IEICE TRANS. INF. & SYST., VOL.E93–D, NO.1 JANUARY 2010

PAPER

Measuring the degree of Synonymy between Words using Relational Similarity between Word Pairs as a Proxy

Danushka BOLLEGALA[†], **Yutaka MATSUO**[†], ***Nonmembers,***
and* Mitsuru ISHIZUKA**[†], ***Member

SUMMARY Two types of similarities between words have been studied in the natural language processing community: *synonymy* and *relational similarity.* A high degree of similarity exist between synonymous words. On the other hand, a high degree of relational similarity exists between analogous word pairs. We present and empirically test a hypothesis that links these two types of similarities. Specifically, we propose a method to measure the degree of synonymy between two words using relational similarity between word pairs as a proxy. Given two words, first, we represent the semantic relations that hold between those words using lexical patterns. We use a sequential pattern clustering algorithm to identify different lexical patterns that represent the same semantic relation. Second, we compute the degree of synonymy between two words using an inter-cluster covariance matrix. We compare the proposed method for measuring the degree of synonymy against previously proposed methods on the Miller-Charles dataset and the WordSimilarity-353 dataset. Our proposed method outperforms all existing Web-based similarity measures, achieving a statistically significant Pearson correlation coefficient of 0.867 on the Miller-Charles dataset.
key words: *synonymy, attributional similarity, relational similarity, Miller-Charles dataset, WordSimilarity-353 dataset*

1. Introduction

Two broad types of similarities between words have been studied in the Natural Language Processing (NLP) community [1]: *synonymy* and *analogy.* If two words can be inter-changed in numerous contexts without altering the meaning of the context, then those two words are regarded as synonyms. For example, the two words *teacher* and *instructor* are synonymous. Measuring the degree of synonymy between two words is an important step in numerous tasks in NLP such as thesauri generation [2], information retrieval (IR) [3] synonym extraction [4], and word sense disambiguation (WSD) [5]. Synonyms demonstrate a high degree of semantic similarity, whereas the latter encompasses a broader class of words including synonyms, hypernyms, meronyms and even antonyms in some cases.

Relationall similarity can be defined as the correspondence that exist between the semantic relations that are implicitly expressed by two pairs of words. For example, let us consider the two word pairs (*lion*, *cat*) and (*ostrich*, *bird*). In the first word pair, the semantic relation **X** *is a large* **Y** exists between the first word (i.e. *lion*), and the second word (i.e. *cat*). Here, we use the place holder variables **X** and **Y** respectively to denote the first and the second word in a word pair. Likewise, in the second word pair, we can also observe the semantic relation ***X*** *is a large* ***Y*** between its first word (i.e. *ostrich*) and its second word (i.e. *bird*). Therefore, it is considered that the two word pairs, (*lion*, *cat*) and (*ostrich*, *bird*) are relationally similar. A high degree of relational similarity can be observed between analogous word pairs. Relational similarity has found to be useful in detecting verbal analogies and the semantic relations that exist between nouns and their modifiers [1], [6]. Unlike synonymy, which considers the similarity of two words, in relational similarity considers the semantic relations that exist between the two words in each word pair, and not on similarity between those words.

We propose a method to measure the degree of synonymy, $Sim_{syn}(A, B)$, between two given words A and B using the relational similarity, $Sim_{rel}((A, B), (C, D))$, between the word pair (A, B) and another word pair (C, D). Here, C and D are synonyms. For example consider measuring the degree of synonymy between *wire* and *pipe.* Here, we are interested in computing the value of $Sim_{syn}(\text{wire}, \text{pipe})$. Typically, wires carry (conduct) electricity, whereas pipes carry liquids such as water or oil. Moreover, both wires and pipes tend to be *cylindrical in shape*, have *longer lengths* and *shorter radii.* Therefore, we can expect some degree of synonymy between the two words *wire* and *pipe.* As an alternative to computing this value directly, we can compare the word pair (*wire*, *pipe*) to a synonymous word pair such as (*car*, *automobile*) using relational similarity. Both cars and automobiles in general are used to carry (transport) people or goods from one point to another. The semantic relation *both* ***X*** *and* ***Y*** *are used to carry* exists between the two words in word pairs (*wire*, *pipe*) and (*car*, *automobile*). We conjecture that the relational similarity, $Sim_{rel}((\text{wire}, \text{pipe}), (\text{car}, \text{automobile}))$, can be used as a proxy for the degree of synonymy, $Sim_{syn}(\text{wire}, \text{pipe})$. Next, we formally define this intuition in the form of a hypothesis. Because this hypothesis *links* the two concepts of degree of synonymy and relational similarity, we name it the *Linking Hypothesis.*

Linking Hypothesis: The degree of synonymy be-

[†]The University of Tokyo
DOI: 10.1587/transinf.E93.D.1

tween two given words A and B, $Sim_{syn}(A, B)$, can be measured as the relational similarity, $Sim_{rel}((A, B), (C, D))$, where C and D are some synonymous words.

In Section 5, we empirically justify the linking hypothesis using two benchmark datasets that have been used extensively in previous work on similarity. Moreover, we show that the proposed similarity measure demonstrates a high degree of correlation with the human notion of synonymy. In fact, the proposed method reports the best correlation coefficient among all previously proposed Web-based similarity measures.

2. Relational Similarity between Word Pairs

We must address two important problems to be able to compute the degree of synonymy using the linking hypothesis: (1) We must be able to measure the relational similarity between two word pairs (A, B) and (C, D); and (2) We must be able to estimate the degree of synonymy using a relational similarity measure. In this Section, we focus on the first problem. In Section 3, we tackle the second problem.

To accurately compute the relational similarity between two word pairs (A, B) and (C, D), we must overcome three challenges. First, we must extract the semantic relations that are implied by a word pair. In our previous example, we must extract the semantic relation ***X*** *is a large* ***Y*** that is implied by the word pair (lion,cat). For this purpose in Section 2.1, we present a method to extract lexical patterns that express the semantic relations that exist between the two words in a word pair. Second, a semantic relation can be expressed using multiple lexical patterns. For example, the lexical patterns ***X*** *is a large* ***Y*** and *large* ***Y****'s such as* ***X****s* both indicate the same semantic relation implied by the word pair (lion, cat). It is important to group different lexical patterns that express the same semantic relation to accurately compute relational similarity. Moreover, by grouping different lexical patterns that express the same semantic relation, we can overcome the data sparseness problem. For this purpose, we introduce a sequential pattern clustering algorithm in Section 2.2. Third, we must compute the relational similarity between the two word pairs using the set of clusters produced in Section 2.2. For this purpose we propose a relational similarity measure defined using the inter-cluster correlation matrix in Section 2.3.

2.1 Extracting Lexical Patterns

To express the semantic relations that exist between the two words in a given word pair, we extract numerous lexical patterns from contexts in which those two words co-occur. For this purpose, we use the subsequence pattern extraction algorithm [7]. Next, we briefly outline the steps of this pattern extraction method.

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

Fig. 1 A snippet returned for the query *"ostrich * * * * * bird"*.

Given two words A and B, we query a web search engine using the wildcard query *"A * * * * * B"* and download snippets. Here, snippets refer to the short texts returned by most web search engines by extracting the local context of the query in a web page. Using snippets for pattern extraction is efficient because it obviates the need to download the web pages, which can be time consuming if there are lots of search results. The "*" operator matches one word or none in a web page. Therefore, our wildcard query retrieves snippets in which A and B co-occur within a window of seven words. We attempt to approximate the local context of two words using wildcard queries. For example, Figure 1 shows a snippet retrieved for the query *"ostrich * * * * * bird"*.

For a snippet S, retrieved for a word pair (A, B), first, we replace the two words A and B, respectively, with two place holder variables **X** and **Y**. Next, we generate all subsequences of words from S 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 consecutive words. However, we do not allow word skips of more than g number of consecutive words. Moreover, the total length of all word skips in a subsequence must 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***.

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

The parameters L, g, G and N are set respectively to 5, 2, 2, and 4 as recommended in [7]. The above-mentioned 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 word skips enables us to capture relations between distant words in a snippet. The *prefixspan* algorithm [8] is used to generate subsequences from a text snippet. For example, some of the patterns extracted form the snippet shown in Figure 1 are: ***X****, a large* ***Y***, ***X*** *a flightless* ***Y***, and ***X****, large* ***Y*** *lives*.

2.2 Clustering Lexical Patterns

A semantic relation can be expressed using more than one lexical pattern. By grouping the semantically related patterns, we can reduce the data sparseness, thereby accurately measure the relational similarity between two word pairs. We use the sequential pattern clustering algorithm [7] for this purpose. We use the distributional hypothesis [9] to find semantically related lexical patterns. The distributional hypothesis states that words that occur in the same context have similar meanings. If two lexical patterns are similarly distributed over a set of word pairs, then from the distributional hypothesis it follows that those two patterns must be semantically similar.

We represent a pattern p by a vector p in which the i-th element is the co-occurrence frequency $f(A_i, B_i, p)$ of p in a word pair (A_i, B_i). Given a set $\mathcal{P}$ of patterns and a similarity threshold θ, Algorithm 1 returns a set of clusters of similar patterns. First, the function *SORT* sorts the patterns in the descending order of their total occurrences in all word pairs. The total occurrences of a pattern p is defined as $\mu(p)$, and is given by,

$$\mu(p) = \sum_{(A,B) \in \mathcal{W}} f(A, B, p). \tag{1}$$

Here, $\mathcal{W}$ is the set of word pairs. Then the outer for-loop (starting at line 3), repeatedly takes a pattern p_i from the ordered set $\mathcal{P}$, and in the inner for-loop (starting at line 6), finds the cluster, c^* $(\in \mathcal{C})$ that is most similar to p_i. Similarity between p_i and the cluster centroid c_j is computed using cosine similarity. The centroid vector c_j of cluster c_j is defined as the vector sum of all pattern vectors for patterns in that cluster (i.e. $c_j = \sum_{p \in c_j} p$). If the maximum similarity exceeds the threshold θ, we append p_i to c^* (line 14). Here, the operator $\oplus$ denotes vector addition. Otherwise, we form a new cluster $\{p_i\}$ and append it to $\mathcal{C}$, the set of clusters. The parameter θ $(\in [0, 1])$ determines the *purity* of the formed clusters and is set experimentally in Section 5.1.

Time complexity of Algorithm 1 results from two factors. First, the initial sort operation requires $\mathcal{O}(n \log n)$ for a set of n lexical patterns. Second, if the average number of clusters is $|C|$, then we require $|C|$ comparisons against the existing clusters to determine the cluster with the maximum similarity to a lexical pattern. Therefore, the time complexity for the cluster assignment step is $\mathcal{O}(|C|n)$. In the worst case scenario (when each pattern is in its own singleton cluster), $|C| = n$. Therefore, the worst-case time complexity of the overall sequential clustering algorithm is $\mathcal{O}(n \log n + n^2)$, which becomes $\mathcal{O}(n^2)$ for large n.

Algorithm 1 Sequential pattern clustering algorithm.

Input: patterns $\mathcal{P} = \{p_1, \ldots, p_n\}$, threshold θ.
Output: A set $\mathcal{C}$ of pattern clusters.

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

Moreover, sorting the patterns by their total word pair frequency prior to clustering ensures that most common relations in the dataset are clustered first and outliers get attached at the end.

2.3 Measuring Relational Similarity

After all patterns are clustered using Algorithm 1, we compute the (i, j) element of the inter-cluster correlation matrix Λ (denoted as $\Lambda_{(i,j)}$) as the inner-product between the centroid vectors $\mathbf{c_i}$ and $\mathbf{c_j}$ of the corresponding clusters i and j. Therefore, if there are m number of clusters in $\mathcal{C}$, Λ will be an $m \times m$ square symmetric matrix. We compute the relational similarity, $Sim_{rel}((A, B), (C, D))$, between two word pairs (A, B) and (C, D) as follows,

$$\begin{aligned} &Sim_{rel}((A, B), (C, D)) = \\ &\sum_{p_i, p_j \in P} [f(A, B, p_i) \times f(C, D, p_i) \times \Lambda_{(i,j)} \times \\ &f(A, B, p_j) \times f(C, D, p_j)]. \end{aligned} \tag{2}$$

Equation 2 can be understood as the weighted outer product between the two pattern frequency vectors corresponding to the two word pairs (A, B) and (C, D). Specifically, we multiply the co-occurrences of two patterns p_i and p_j with the two word pairs (A, B) and (C, D). The term $\Lambda_{(i,j)}$ denotes the correlation between the two clusters $\mathbf{c}_i$ and $\mathbf{c}_j$, that subsumes respectively p_i and p_j. For notational convenience we use the same indexes for pattern as well as for the cluster that they belong to.

3. Computing the Degree of Synonymy

Following the linking hypothesis, we define the degree of synonymy, $Sim_{syn}(A, B)$, between two words A and B as follows,

$$Sim_{syn}(A, B) = \frac{1}{|\mathcal{T}|} \sum_{(C,D)\in\mathcal{T}} Sim_{rel}((A, B), (C, D)). \quad (3)$$

Here, $\mathcal{T}$ is a set of synonymous word pairs selected from the WordNet as described in the next paragraph, and $|\mathcal{T}|$ denotes the number of word pairs in $\mathcal{T}$. Intuitively, Equation 3 compares the semantic relations that exist between A and B (expressed using lexical patterns), against the semantic relations that typically exist between synonymous words. If the semantic relations that exist between A and B are highly similar (i.e. resulting in a high relational similarity) to that between synonymous words, then we can infer that A and B themselves must also be synonymous according to the linking hypothesis.

To construct the set $\mathcal{T}$, we select synonymous words from WordNet synsets. A synset is a set of synonymous words assigned to a particular sense of a word in WordNet. To determine the number of synonymous word pairs (i.e. $\mathcal{T}$) to be used in our model, in our preliminary experiments, we tried different numbers of randomly selected synsets in the range $[1000, 5000]$ from the WordNet. Using each set of word pairs as $\mathcal{T}$, we measured the average similarity among a set of 500 synonymous word pairs, which we set aside as development data. We found that the average similarity given by Equation 3 attains a maximum and was invariant when 2000 or more synsets were used. Therefore, we use 2000 synsets of nouns from WordNet as $\mathcal{T}$ in the remainder of the experiments described in this paper. To compute the semantic similarity between two words A and B using Equation 3, we must compute the relational similarity between word pair (A, B) and all word pairs in the training set $\mathcal{T}$. Therefore, this procedure requires $\mathcal{O}(|\mathcal{T}|)$ complexity which is linear in the size of the training dataset.

For example, consider computing the semantic similarity between the two words *food* and *fruit*. We first extract lexical patterns such as ***Y****s are healthy* ***X*** from Web snippets and then apply the clustering Algorithm 1 to cluster the lexical patterns. Next, we use Equation 3 and compare the word pair (*food*,*fruit*) against synonymous word pairs in $\mathcal{T}$. The degree of synonymy between those two words is computed to be 0.94 as shown in Table 1.

4. Benchmark Datasets

Evaluating a measure of synonymy is difficult because the notion of synonymy is subjective. Miller-Charles [10] dataset (hereon referred to as the MC dataset) has been frequently used to benchmark semantic similarity measures. This dataset contains 30 word pairs rated on a scale from 0 (no similarity) to 4 (perfect synonymy) by a group of 38 human subjects. However, most previous work have used only 28 pairs for evaluation because one word was not registered in WordNet 3.0. Consequently, we follow those previous work and use only 28 word pairs such that we can directly compare our results with previous work [1]. In addition to MC dataset, we also evaluate on the WordSimilarity-353 [11] dataset (hereon referred to as the WS dataset). In contrast to MC dataset which has only 30 word pairs, WS dataset contains 353 word pairs. Each pair has 13-16 human judgments, which are averaged for each pair to produce a single relatedness score. The degree of correlation between the human ratings in a benchmark dataset and the similarity scores produced by a similarity measure, is considered as a measurement of how well the similarity measure captures the notion of synonymy possessed by humans. Following previous work we use Pearson and Spearman correlation coefficients respectively, to evaluate on MC and WS datasets.

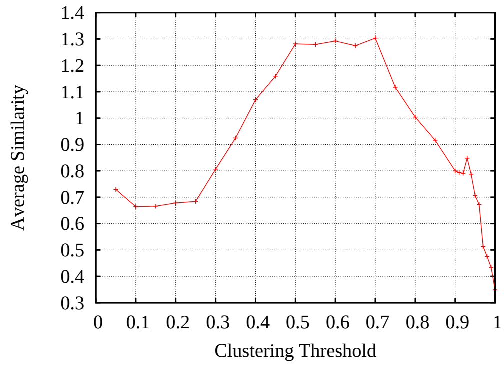

Fig. 2 Average similarity vs. clustering threshold θ

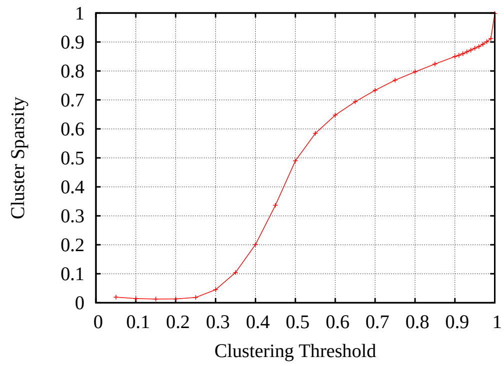

Fig. 3 Sparsity vs. clustering threshold θ

5. Experiments

5.1 Parameter Tuning

We use the set of 2000 synonymous word pairs selected from WordNet synsets as described in Section 3 to determine the optimum value of the clustering threshold, θ, in Algorithm 1. First, we use the YahooBOSS

API† and download 1000 snippets for each of those word pairs. From those snippets, the sequential pattern extraction algorithm (Section 2.1) extracts $5,238,637$ unique patterns. However, only $1,680,914$ of those patterns occur more than twice. Low frequency patterns are noisy and unreliable. Therefore, we selected patterns that occur at least 10 times in the snippet collection. After this preprocessing we obtain $140,691$ unique lexical patterns. The remainder of the experiments described in the paper use those patterns.

Next, we vary the value of theta θ from 0 to 1, and use Algorithm 1 to cluster the extracted set of patterns. We compute the inter-cluster correlation matrix Λ as described in Section 2.2 and use Equations 2 and 3 to compute the average similarity between words in $\mathcal{T}$, the set of 2000 synonymous word pairs selected from the WordNet synsets. Finally, $\hat{\theta}$, the optimal value of the clustering threshold θ, is set to the value that produces a set of clusters that maximizes the average similarity between words in $\mathcal{T}$ as follows,

$$\hat{\theta} = \mathrm{argmax}_{\theta\in[0,1]} \left(\frac{1}{|\mathcal{T}|}\sum\nolimits_{(A,B)\in\mathcal{T}} Sim_{syn}(A,B)\right) \quad (4)$$

Average similarity scores for various θ values are shown in Figure 2. From Figure 2, 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 Figure 3 where we plot the sparsity of the set of clusters (i.e. the ratio between singletons to total clusters) against the threshold θ. From Figure 3, we see that 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 1.303 is obtained with $\theta = 0.7$, corresponding to $17,015$ total clusters out of which $12,476$ are singletons with exactly one pattern (sparsity $= 0.733$). For the remainder of the experiments in this paper we set θ to this optimal value and use the corresponding set of clusters to compute relational similarity by Equation 2. Similarity scores computed using Equation 2 can be greater than 1 (see Figure 2) because of the terms corresponding to the non-diagonal elements in Λ. We do not normalize the similarity scores to $[0,1]$ range in our experiments because the evaluation metrics we use are insensitive to linear transformations of similarity scores.

5.2 Correlation with Human Ratings

Table 1 compares the proposed method against human ratings in the MC dataset, and previously proposed Web-based similarity measures: Jaccard, Dice, Overlap, PMI [12], Normalized Google Distance (NGD) [13], Sahami and Heilman (SH) [3], co-occurrence double checking model (CODC) [14], and support vector machine-based (SVM) approach [12]. The bottom row of Table 1 shows the Pearson correlation coefficient of similarity scores produced by each algorithm with MC. All similarity scores, except for the human-ratings in the MC dataset, are normalized to $[0,1]$ range for the ease of comparison. It is noteworthy that the Pearson correlation coefficient is invariant under a linear transformation. All similarity scores shown in Table 1 except for the proposed method are taken from the original published papers.

The highest correlation is reported by the proposed semantic similarity measure. The improvement of the proposed method is statistically significant (confidence interval $[0.73, 0.93]$) against all the similarity measures compared in Table 1 except against the SVM approach. From Table 1 we see that measures that use contextual information from snippets (e.g. SH, CODC, SVM, and proposed) outperform the ones that use only co-occurrence statistics (e.g. Jaccard, overlap, Dice, PMI, and NGD) such as page-counts. This is because similarity measures that use contextual information are better equipped to compute the similarity between polysemous words. Although both SVM and proposed methods use lexical patterns, unlike the proposed method, the SVM method does not consider the relatedness between patterns. The superior performance of the proposed method is attributable to its consideration of relatedness of patterns.

Table 2 summarizes the previously proposed WordNet-based semantic similarity measures. Despite the fact that the proposed method does not use manually compiled resources such as WordNet for computing similarity, its performance is comparable to similarity measures that use WordNet. We believe that the proposed method will be useful to compute the degree of synonymy between named-entities for which manually created resources are either incomplete or do not exist.

We evaluate the proposed method using the WS dataset as shown in Table 3. Following previous work, we use Spearman rank correlation coefficient, which does not require ratings to be linearly dependent, for the evaluations on this dataset. Likewise with the MC ratings, we measure the correlation between the similarity scores produced by the proposed method for word pairs in the WS dataset and the human ratings. A higher Spearman correlation coefficient (value=0.504, confidence interval $[0.422, 0.578]$) indicates a better agreement with the human notion of semantic similarity. From Table 3, we can see that the proposed method outperforms a wide variety of similarity measures developed using numerous resources including lexical resources such as WordNet and knowledge sources

†http://developer.yahoo.com/search/boss/

Table 1 Semantic similarity scores on Miller-Charles dataset

Word Pair	MC	Jaccrad	Dice	Overlap	PMI	NGD	SH	CODC	SVM	Proposed
automobile-car	3.920	0.650	0.664	0.831	0.427	0.466	0.225	0.008	0.980	0.918
journey-voyage	3.840	0.408	0.424	0.164	0.468	0.556	0.121	0.005	0.996	1.000
gem-jewel	3.840	0.287	0.300	0.075	0.688	0.566	0.052	0.012	0.686	0.817
boy-lad	3.760	0.177	0.186	0.593	0.632	0.456	0.109	0.000	0.974	0.958
coast-shore	3.700	0.783	0.794	0.510	0.561	0.603	0.089	0.006	0.945	0.975
asylum-madhouse	3.610	0.013	0.014	0.082	0.813	0.782	0.052	0.000	0.773	0.794
magician-wizard	3.500	0.287	0.301	0.370	0.863	0.572	0.057	0.008	1.000	0.997
midday-noon	3.420	0.096	0.101	0.116	0.586	0.687	0.069	0.010	0.819	0.987
furnace-stove	3.110	0.395	0.410	0.099	1.000	0.638	0.074	0.011	0.889	0.878
food-fruit	3.080	0.751	0.763	1.000	0.449	0.616	0.045	0.004	0.998	0.940
bird-cock	3.050	0.143	0.151	0.144	0.428	0.562	0.018	0.006	0.593	0.867
bird-crane	2.970	0.227	0.238	0.209	0.516	0.563	0.055	0.000	0.879	0.846
implement-tool	2.950	1.000	1.000	0.507	0.297	0.750	0.098	0.005	0.684	0.496
brother-monk	2.820	0.253	0.265	0.326	0.623	0.495	0.064	0.007	0.377	0.265
crane-implement	1.680	0.061	0.065	0.100	0.194	0.559	0.039	0.000	0.133	0.056
brother-lad	1.660	0.179	0.189	0.356	0.645	0.505	0.058	0.005	0.344	0.132
car-journey	1.160	0.438	0.454	0.365	0.205	0.410	0.047	0.004	0.286	0.165
monk-oracle	1.100	0.004	0.005	0.002	0.000	0.579	0.015	0.000	0.328	0.798
food-rooster	0.890	0.001	0.001	0.412	0.207	0.568	0.022	0.000	0.060	0.018
coast-hill	0.870	0.963	0.965	0.263	0.350	0.669	0.070	0.000	0.874	0.356
forest-graveyard	0.840	0.057	0.061	0.230	0.495	0.612	0.006	0.000	0.547	0.442
monk-slave	0.550	0.172	0.181	0.047	0.611	0.698	0.026	0.000	0.375	0.243
coast-forest	0.420	0.861	0.869	0.295	0.417	0.545	0.060	0.000	0.405	0.150
lad-wizard	0.420	0.062	0.065	0.050	0.426	0.657	0.038	0.000	0.220	0.231
cord-smile	0.130	0.092	0.097	0.015	0.208	0.460	0.025	0.000	0	0.006
glass-magician	0.110	0.107	0.113	0.396	0.598	0.488	0.037	0.000	0.180	0.050
rooster-voyage	0.080	0.000	0.000	0.000	0.228	0.487	0.049	0.000	0.017	0.052
noon-string	0.080	0.116	0.123	0.040	0.102	0.488	0.024	0.000	0.018	0.000
Correlation	**-**	**0.260**	**0.267**	**0.382**	**0.549**	**0.205**	**0.580**	**0.694**	**0.834**	**0.867**

Table 2 Comparison with WordNet-based similarity measures.

Method	Correlation
Edge-counting	0.664
Jiang & Conrath [15]	0.848
Lin [4]	0.822
Resnik [5]	0.745
Li et al. [16]	0.891

Table 3 Results on WordSimilarity-353 dataset.

Method	Correlation
WordNet Edges [17]	0.27
Hirst & St-Onge [18]	0.34
Jiang & Conrath [15]	0.34
WikiRelate! [19]	0.19-0.48
Leacock & Chodrow [20]	0.36
Lin [21]	0.36
Resnik [5]	0.37
Proposed	0.50

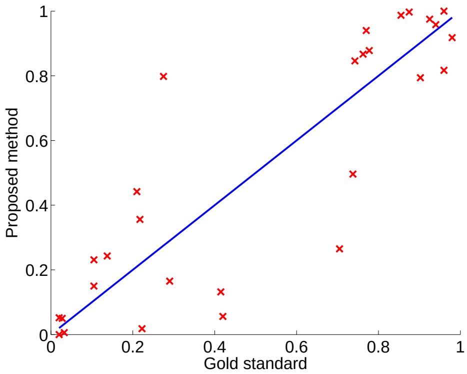

Fig. 4 Correlation with human ratings in the Miller-Charles dataset. (Pearson correlation coefficient = 0.867)

such as Wikipedia (i.e. WikiRelate!). In contrast to the MC dataset which only contains common English words selected from the WordNet, the WS dataset contains word pairs where one or both words are named entities (e.g. (*Maradona*, *football*) and (*Jerusalem*, *Israel*)). Consequently, WordNet-based similarity measures report low performance on WS dataset compared to the MC dataset. On the other hand, the proposed method use snippets retrieved from a web search engine, and is capable of extracting expressive lexical patterns that can explicitly state the relationship between two entities.

Figures 4 and 5 show the correlation between the degree of synonymy values produced by our proposed method against human annotated ratings respectively in the MC dataset and the WS dataset. From those figures we see that the similarity values produced by the proposed method are highly correlated with that in the benchmark datasets. Moreover, from Figure 4 we see that there is a comparatively higher agreement between the two methods for low and high similarity word pairs. We believe that all the above-mentioned experimental results empirically justify the linking hypothesis proposed in this paper.

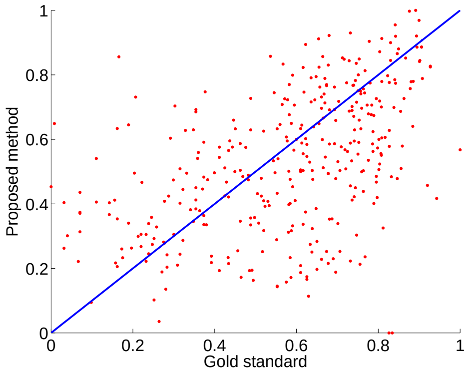

Fig. 5 Correlation with human ratings in the WordSimilarity-353 dataset. (Spearman correlation coefficient = 0.504)

6. Related Work

Given a taxonomy of concepts, a straightforward method to calculate similarity between two words (or concepts) is to find the length of the shortest path connecting the two words in the taxonomy [22]–[24]. 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 [25].

Resnik [5] 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.

Cilibrasi and Vitanyi [13] proposed *Normalized Google Distance* (NGD), which is defined as the normalized information distance between two strings. Unfortunately NGD only uses page-counts of words and ignores the context in which the words appear. Therefore, it produces inaccurate similarity scores when one or both words between which similarity is computed are polysemous. Sahami and Heilman [3] 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. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors.

Chen et al., [14] propose a web-based double-checking model in which, for two words X and Y, snippets are collected from a web search engine. They count the number of X in the snippets for Y, and Y in the snippets for X and combine them non-linearly to compute the similarity between X and Y. This method heavily depends on the search engine's ranking algorithm. Although two words X and Y may be very similar, there is no reason to believe that one can find Y in the snippets for X, or vice versa.

In our previous work [12], we proposed a semantic similarity measure using page counts and snippets retrieved from a Web search engine. To compute the similarity between two words X and Y, we queried a web search engine using the query X *AND* Y and extract lexical patterns that combine X and Y from snippets. A feature vector is formed using frequencies of 200 lexical patterns in snippets and four co-occurrence measures: Dice coefficient, overlap coefficient, Jaccard coefficient and pointwise mutual information. We trained a two-class support vector machine using automatically selected synonymous and non-synonymous word pairs from WordNet. This method reports a Pearson correlation coefficient of 0.837 with Miller-Charles ratings. However, it does not consider the relatedness between patterns.

Zhou et al. [26] extract paraphrases to compare a summary made by a system against a reference summary. Their method is based on the idea that two different phrases of the same meaning might have the same translation in a foreign language. They perform word alignment on a corpus of parallel texts to find phrase translation pairs. However, our method does not require parallel corpora and can be used when such resources are not available.

Nakov and Kozareva [27] use attributional features to improve the measurement of relational similarity between nominals. They extract hypernyms and co-hyponyms from the Web as additional features for classifying semantic relations. Although this work combines both relational and attributional similarities, its goal is different from ours – we use relational similarity to compute the degree of synonymy, whereas they use attributional features to measure relational similarity.

7. Conclusion

We proposed a method to measure the degree of synonymy between two words using the relational similarity between those two words to numerous other synonymous word pairs. For this purpose, we proposed the linking hypothesis that connects the two concepts of synonymy and relational similarities. Using two standard benchmark datasets, we empirically justified the linking hypothesis. Our proposed measure outperformed all existing Web-based similarity measures demonstrating a high level of correlation with human annotated similarity ratings.

References

[1] P.D. Turney, "Similarity of semantic relations," Computational Linguistics, vol.32, no.3, pp.379–416, 2006.
[2] J. Curran, "Ensemble menthods for automatic thesaurus extraction," EMNLP, pp.222–229, 2002.
[3] M. Sahami and T. Heilman, "A web-based kernel function

for measuring the similarity of short text snippets," WWW, pp.377–386, 2006.
[4] D. Lin, "Automatic retreival and clustering of similar words," COLING, pp.768–774, 1998.
[5] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," IJCAI, pp.448–453, 1995.
[6] P. Turney, "Expressing implicit semantic relations without supervision," COLING-ACL, pp.313–320, 2006.
[7] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring the similarity between implicit semantic relations from the web," WWW, pp.651 – 660, 2009.
[8] 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.
[9] Z. Harris, "Distributional structure," Word, vol.10, pp.146–162, 1954.
[10] G. Miller and W. Charles, "Contextual correlates of semantic similarity," Language and Cognitive Processes, vol.6(1), pp.1–28, 1998.
[11] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," ACM TOIS, vol.20, pp.116–131, 2002.
[12] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring semantic similarity between words using web search engines," WWW, pp.757–766, 2007.
[13] R. Cilibrasi and P. Vitanyi, "The google similarity distance," IEEE Transactions on Knowledge and Data Engineering, vol.19, no.3, pp.370–383, 2007.
[14] H. Chen, M. Lin, and Y. Wei, "Novel association measures using web search with double checking," COLING-ACL, pp.1009–1016, 2006.
[15] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," ROCLING, pp.19 – 33, 1998.
[16] 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.
[17] M. Jarmasz, "Roget's thesaurus as a lexical resource for natural language processing," Master's thesis, University of Ottawa, 1993.
[18] G. Hirst and D. St-Onge, "Lexical chains as representations of context for the detection and correction of malapropisms." WordNet: An Electronic Lexical Database, 1997.
[19] M. Strube and S.P. Ponzetto, "Wikirelate! computing semantic relatedness using wikipedia," AAAI, pp.1419–1424, 2006.
[20] C. Leacock and M. Chodorow, Combining Local Context and WordNet Similarity for Word Sense Identification, MIT, 1998.
[21] D. Lin, "An information-theoretic definition of similarity," ICML, pp.296–304, 1998.
[22] 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.
[23] R.E. Menedez-Mora and R. Ichise, "Toward simulating the human way of comparing concepts," IEICE Transactions, vol.E94-D(7), pp.1419 – 1429, 2011.
[24] G. Pirró and N. Seco, "Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content," Move to Meaningful Internet Systems (OTM), pp.1271 – 1288, 2008.
[25] V. Schickel-Zuber and B. Faltings, "OSS: A semantic similarity function based on hierarchical ontologies," IJCAI, pp.551–556, 2007.
[26] L. Zhou, C.Y. Lin, D.S. Munteanu, and E. Hovy, "Paraeval: Using paraphrases to evaluate summaries automatically," HLT-NAACL, pp.447 – 454, 2006.
[27] P. Nakov and Z. Kozareva, "Combining relational and attributional similarity for semantic relation classification," RANLP, pp.323 – 330, 2011.

Danushka Bollegala received his BS, MS and PhD degrees from the University of Tokyo, Japan in 2005, 2007, and 2009. He is currently an assistant professor at the Graduate School of Information Science and Technology, the University of Tokyo. 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 is interested in social network mining, and text processing in the context of artificial intelligence research.

Mitsuru Ishizuka 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.