similarly distributed over word pairs must be semantically similar.

We represent a pattern a by a vector $\mathbf{a}$ of word-pair frequencies. We designate $\mathbf{a}$, the *word-pair frequency* vector of pattern a. It is analogous to the *document frequency* vector of a word, as used in information retrieval. The value of the element corresponding to a word pair (P_i, Q_i) in $\mathbf{a}$, is the frequency, $f(P_i, Q_i, a)$, that the pattern a occurs with the word pair (P_i, Q_i). As demonstrated later, the proposed pattern extraction algorithm typically extracts a large number of lexical patterns. Clustering algorithms based on pairwise comparisons among all patterns are prohibitively time consuming when the patterns are numerous. Next, we present a sequential clustering algorithm to efficiently cluster the extracted patterns.

Given a set Λ of patterns 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 into descending order of their total occurrences in all word pairs. The total occurrence $\mu(a)$ of a pattern a is the sum of frequencies over all word pairs, and is given by,

$$\mu(a) = \sum_i f(P_i, Q_i, a). \tag{5}$$

After sorting, the most common patterns appear at the beginning in Λ, whereas rare patterns (i.e., patterns that occur with only few word pairs) get shifted to the end. Next, in line 2, we initialize the set of clusters, C, to the empty set. The outer for-loop (starting at line 3), repeatedly takes a pattern $\mathbf{a_i}$ from the ordered set Λ, and in the inner for-loop (starting at line 6), finds the cluster, c^* ($\in C$) that is most similar to $\mathbf{a_i}$. First, we represent a cluster by the centroid of all word pair frequency vectors corresponding to the patterns in that cluster to compute the similarity between a pattern and a cluster. Next, we compute the cosine similarity between the cluster centroid ($\mathbf{c_j}$), and the word pair frequency vector of the pattern ($\mathbf{a_i}$). If the similarity between a pattern $\mathbf{a_i}$, and its most similar cluster, $\mathbf{c}^*$, is greater than the threshold θ, we append $\mathbf{a_i}$ to $\mathbf{c}^*$ (line 14). We use the operator $\oplus$ to denote the vector addition between $\mathbf{c}^*$ and $\mathbf{a_i}$. Then we form a new cluster $\{\mathbf{a_i}\}$ and append it to the set of clusters, C, if $\mathbf{a_i}$ is not similar to any of the existing clusters beyond the threshold θ.

By sorting the lexical patterns in the descending order of their frequency and clustering the most frequent patterns first, we form clusters for more common relations first. This enables us to separate rare patterns which are likely to be outliers from attaching to otherwise clean clusters. The greedy sequential nature of the algorithm avoids pair-wise comparisons between all lexical patterns. This is particularly important because when the number of lexical patterns is large as in our experiments (e.g. over $100,000$), pair-wise comparisons between all patterns is computationally prohibitive. The proposed clustering algorithm attempts to identify the lexical patterns that are similar to each other more than a given threshold value. By adjusting the threshold we can obtain clusters with different granularity.

The only parameter in Algorithm 1, the similarity threshold, θ, 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 yield numerous clusters (increased model complexity). In Section 3.6, we investigate, experimentally, the effect of θ on the overall performance of the proposed relational similarity measure.

The initial sort operation in Algorithm 1 can be carried out in time complexity of $O(n\log n)$, where n is the number of patterns to be clustered. Next, the sequential assignment of lexical patterns to the clusters requires complexity of $O(n|C|)$, where $|C|$ is the number of clusters. Typically, n is much larger than $|C|$ (i.e. $n \gg |C|$). Therefore, the overall time complexity of Algorithm 1 is dominated by the sort operation, hence $O(n\log n)$. The sequential nature of the algorithm avoids pairwise comparisons among 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.

3.5 Measuring Semantic Similarity

In Section 3.2 we defined four co-occurrence measures using page counts. Moreover, in Sections 3.3 and 3.4 we showed how to extract clusters of lexical patterns from snippets to represent numerous semantic relations that exist between two words. In this section, we describe a machine learning approach to combine

both page counts-based co-occurrence measures, and snippets-based lexical pattern clusters to construct a robust semantic similarity measure.

Given N clusters of lexical patterns, first, we represent a pair of words (P, Q) by an $(N+4)$ dimensional feature vector $\mathbf{f}_{PQ}$. The four page counts-based co-occurrence measures defined in Section 3.2 are used as four distinct features in $\mathbf{f}_{PQ}$. For completeness let us assume that $(N+1)$-st, $(N+2)$-nd, $(N+3)$-rd, and $(N+4)$-th features are set respectively to WebJaccard, WebOverlap, WebDice, and WebPMI. Next, we compute a feature from each of the N clusters as follows. First, we assign a weight w_{ij} to a pattern a_i that is in a cluster c_j as follows,

$$w_{ij} = \frac{\mu(a_i)}{\sum_{t \in c_j} \mu(t)}. \tag{6}$$

Here, $\mu(a)$ is the total frequency of a pattern a in all word pairs, and it is given by (5). Because we perform a hard clustering on patterns, a pattern can belong to only one cluster (i.e. $w_{ij} = 0$ for $a_i \notin c_j$). Finally, we compute the value of the j-th feature in the feature vector for a word pair (P, Q) as follows,

$$\sum_{a_i \in c_j} w_{ij} f(P, Q, a_i). \tag{7}$$

The value of the j-th feature of the feature vector $\mathbf{f}_{PQ}$ representing a word pair (P, Q) can be seen as the weighted sum of all patterns in cluster c_j that co-occur with words P and Q. We assume that all patterns in a cluster to represent a particular semantic relation. Consequently, the j-th feature value given by (7) expresses the significance of the semantic relation represented by cluster j for word pair (P, Q). For example, if the weight w_{ij} is set to 1 for all patterns a_i in a cluster c_j, then the j-th feature value is simply the sum of frequencies of all patterns in cluster c_j with words P and Q. However, assigning an equal weight to all patterns in a cluster is not desirable in practice because some patterns can contain misspellings and/or can be grammatically incorrect. Equation (6) assigns a weight to a pattern proportionate to its frequency in a cluster. If a pattern has a high frequency in a cluster, then it is likely to be a canonical form of the relation represented by all the patterns in that cluster. Consequently, the weighting scheme described by Equation (6) prefers high frequent patterns in a cluster.

To train a two-class SVM to detect synonymous and non-synonymous word pairs, we utilize a training dataset $S = \{(P_k, Q_k, y_k)\}$ of word pairs. S consists of synonymous word pairs (positive training instances) and non-synonymous word pairs (negative training instances). Training dataset S is generated automatically from WordNet synsets as described later in Section 3.6. Label $y_k \in \{-1, 1\}$ indicates whether the word pair (P_k, Q_k) is a synonymous word pair (i.e. $y_k = 1$) or a non-synonymous word pair (i.e. $y_k = -1$). For each word pair in S, we create an $(N+4)$ dimensional feature vector as described above. To simplify the notation, let us denote the feature vector of a word pair (P_k, Q_k) by $\mathbf{f_k}$. Finally, we train a two-class SVM using the labeled feature vectors.

Once we have trained an SVM using synonymous and non-synonymous word pairs, we can use it to compute the semantic similarity between two given words. Following the same method we used to generate feature vectors for training, we create an $(N+4)$ dimensional feature vector $\mathbf{f}^*$ for a pair of words (P^*, Q^*), between which we must measure semantic similarity. We define the semantic similarity $\mathrm{sim}(P^*, Q^*)$ between P^* and Q^* as the posterior probability, $p(y^* = 1|\mathbf{f}^*)$, that the feature vector $\mathbf{f}^*$ corresponding to the word-pair (P^*, Q^*) belongs to the synonymous-words class (i.e. $y^* = 1$). $\mathrm{sim}(P^*, Q^*)$ is given by,

$$\mathrm{sim}(P^*, Q^*) = p(y^* = 1|\mathbf{f}^*). \tag{8}$$

Because SVMs are large margin classifiers, the output of an SVM is the distance from the classification hyperplane. The distance $d(\mathbf{f}^*)$ to an instance $\mathbf{f}^*$ from the classification hyperplane is given by,

$$d(\mathbf{f}^*) = h(\mathbf{f}^*) + b.$$

Here, b is the bias term and the hyperplane, $h(\mathbf{f}^*)$, is given by,

$$h(\mathbf{f}^*) = \sum_i y_k \alpha_k K(\mathbf{f_k}, \mathbf{f}^*).$$

Here, α_k is the Lagrange multiplier corresponding to the support vector $\mathbf{f_k}$[3], and $K(\mathbf{f_k}, \mathbf{f}^*)$ is the value of the kernel function for a training instance $\mathbf{f_k}$ and the instance to classify, $\mathbf{f}^*$. However, $d(\mathbf{f}^*)$ is not a calibrated posterior probability. Following Platt [30], we use sigmoid functions to convert this uncalibrated distance into a calibrated posterior probability. The probability, $p(y = 1|d(\mathbf{f}))$, is computed using a sigmoid function defined over $d(\mathbf{f})$ as follows,

$$p(y = 1|d(\mathbf{f})) = \frac{1}{1 + \exp(\lambda d(\mathbf{f}) + \mu)}.$$

Here, λ and μ are parameters which are determined by maximizing the likelihood of the training data. Log-likelihood of the training data is given by,

$$\begin{aligned} L(\lambda, \mu) &= \sum_{k=1}^{N} \log p(y_k|\mathbf{f_k}; \lambda, \mu) \\ &= \sum_{k=1}^{N} \{t_k \log(p_k) + (1 - t_k) \log(1 - p_k)\}. \end{aligned} \tag{9}$$

Here, to simplify the notation we have used $t_k = (y_k + 1)/2$ and $p_k = p(y_k = 1|\mathbf{f_k})$. The maximization in (9) with respect to parameters λ and μ is performed using model-trust minimization [31].

3. From K.K.T. conditions it follows that the Lagrange multipliers corresponding to non-support vectors become zero.

TABLE 1
No. of patterns extracted for training data.

word pairs	synonymous	non-synonymous
# word pairs	3000	3000
# extracted patterns	5365398	515848
# selected patterns	270762	38978

3.6 Training

To train the two-class SVM described in Section 3.5, we require both synonymous and non-synonymous word pairs. We use WordNet, a manually created English dictionary, to generate the training data required by the proposed method. For each sense of a word, a set of synonymous words is listed in WordNet synsets. We randomly select 3000 nouns from WordNet, and extract a pair of synonymous words from a synset of each selected noun. If a selected noun is polysemous, then we consider the synset for the dominant sense. Obtaining a set of non-synonymous word pairs (negative training instances) is difficult, because there does not exist a large collection of manually created non-synonymous word pairs. Consequently, to create a set of non-synonymous word pairs, we adopt a random shuffling technique. Specifically, we first randomly select two synonymous word pairs from the set of synonymous word pairs created above, and exchange two words between word pairs to create two new word pairs. For example, from two synonymous word pairs (A, B) and (C, D), we generate two new pairs (A, C) and (B, D). If the newly created word pairs do not appear in any of the word net synsets, we select them as non-synonymous word pairs. We repeat this process until we create 3000 non-synonymous word pairs. Our final training dataset contains 6000 word pairs (i.e. 3000 synonymous word pairs and 3000 non-synonymous word pairs).

Next, we use the lexical pattern extraction algorithm described in Section 3.3 to extract numerous lexical patterns for the word pairs in our training dataset. We experimentally set the parameters in the pattern extraction algorithm to $L = 5$, $g = 2$, $G = 4$, and $T = 5$. Table 1 shows the number of patterns extracted for synonymous and non-synonymous word pairs in the training dataset. As can be seen from Table 1, the proposed pattern extraction algorithm typically extracts a large number of lexical patterns. Figs. 5 and 6 respectively, show the distribution of patterns extracted for synonymous and non-synonymous word pairs. Because of the noise in web snippets such as, ill-formed snippets and misspells, most patterns occur only a few times in the list of extracted patterns. Consequently, we ignore any patterns that occur less than 5 times. Finally, we de-duplicate the patterns that appear for both synonymous and non-synonymous word pairs to create a final set of 302286 lexical patterns. The remainder of the experiments described in the paper use this set of lexical patterns.

We determine the clustering threshold θ as follows,

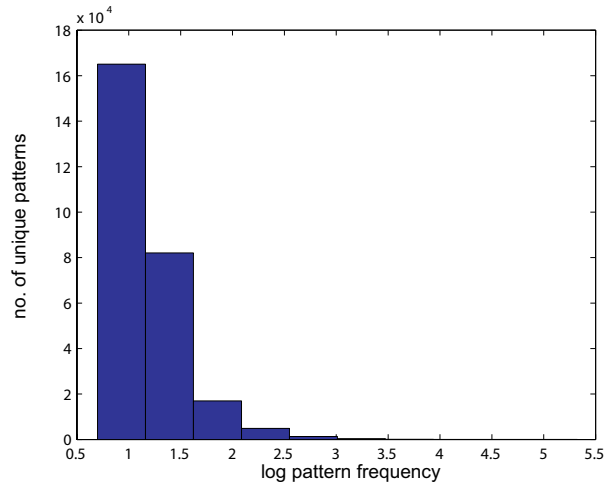

Fig. 5. Distribution of patterns extracted from synonymous word pairs.

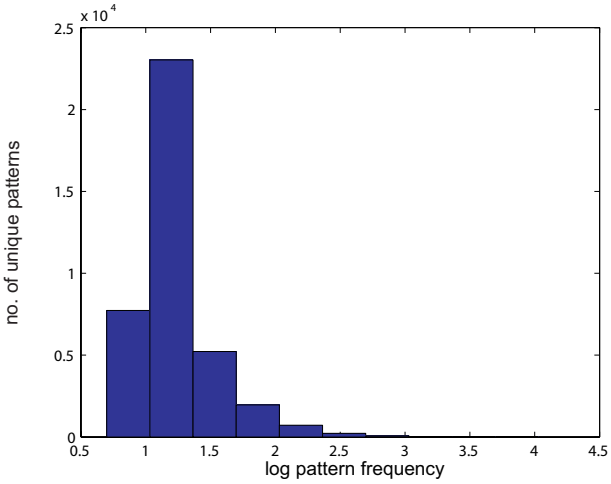

Fig. 6. Distribution of patterns extracted from non-synonymous word pairs.

First, we we run Algorithm 1 for different θ values, and with each set of clusters we compute feature vectors for synonymous word pairs as described in Section 3.5. Let W denote the set of synonymous word pairs (i.e. $W = \{(P_i, Q_i) | (P_i, Q_i, y_i) \in S, y_i = 1\}$). Moreover, let $\mathbf{f_W}$ be the centroid vector of all feature vectors representing synonymous word pairs, which is given by,

$$\mathbf{f_W} = \frac{1}{|W|} \sum_{(P,Q) \in W} \mathbf{f_{PQ}}. \tag{10}$$

Next, we compute the average Mahalanobis distance, $D(\theta)$, between $\mathbf{f_W}$ and feature vectors that represent synonymous as follows,

$$D(\theta) = \frac{1}{|W|} \sum\nolimits_{(P,Q) \in W} \text{Mahala}(\mathbf{f_W}, \mathbf{f_{PQ}}). \tag{11}$$

Here, $|W|$ is the number of word pairs in W, and

$\text{mahala}(\mathbf{f_W}, \mathbf{f_{PQ}})$ is the Mahalanobis distance defined by,

$$\text{mahala}(\mathbf{f_w}, \mathbf{f_{PQ}}) = (\mathbf{f_w} - \mathbf{f_{PQ}})^{\mathrm{T}} C^{-1} (\mathbf{f_w} - \mathbf{f_{PQ}}) \quad (12)$$

Here, C^{-1} is the inverse of the inter-cluster correlation matrix, C, where the (i, j) element of C is defined to be the inner-product between the vectors $\mathbf{c_i}$, $\mathbf{c_j}$ corresponding to clusters c_i and c_j. Finally, we set the optimum value of clustering threshold, $\hat{\theta}$, to the value of θ that minimizes the average Mahalanobis distance as follows,

$$\hat{\theta} = \underset{\theta \in [0,1]}{\arg\min}\, D(\theta).$$

Alternatively, we can define the reciprocal of $D(\theta)$ as average similarity, and minimize this quantity. Note that the average in (11) is taken over a large number of synonymous word pairs (3000 word pairs in W), which enables us to determine θ robustly. Moreover, we consider Mahalanobis distance instead of Euclidean distances, because a set of pattern clusters might not necessarily be independent. For example, we would expect a certain level of correlation between the two clusters that represent an *is-a* relation and a *has-a* relation. Mahalanobis distance consider the correlation between clusters when computing distance. Note that if we take the identity matrix as C in (12), then we get the Euclidean distance.

Fig. 7 plots average similarity between centroid feature vector and all synonymous word pairs for different values of θ. From Fig. 7, we see that initially average similarity increases when θ is increased. This is because clustering of semantically related patterns reduces the sparseness in feature vectors. Average similarity is stable within a range of θ values between 0.5 and 0.7. However, increasing θ beyond 0.7 results in a rapid drop of average similarity. To explain this behavior consider Fig. 8 where we plot the sparsity of the set of clusters (i.e. the ratio between singletons to total clusters) against threshold θ. As seen from Fig. 8, high θ values result in a high percentage of singletons because only highly similar patterns will form clusters. Consequently, feature vectors for different word pairs do not have many features in common. The maximum average similarity score of 1.31 is obtained with $\theta = 0.7$, corresponding to $32,207$ total clusters out of which $23,836$ are singletons with exactly one pattern (sparsity = 0.74). For the remainder of the experiments in this paper we set θ to this optimal value and use the corresponding set of clusters.

We train an SVM with a radial basis function (RBF) kernel. Kernel parameter γ and soft-margin trade-off C is respectively set to 0.0078125 and 131072 using 5-fold cross-validation on training data. We used LibSVM[4] as the SVM implementation. Remainder of the experiments in the paper use this trained SVM model.

4. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

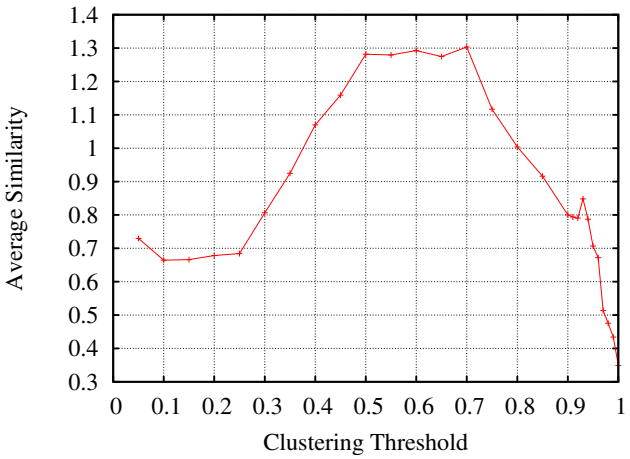

Fig. 7. Average similarity vs. clustering threshold θ

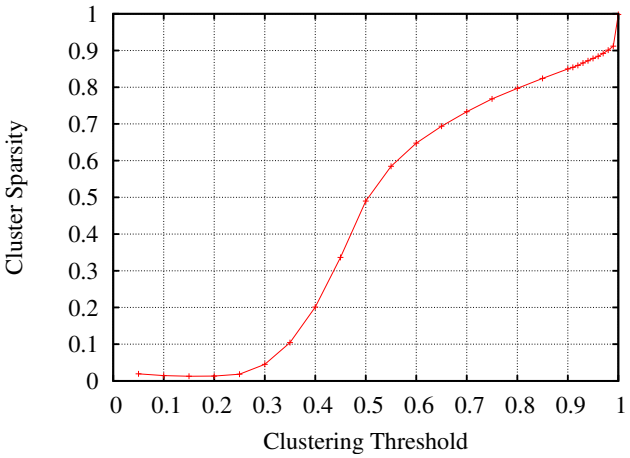

Fig. 8. Sparsity vs. clustering threshold θ

4 Experiments

4.1 Benchmark Datasets

Following the previous work, we evaluate the proposed semantic similarity measure by comparing it with human ratings in three benchmark datasets: Miller-Charles (MC) [10], Rubenstein-Goodenough (RG) [32], and WordSimilarity-353 (WS) [33]. Each dataset contains a list of word pairs rated by multiple human annotators (MC: 28 pairs, 38 annotators, RG: 65 pairs, 36 annotators, WS: 353 pairs, 13 annotators). A semantic similarity measure is evaluated using the correlation between the similarity scores produced by it for the word pairs in a benchmark dataset and the human ratings. Both Pearson correlation coefficient and Spearman correlation coefficient have been used as evaluation measures in previous work on semantic similarity. It is noteworthy that Pearson correlation coefficient can get severely affected by non-linearities in ratings. Contrastingly, Spearman correlation coefficient first assigns ranks to each list of scores, and then computes correlation between the two lists of ranks. Therefore, Spearman correlation is more appropriate for evaluating semantic similarity measures,

which might not be necessarily linear. In fact, as we shall see later, most semantic similarity measures are non-linear. Previous work that used RG and WS datasets in their evaluations have chosen Spearman correlation coefficient as the evaluation measure. However, for the MC dataset, which contains only 28 word pairs, Pearson correlation coefficient has been widely used. To be able to compare our results with previous work, we use both Pearson and Spearman correlation coefficients for experiments conducted on MC dataset, and Spearman correlation coefficient for experiments on RG and WS datasets. It is noteworthy that we exclude all words that appear in the benchmark dataset from the training data created from WordNet as described in Section 3.6. Benchmark datasets are reserved for evaluation purposes only and we do not train or tune any of the parameters using benchmark datasets.

4.2 Semantic Similarity

Table 2 shows the experimental results on MC dataset for the proposed method (**Proposed**); previously proposed web-based semantic similarity measures: Sahami and Heilman [2] (**SH**), Co-occurrence double checking model [4] (**CODC**), and Normalized Google Distance [12] (**NGD**); and the four page counts-based co-occurrence measures in Section 3.2. **No Clust** baseline , which resembles our previously published work [34], is identical to the proposed method in all aspects except for that it does not use cluster information. It can be understood as the proposed method with each extracted lexical pattern in its own cluster. No Clust is expected to show the effect of clustering on the performance of the proposed method. All similarity scores in Table 2 are normalized into $[0, 1]$ range for the ease of comparison, and Fisher's confidence intervals are computed for Spearman and Pearson correlation coefficients. NGD is a distance measure and was converted to a similarity score by taking the inverse. Original papers that proposed NGD and SH measures did not present their results on MC dataset. Therefore, we re-implemented those systems following the original papers. The proposed method achieves the highest Pearson and Spearman coefficients in Table 2 and outperforms all other web-based semantic similarity measures. WebPMI reports the highest correlation among all page counts-based co-occurrence measures. However, methods that use snippets such as SH and CODC, have better correlation scores. MC dataset contains polysemous words such as *father* (priest vs. parent), *oracle* (priest vs. database system) and *crane* (machine vs. bird), which are problematic for page counts-based measures that do not consider the local context of a word. The No Clust baseline which combines both page counts and snippets outperforms the CODC measure by a wide margin of 0.2 points. Moreover, by clustering the lexical patterns we can further improve the No Clust baseline.

Table 3 summarizes the experimental results on RG and WS datasets. Likewise on the MC dataset, the proposed method outperforms all other methods on RG and WS datasets. In contrast to MC dataset, the proposed method outperforms the No Clust baseline by a wide margin in RG and WS datasets. Unlike the MC dataset which contains only 28 word pairs, RG and WS datasets contain a large number of word pairs. Therefore, more reliable statistics can be computed on RG and WS datasets. Fig. 9 shows the similarity scores produced by six methods against human ratings in the WS dataset. We see that all methods deviate from the $y = x$ line, and are not linear. We believe this justifies the use of Spearman correlation instead of Pearson correlation by previous work on semantic similarity as the preferred evaluation measure.

TABLE 3
Correlation with RG and WS datasets.

Method	WS	RG
WebJaccard	0.26 $[0.16, 0.35]$	0.51 $[0.30, 0.67]$
WebDice	0.26 $[0.16, 0.35]$	0.51 $[0.30, 0.67]$
WebOverlap	0.27 $[0.17, 0.36]$	0.54 $[0.34, 0.69]$
WebPMI	0.36 $[0.26, 0.45]$	0.49 $[0.28, 0.66]$
CODC [4]	0.55 $[0.48, 0.62]$	0.65 $[0.49, 0.77]$
SH [2]	0.36 $[0.26, 0.45]$	0.31 $[0.07, 0.52]$
NGD [12]	0.40 $[0.31, 0.48]$	0.56 $[0.37, 0.71]$
No Clust	0.53 $[0.45, 0.60]$	0.73 $[0.60, 0.83]$
Proposed	0.74 $[0.69, 0.78]$	0.86 $[0.78, 0.91]$

TABLE 4
Comparison with previous work on MC dataset.

Method	Source	Pearson
Wikirelate! [35]	Wikipedia	0.46
Sahami & Heilman [2]	Web Snippets	0.58
Gledson [36]	Page Counts	0.55
Wu & Palmer [37]	WordNet	0.78
Resnik [8]	WordNet	0.74
Leacock [38]	WordNet	0.82
Lin [11]	WordNet	0.82
Jiang & Conrath [39]	WordNet	0.84
Jarmasz [40]	Roget's	0.87
Li et al. [9]	WordNet	0.89
Schickel-Zuber [41]	WordNet	0.91
Agirre et al [42]	WordNet+Corpus	0.93
Proposed	WebSnippets+Page Counts	0.87

Tables 4, 5, and 6 respectively compare the proposed method against previously proposed semantic similarity measures. Despite the fact that the proposed method does not require manually created resources such as WordNet, Wikipedia or fixed corpora, the performance of the proposed method is comparable with methods that use such resources. The non-dependence on dictionaries is particularly attractive when measuring the similarity between named-entities which are not well-covered by dictionaries such as WordNet. We further evaluate the ability of the proposed method to compute the semantic similarity between named-entities in Section 4.3.

In Table 7 we analyze the effect of clustering. We compare No Clust (i.e. does not use any clustering information in feature vector creation), singletons excluded (remove all clusters with only one pattern), and single-

TABLE 2
Semantic similarity scores on MC dataset

word pair	MC	WebJaccard	WebDice	WebOverlap	WebPMI	CODC [4]	SH [2]	NGD [12]	No Clust	Proposed
automobile-car	1.00	0.65	0.66	0.83	0.43	0.69	1.00	0.15	0.98	0.92
journey-voyage	0.98	0.41	0.42	0.16	0.47	0.42	0.52	0.39	1.00	1.00
gem-jewel	0.98	0.29	0.30	0.07	0.69	1.00	0.21	0.42	0.69	0.82
boy-lad	0.96	0.18	0.19	0.59	0.63	0.00	0.47	0.12	0.97	0.96
coast-shore	0.94	0.78	0.79	0.51	0.56	0.52	0.38	0.52	0.95	0.97
asylum-madhouse	0.92	0.01	0.01	0.08	0.81	0.00	0.21	1.00	0.77	0.79
magician-wizard	0.89	0.29	0.30	0.37	0.86	0.67	0.23	0.44	1.00	1.00
midday-noon	0.87	0.10	0.10	0.12	0.59	0.86	0.29	0.74	0.82	0.99
furnace-stove	0.79	0.39	0.41	0.10	1.00	0.93	0.31	0.61	0.89	0.88
food-fruit	0.78	0.75	0.76	1.00	0.45	0.34	0.18	0.55	1.00	0.94
bird-cock	0.77	0.14	0.15	0.14	0.43	0.50	0.06	0.41	0.59	0.87
bird-crane	0.75	0.23	0.24	0.21	0.52	0.00	0.22	0.41	0.88	0.85
implement-tool	0.75	1.00	1.00	0.51	0.30	0.42	0.42	0.91	0.68	0.50
brother-monk	0.71	0.25	0.27	0.33	0.62	0.55	0.27	0.23	0.38	0.27
crane-implement	0.42	0.06	0.06	0.10	0.19	0.00	0.15	0.40	0.13	0.06
brother-lad	0.41	0.18	0.19	0.36	0.64	0.38	0.24	0.26	0.34	0.13
car-journey	0.28	0.44	0.45	0.36	0.20	0.29	0.19	0.00	0.29	0.17
monk-oracle	0.27	0.00	0.00	0.00	0.00	0.00	0.05	0.45	0.33	0.80
food-rooster	0.21	0.00	0.00	0.41	0.21	0.00	0.08	0.42	0.06	0.02
coast-hill	0.21	0.96	0.97	0.26	0.35	0.00	0.29	0.70	0.87	0.36
forest-graveyard	0.20	0.06	0.06	0.23	0.49	0.00	0.00	0.54	0.55	0.44
monk-slave	0.12	0.17	0.18	0.05	0.61	0.00	0.10	0.77	0.38	0.24
coast-forest	0.09	0.86	0.87	0.29	0.42	0.00	0.25	0.36	0.41	0.15
lad-wizard	0.09	0.06	0.07	0.05	0.43	0.00	0.15	0.66	0.22	0.23
cord-smile	0.01	0.09	0.10	0.02	0.21	0.00	0.09	0.13	0.00	0.01
glass-magician	0.01	0.11	0.11	0.40	0.60	0.00	0.14	0.21	0.18	0.05
rooster-voyage	0.00	0.00	0.00	0.00	0.23	0.00	0.20	0.21	0.02	0.05
noon-string	0.00	0.12	0.12	0.04	0.10	0.00	0.08	0.21	0.02	0.00
Spearman	1.00	0.39	0.39	0.40	0.52	0.69	0.62	0.13	0.83	0.85
Lower	1.00	0.02	0.02	0.04	0.18	0.42	0.33	−0.25	0.66	0.69
Upper	1.00	0.67	0.67	0.68	0.75	0.84	0.81	0.48	0.92	0.93
Pearson	1.00	0.26	0.27	0.38	0.55	0.69	0.58	0.21	0.83	0.87
Lower	1.00	−0.13	−0.12	0.01	0.22	0.42	0.26	−0.18	0.67	0.73
Upper	1.00	0.58	0.58	0.66	0.77	0.85	0.78	0.54	0.92	0.94

TABLE 5
Comparison with previous work on RG dataset.

Method	Source	Spearman
Wikirelate! [35]	Wikipedia	0.56
Gledson [36]	Page Counts	0.55
Jiang & Conrath [39]	WordNet	0.73
Hirst & St. Onge [43]	WordNet	0.73
Resnik [8]	WordNet	0.80
Lin [11]	WordNet	0.83
Leacock [38]	WordNet	0.85
Proposed	WebSnippets+Page Counts	0.86

TABLE 6
Comparison with previous work on WS dataset.

Method	Source	Spearman
Jarmasz [40]	WordNet	0.35
Wikirelate! [35]	Wikipedia	0.48
Jarmasz [40]	Roget's	0.55
Hughes & Ramage [44]	WordNet	0.55
Finkelstein et al. [33]	Corpus+WordNet	0.56
Gabrilovich [45]	ODP	0.65
Gabrilovich [45]	Wikipedia	0.75
Proposed	WebSnippets+Page Counts	0.74

tons included (considering all clusters). From Table 7 we see in all three datasets, we obtain the best results by considering all clusters (singletons incl.). If we remove all singletons, then the performance drops below No Clust.

TABLE 7
Effect of pattern clustering (Spearman).

Method	MC	RG	WS
No Clust	0.83	0.73	0.53
[upper, lower]	[0.66, 0.92]	[0.60, 0.83]	[0.45, 0.60]
singletons excl.	0.59	0.68	0.49
[upper, lower]	[0.29, 0.79]	[0.52, 0.79]	[0.40, 0.56]
singletons incl.	0.85	0.86	0.74
[upper, lower]	[0.69, 0.93]	[0.78, 0.91]	[0.69, 0.78]

TABLE 8
Page counts vs Snippets. (Spearman)

Method	MC	RG	WS
Page counts only	0.57	0.57	0.37
[upper, lower]	[0.25, 0.77]	[0.39, 0.72]	[0.29, 0.46]
Snippets only	0.82	0.85	0.72
[upper, lower]	[0.65, 0.91]	[0.77, 0.90]	[0.71, 0.79]
Both	0.85	0.86	0.74
[upper, lower]	[0.69, 0.93]	[0.78, 0.91]	[0.69, 0.78]

Note that out of the 32.207 clusters used by the proposed method, 23, 836 are singletons (sparsity=0.74). Therefore, if we remove all singletons, we cannot represent some word pairs adequately, resulting in poor performance.

Table 8 shows the contribution of page counts-based similarity measures, and lexical patterns extracted from snippets, on the overall performance of the proposed

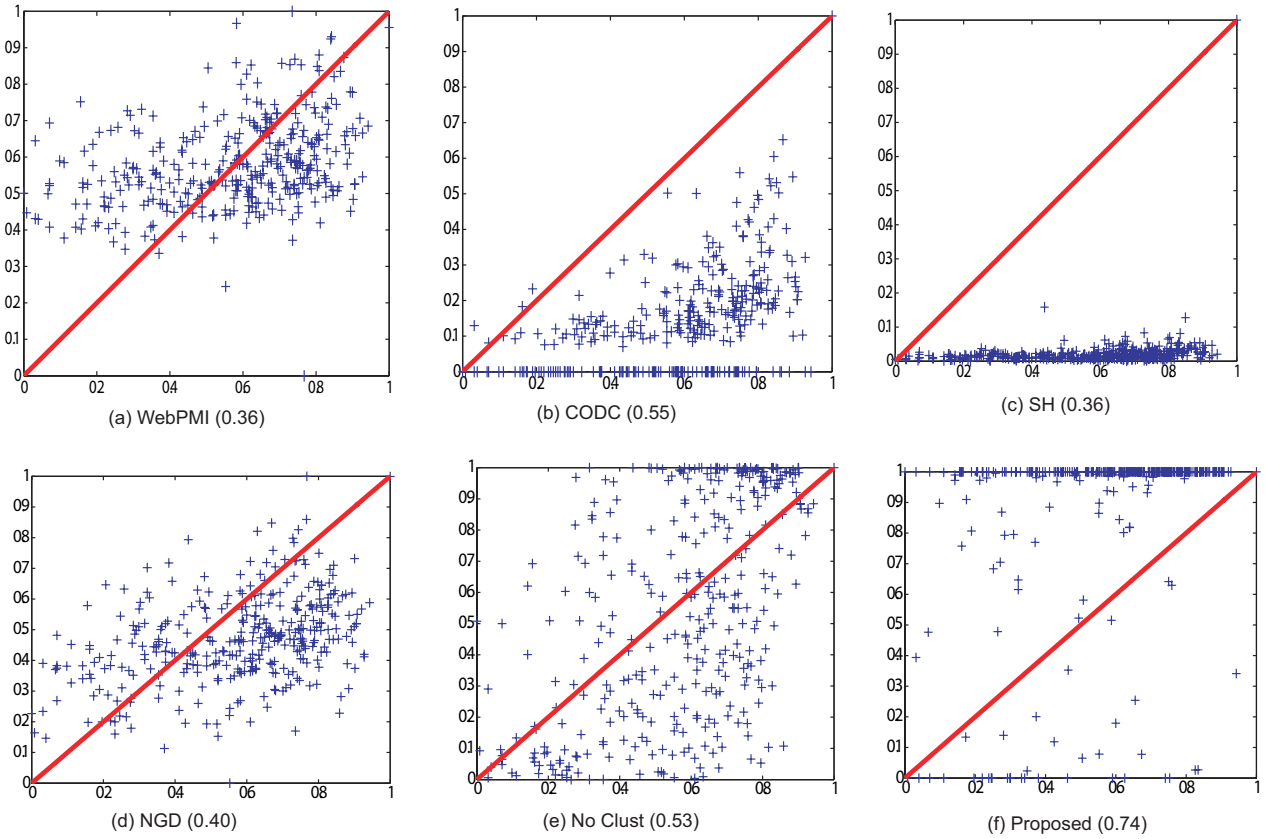

Fig. 9. Spearman rank correlation between various similarity measures (y-axis) and human ratings (x-axis) on the WS dataset.

method. To evaluate the effect of page counts-based co-occurrence measures on the proposed method, we generate feature vectors only using the four page counts-based co-occurrence measures, to train an SVM. Similarly, to evaluate the effect of snippets, we generate feature vectors only using lexical pattern clusters. From Table 8 we see that on all three datasets, snippets have a greater impact on the performance of the proposed method than page counts. By considering both page counts as well as snippets we can further improve the performance reported by individual methods. The improvement in performance when we use snippets only is statistically significant over that when we use page counts only in RG and WS datasets. However, the performance gain in the combination is not statistically significant. We believe that this is because most of the words in the benchmark datasets are common nouns that co-occur a lot in web snippets. On the other hand, having page counts in the model is particularly useful when two words do not appear in numerous lexical patterns.

4.3 Community Mining

Measuring the semantic similarity between named entities is vital in many applications such as query expansion [2], entity disambiguation (e.g. namesake disambiguation) and community mining [46]. Because most named entities are not covered by WordNet, similarity measures that are based on WordNet cannot be used directly in these tasks. Unlike common English words, named entities are being created constantly. Manually maintaining an up-to-date taxonomy of named entities is costly, if not impossible. The proposed semantic similarity measure is appealing for these applications because it does not require pre-compiled taxonomies.

In order to evaluate the performance of the proposed measure in capturing the semantic similarity between named-entities, we set up a community mining task. We select 50 personal names from 5 communities: *tennis players*, *golfers*, *actors*, *politicians* and *scientists*[5] , (10 names from each community) from the open directory project (DMOZ)[6]. For each pair of names in our data set, we measure their similarity using the proposed method and baselines. We use group-average agglomerative hierarchical clustering (GAAC) to cluster the names in our dataset into five clusters.

Initially, each name is assigned to a separate cluster. In subsequent iterations, group average agglomerative clustering process, merges the two clusters with highest correlation. Correlation, $\mathrm{Corr}(\Gamma)$ between two clusters A

5. www.miv.t.u-tokyo.ac.jp/danushka/data/people.tgz
6. http://dmoz.org

and B is defined as the following,

$$\mathrm{Corr}(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)}\sum_{(u,v)\in\Gamma}\mathrm{sim}(u,v)$$

Here, Γ is the merger of the two clusters A and B. $|\Gamma|$ denotes the number of elements (persons) in Γ and $\mathrm{sim}(u,v)$ is the semantic similarity between two persons u and v in Γ. We terminate GAAC process when exactly five clusters are formed. We adopt this clustering method with different semantic similarity measures $\mathrm{sim}(u,v)$ to compare their accuracy in clustering people who belong to the same community.

We employed the *B-CUBED* metric [47] to evaluate the clustering results. The B-CUBED evaluation metric was originally proposed for evaluating cross-document co-reference chains. It does not require the clusters to be labelled. We compute precision, recall and F-score for each name in the data set and average the results over the dataset. For each person p in our data set, let us denote the cluster that p belongs to by $C(p)$. Moreover, we use $A(p)$ to denote the affiliation of person p, e.g., A("Tiger Woods") ="Tennis Player". Then we calculate precision and recall for person p as,

$$\mathrm{Precision}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{No. of people in C(p)}},$$

$$\mathrm{Recall}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{Total No. of people with affiliation A(p)}}.$$

Since, we selected 10 people from each of the five categories, the total number of people with a particular affiliation is 10 for all the names p. Then, the F-score of person p is defined as,

$$\mathrm{F}(p) = \frac{2\times\mathrm{Precision}(p)\times\mathrm{Recall}(p)}{\mathrm{Precision}(p)+\mathrm{Recall}(p)}.$$

Overall precision, recall and F-score are computed by taking the averaged sum over all the names in the dataset.

$$\mathrm{Precision} = \frac{1}{N}\sum_{p\in DataSet}\mathrm{Precision}(p)$$

$$\mathrm{Recall} = \frac{1}{N}\sum_{p\in DataSet}\mathrm{Recall}(p)$$

$$F\text{−Score} = \frac{1}{N}\sum_{p\in DataSet}\mathrm{F}(p)$$

Here, $DataSet$ is the set of 50 names selected from the open directory project. Therefore, $N = 50$ in our evaluations.

Experimental results are shown in Table 9. The proposed method shows the highest entity clustering accuracy in Table 9 with a statistically significant ($p \leq 0.01$ Tukey HSD) F score of 0.86. Sahami et al. [2]'s snippet-based similarity measure, WebJaccard, WebDice and WebOverlap measures yield similar clustering accuracies. By clustering semantically related lexical patterns, we see that both precision as well as recall can be improved in a community mining task.

TABLE 9
Results for Community Mining

Method	Precision	Recall	F Measure
WebJaccard	0.59	0.71	0.61
WebOverlap	0.59	0.68	0.59
WebDice	0.58	0.71	0.61
WebPMI	0.26	0.42	0.29
Sahami [2]	0.63	0.66	0.64
Chen [4]	0.47	0.62	0.49
No Clust	0.79	0.80	0.78
Proposed	0.85	0.87	0.86

5 Conclusion

We proposed a semantic similarity measures using both page counts and snippets retrieved from a web search engine for two words. Four word co-occurrence measures were computed using page counts. We proposed a lexical pattern extraction algorithm to extract numerous semantic relations that exist between two words. Moreover, a sequential pattern clustering algorithm was proposed to identify different lexical patterns that describe the same semantic relation. Both page counts-based co-occurrence measures and lexical pattern clusters were used to define features for a word pair. A two-class SVM was trained using those features extracted for synonymous and non-synonymous word pairs selected from WordNet synsets. Experimental results on three benchmark datasets showed that the proposed method outperforms various baselines as well as previously proposed web-based semantic similarity measures, achieving a high correlation with human ratings. Moreover, the proposed method improved the F score in a community mining task, thereby underlining its usefulness in real-world tasks, that include named-entities not adequately covered by manually created resources.

References

[1] A. Kilgarriff, "Googleology is bad science," *Computational Linguistics*, vol. 33, pp. 147–151, 2007.

[2] M. Sahami and T. Heilman, "A web-based kernel function for measuring the similarity of short text snippets," in *Proc. of 15th International World Wide Web Conference*, 2006.

[3] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Disambiguating personal names on the web using automatically extracted key phrases," in *Proc. of the 17th European Conference on Artificial Intelligence*, 2006, pp. 553–557.

[4] H. Chen, M. Lin, and Y. Wei, "Novel association measures using web search with double checking," in *Proc. of the COLING/ACL 2006*, 2006, pp. 1009–1016.

[5] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proc. of 14th COLING*, 1992, pp. 539–545.

[6] 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-2006*, 2006.

[7] R. Rada, H. Mili, E. Bichnell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9(1), pp. 17–30, 1989.

[8] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proc. of 14th International Joint Conference on Aritificial Intelligence*, 1995.

[9] D. M. Y. Li, Zuhair A. Bandar, "An approch for measuring semantic similarity between words using multiple information sources," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15(4), pp. 871–882, 2003.

[10] G. Miller and W. Charles, "Contextual correlates of semantic similarity," *Language and Cognitive Processes*, vol. 6(1), pp. 1–28, 1998.

[11] D. Lin, "An information-theoretic definition of similarity," in *Proc. of the 15th ICML*, 1998, pp. 296–304.

[12] R. Cilibrasi and P. Vitanyi, "The google similarity distance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 370–383, 2007.

[13] M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi, "The similarity metric," *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 3250–3264, 2004.

[14] P. Resnik, "Semantic similarity in a taxonomy: An information based measure and its application to problems of ambiguity in natural language," *Journal of Aritificial Intelligence Research*, vol. 11, pp. 95–130, 1999.

[15] R. Rosenfield, "A maximum entropy approach to adaptive statistical modelling," *Computer Speech and Language*, vol. 10, pp. 187–228, 1996.

[16] D. Lin, "Automatic retrieival and clustering of similar words," in *Proc. of the 17th COLING*, 1998, pp. 768–774.

[17] J. Curran, "Ensemble menthods for automatic thesaurus extraction," in *Proc. of EMNLP*, 2002.

[18] C. Buckley, G. Salton, J. Allan, and A. Singhal, "Automatic query expansion using smart: Trec 3," in *Proc. of 3rd Text REtreival Conference*, 1994, pp. 69–80.

[19] V. Vapnik, *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

[20] K. Church and P. Hanks, "Word association norms, mutual information and lexicography," *Computational Linguistics*, vol. 16, pp. 22–29, 1991.

[21] Z. Bar-Yossef and M. Gurevich, "Random sampling from a search engine's index," in *Proceedings of 15th International World Wide Web Conference*, 2006.

[22] F. Keller and M. Lapata, "Using the web to obtain frequencies for unseen bigrams," *Computational Linguistics*, vol. 29(3), pp. 459–484, 2003.

[23] M. Lapata and F. Keller, "Web-based models ofr natural language processing," *ACM Transactions on Speech and Language Processing*, vol. 2(1), pp. 1–31, 2005.

[24] 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*, 2005, pp. 1297–1304.

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

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

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

[28] 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*, vol. 16, no. 11, pp. 1424–1440, 2004.

[29] Z. Harris, "Distributional structure," *Word*, vol. 10, pp. 146–162, 1954.

[30] J. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," *Advances in Large Margin Classifiers*, pp. 61–74, 2000.

[31] P. Gill, W. Murray, and M. Wright, *Practical optimization*. Academic Press, 1981.

[32] H. Rubenstein and J. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, pp. 627–633, 1965.

[33] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," *ACM Transactions on Information Systems*, vol. 20, pp. 116–131, 2002.

[34] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring semantic similarity between words using web search engines," in *Proc. of WWW '07*, 2007, pp. 757–766.

[35] M. Strube and S. P. Ponzetto, "Wikirelate! computing semantic relatedness using wikipedia," in *Proc. of AAAI'06*, 2006, pp. 1419–1424.

[36] A. Gledson and J. Keane, "Using web-search results to measure word-group similarity," in *Proc. of COLING'08*, 2008, pp. 281–288.

[37] Z. Wu and M. Palmer, "Verb semantics and lexical selection," in *Proc. of ACL'94*, 1994, pp. 133–138.

[38] C. Leacock and M. Chodorow, "Combining local context and wordnet similarity for word sense disambiguation," *WordNet: An Electronic Lexical Database*, vol. 49, pp. 265–283, 1998.

[39] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *Proc. of the International Conference on Research in Computational Linguistics ROCLING X*, 1997.

[40] M. Jarmasz, "Roget's thesaurus as a lexical resource for natural language processing," University of Ottowa, Tech. Rep., 2003.

[41] V. Schickel-Zuber and B. Faltings, "Oss: A semantic similarity function based on hierarchical ontologies," in *Proc. of IJCAI'07*, 2007, pp. 551–556.

[42] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Pasca, and A. Soroa, "A study on similarity and relatedness using distributional and wordnet-based approaches," in *Proc. of NAACL-HLT'09*, 2009.

[43] G. Hirst, , and D. St-Onge, "Lexical chains as representations of context for the detection and correction of malapropisms." *WordNet: An Electronic Lexical Database*, pp. 305–?32, 1998.

[44] T. Hughes and D. Ramage, "Lexical semantic relatedness with random graph walks," in *Proc. of EMNLP-CoNLL'07*, 2007, pp. 581–589.

[45] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *Proc. of IJCAI'07*, 2007, pp. 1606–1611.

[46] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka, "Polyphonet: An advanced social network extraction system," in *Proc. of 15th International World Wide Web Conference*, 2006.

[47] A. Bagga and B. Baldwin, "Entity-based cross document coreferencing using the vector space model," in *Proc. of 36th COLING-ACL*, 1998, pp. 79–85.

Danushka Bollegala received his BS, MS and PhD degrees from the University of Tokyo, Japan in 2005, 2007, and 2009. He is currently a research fellow of the Japanese society for the promotion of science (JSPS). His research interests are natural language processing, Web mining and artificial intelligence.

Yutaka Matsuo is an associate professor at Institute of Engineering Innovation, the University of Tokyo, Japan. He received his BS, MS, and PhD degrees from the University of Tokyo in 1997, 1999, and 2002. He joined National Institute of Advanced Industrial Science and Technology (AIST) from 2002 to 2007. He is interested in social network mining, text processing, and semantic web in the context of artificial intelligence research.

Mitsuru Ishizuka (M'79) is a professor at Graduate School of Information Science and Technology, the University of Tokyo, Japan. He received his BS and PhD degrees in electronic engineering from the University of Tokyo in 1971 and 1976. His research interests include artificial intelligence, Web intelligence, and lifelike agents.