A Supervised Ranking Approach for Detecting Relationally Similar Word Pairs

Danushka Bollegala*
*Graduate School of Information Sciences,
The University of Tokyo,
7-3-1 Hongo Bunkyo-ku, Tokyo, Japan.
Email: danushka@iba.t.u-tokyo.ac.jp

***Abstract*—The similarity between the semantic relations that exist between two word pairs is defined as their *relational similarity*. For example, the semantic relation, is a large holds between the words in the word pair (lion, cat) and (ostrich, bird), because *lion* is a large cat, and *ostrich* is the largest living *bird* on earth. Consequently, the two word pairs, (lion, cat) and (ostrich, bird), are considered to be relationally similar. A high degree of relational similarity can be observed between analogous pairs of words. Measuring the relational similarity between word pairs is important in numerous natural language processing tasks such as solving word analogy questions, classifying noun-modifier relations and disambiguating word senses. We propose a supervised ranking-based method to detect relationally similar word pairs to a given word pair using information retrieved from a Web search engine. First, each pair of words is represented by a vector of automatically extracted lexical patterns. Then a ranking Support Vector Machine is trained to recognize word pairs with similar semantic relations to a given word pair. To train and evaluate the proposed method, we use a benchmark dataset that contains 374 SAT multiple-choice word-analogy questions. To represent the relations that exist between two word pairs, we experiment with 11 different feature functions, including both symmetric and asymmetric feature functions. Our experimental results show that the proposed ranking-based approach outperforms several previously proposed relational similarity measures on this benchmark dataset, achieving an SAT score of 46.9.**

***Keywords*—relational similarity, ranking SVMs**

I. Introduction

Relational similarity can be defined as the correspondence between semantic relations. In this paper, we consider semantic relations that exist between two words. For example, the semantic relation, **X** *is a large* **Y** holds between the words in the word pair (lion, cat) and (ostrich, bird), because lion is a large cat, whereas ostrich is a large bird. Here, we use variables **X** and **Y** as placeholders for words between which a relation exists. Consequently, the two word pairs, (lion, cat) and (ostrich, bird), are considered to be relationally similar. If four words A, B, C, and D form a proportional analogy $A : B :: C : D$, then we can observe a high degree of relational similarity between the two word pairs (A, B) and (C, D).

To accurately measure the relational similarity between two word pairs, we must overcome several challenges. First, the relations themselves are only implicitly stated by a word pair. We must first extract the relation that exists between the two words in each of the word pairs. For example, from the word pair (ostrich, bird), we must first identify the relation **X** *is a large* **Y**. Second, there might exist more than one relation between two words. For example, in addition to the relation **X** *is a large* **Y**, there exists the relation **X** *is a flightless* **Y** between the two words ostrich and bird. Third, we must identify how much each relation contributes to the relational similarity between two word pairs. For example, a general relation such as **X** *and* **Y** which holds between many words might contribute less towards relational similarity, whereas a specific relation such as **X** *is a large* **Y** might contribute more. However, it is not known a priori as to how much each relation contributes to the relational similarity between two given word pairs. In this paper, we follow a supervised machine learning approach to measure the relational similarity between two given word pairs.

We model this problem as a ranking problem in which given a set of word pairs, $\mathcal{S}$, we rank this set according to their relational similarity with another word pair (A, B). Following previous work on relational similarity, we represent the implicit relations that exist between two words using lexical patterns. Specifically, we use the method proposed by Bollegala et al. [1] to extract lexical patterns to represent semantic relations that exist between two given words using text-snippets returned from a Web search engine. Snippets provide useful information about the relations that hold between words. For example, Google[1] returns the snippet *...the ostrich is the largest bird in the world and can be found in South Africa...* for the conjunctive query *ostrich AND bird*. This snippet alone suggests that ostrich is a large bird. We automatically extract lexical patterns that describe the relations implied by the two words in a word pair and compute the relational similarity using a machine learning approach. Moreover, by using snippets, we can obviate the need to download Web pages which can be time consuming if those pages are large.

Relational similarity is a dynamic phenomenon. In particular, relations between named entities change over time and across domains. Therefore, it is costly or even impossible to manually update language resources to reflect those changes. The proposed method does not require language resources such as taxonomies or dictionaries which makes it attractive when measuring relational similarity between named entities. In

[1] http://google.com

Question: Ostrich is to Bird as:
a. Cub is to Bear
b. *Lion is to Cat*
c. Ewe is to Sheep
d. Turkey is to Chicken
e. Jeep is to Truck

Fig. 1. An SAT word analogy question. The correct answer is (b).

comparison to attributional similarity (i.e. the correspondence between attributes that exist in two objects), which has been studied extensively for its numerous applications [2], the problem of measuring relational similarity has received less attention. In order to measure relational similarity between two pairs of words, one must first identify the relations that hold between the two words in each pair and then compare the relations between the pairs.

Scholastic Aptitude Test (SAT) word analogy questions have been used for evaluating relational similarity measures. An SAT word analogy question consists of a word pair (which we designate as the *question* word pair) and five candidate answer word pairs (which we designate as the *candidate* word pairs). Only one of the candidate word pairs is analogous to the source word pair in an SAT word analogy question. An examinee is required to select the analogous word pair to the question word pair. An example is shown in Figure 1. The relatively low average human score of 57% reported for the SAT word analogy questions indicates that detecting the correct word pair among a given set of candidates is difficult for even humans.

Relational similarity measures are useful for numerous tasks in natural language processing such as classification of semantic relations in noun-modifier pairs, word sense disambiguation (WSD) and automatic thesaurus generation. Noun-modifier pairs such as *flu virus*, *storm cloud*, *expensive book*, etc. are frequent in English language. In fact, WordNet contains more than $26,000$ noun-modifier pairs. Natase and Szpakowicz [3] classified noun-modifiers into five classes according to the relations between the noun and the modifier. Turney [4] used a relational similarity measure to compute the similarity between noun-modifier pairs and classify them according to the semantic relations that hold between a noun and its modifier.

We model the problem of solving word analogy questions as a one of supervised ranking. Specifically, given a set candidates of word pairs (i.e. such as the candidate answers of a SAT question), we propose a machine learning method to detect the *most relationally similar* word pair to the question. We use lexical patterns as features to train a ranking support vector machine.

II. Related Work

Turney et al. [5] combined 13 independent modules by considering the weighted sum of the outputs of each individual module to solve SAT analogy questions. The best performing individual module was based on Vector Space Model (VSM). In the VSM approach to measuring relational similarity [6], first a vector is created for a word-pair (X, Y) by counting the frequencies of various lexical patterns containing *X* and *Y*. In their experiments they used 128 manually created patterns such as "*X* of *Y*", "*Y* of *X*", "*X* to *Y*" and "*Y* to *X*". These patterns are then used as queries to a search engine and the number of hits for each query is used as elements in a vector to represent the word pair. Finally, the relational similarity is computed as the cosine of the angle between the two vectors representing each word-pair. This VSM approach achieves a score of 47% on college-level multiple-choice SAT analogy questions. A SAT analogy question consists of a target word-pair and five choice word-pairs. The choice word-pair that has the highest relational similarity with the target word-pair in the question is selected by the system as the correct answer.

Turney [4] proposes Latent Relational Analysis (LRA) by extending the VSM approach in three ways: a) lexical patterns are automatically extracted from a corpus, b) the Singular Value Decomposition (SVD) is used to smooth the frequency data, and c) synonyms are used to explore variants of the word-pairs. LRA achieves a score of 56% on SAT analogy questions. Both VSM and LRA require a large number of search engine queries to create a vector representing a word-pair. For example, with 128 patterns, VSM approach requires at least 256 queries to compute relational similarity. LRA considers synonymous variants of the given word pairs, thus requiring even more search engine queries. In contrast, our proposed method matches numerous patterns among the text snippets retrieved from a Web search engine for the two words in a word pair. We do not search using each extracted pattern as done in VSM and LRA methods. Therefore, the number of Web search queries does not increase with the number of patterns extracted. This enables us to represent relations using a large number of patterns.

Veale [7] proposed a relational similarity measure based on taxonomic similarity in WordNet. He evaluates the quality of a candidate analogy *A:B::C:D* by looking for paths in WordNet, joining *A* to *B* and *C* to *D*. Then, the relational similarity is computed based on the similarity between the *A:B* paths and *C:D* paths. If the set of WordNet relations that connects A to B and the set of WordNet relations that connects C to D has many relations in common (i.e. a high overlap between the two sets), then the relational similarity between two word pairs (A, B) and (C, D) is high. His method achieves a score of 43 on the SAT word analogy questions. However, the dependence on the WordNet means that this method cannot compute the relational similarity when the word pairs have words that do not appear in the WordNet.

Bollegala et al. [8] proposed a supervised metric learning approach to solve SAT word analogy questions. First, they use SAT questions to induce pairwise distance constraints between word pairs. Next, a Mahalanobis distance metric is learnt from the pairwise constraints. They use the information theoretic metric learning (ITML) algorithm to learn the Mahalanobis distance metric. Evaluations are conducted both on SAT word analogy questions as well as on a novel dataset (ENT dataset) that consists of named entities frequently found on the Web.

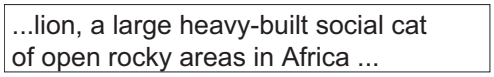

Fig. 2. A snippet returned by Google for the query "lion * * * * * * * cat".

They use lexical patterns to represent the relations that exist between two words. They show that by clustering the lexical patterns that represent the same semantic relation, one can improve the accuracy of the relational similarity measurement. Moreover, a sequential clustering algorithm is presented that can efficiently cluster a large set of lexical patterns.

III. Method

A. Pattern Extraction

We use the subsequence lexical pattern extraction algorithm first proposed by Bollegala et al. [8] for the task of representing the implicit semantic relations that exist between two words. For completeness, we briefly describe this algorithm here. For further details refer the original paper.

To identify the implicit relations between two words **X** and **Y**, we first query a web search engine using the phrasal query "**X** * * * * * * * **Y**". Here, the wildcard operator "*" would match any word or nothing. This query retrieves snippets that contain both **X** and **Y** within a window of 7 words. For example, Google returns the snippet shown in Fig.2 for the word pair (lion, cat). We use *PrefixSpan* (i.e., prefix-projected sequential pattern mining) [9] algorithm to extract frequent subsequences from snippets that contain both **X** and **Y**. PrefixSpan extracts all word subsequences which occur more than a specified frequency in snippets. We select subsequences that contain both query words (eg. lion and cat) and replace the query words respectively with variables **X** and **Y** to construct lexical patterns. For example, some of the patterns extracted by the proposed algorithm from the snippet in Figure 2 are **X** *a large* **Y**, **X** *a large* **Y** *of* and **X** *, a large social* **Y**. PrefixSpan algorithm is particularly attractive for the current task because it can efficiently extract a large number of lexical patterns. Moreover, its ability to skip words when creating patterns enables us to capture relations between words that appear at a distance in snippets.

B. Pattern Selection

We used the pattern extraction algorithm described in section III-A to extract lexical patterns for 374 SAT multiple-choice analogy questions. This dataset was first proposed by Turney and Littman [6] as a benchmark dataset to evaluate relational similarity measures. Generally, there are six word pairs in each question (i.e. one word pair for the question and five choices) which amounts to 2176 (cf. some questions have less than 5 candidates) word-pairs. For each word pair, using Yahoo BOSS Search API[2] we download 7000 snippets on average. Yahoo BOSS Search API allows only 1000 snippets to be retrieved for a single query. To overcome this limitation, we issue multiple contextual queries by varying the number of asterisks between the two words in a query. This process is repeated with the two words inter-changed. Finally, duplicate snippets are removed from the retrieved search results.

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

TABLE I
MOST FREQUENT PATTERNS IN THE CORPUS.

Rank	Freq.	Pattern	Rank	Freq.	Pattern
1	272892	**Y** and **X**	2	240802	**X** and **Y**
3	91720	**X** of **Y**	4	78125	**Y** or **X**
5	78011	**X** the **Y**	6	74428	**Y** to **X**
7	71260	**X** or **Y**	8	65470	**Y** of **X**
9	63986	**X** to **Y**	10	55062	**X** **Y** s

In addition to the snippets, we also select the titles of pages (also returned by the Yahoo BOSS Search API alongside with the search results) for extracting patterns. The corpus of snippets and titles downloaded from all SAT word pairs contains $412,110,644$ tokens. We run the pattern extraction algorithm described in the previous section on this corpus and extract $12,712,608$ lexical patterns. However, this set of patterns is very sparse and most patterns occur only a few times in the corpus. Consequently, we select patterns that occur at least 50 times in the corpus for the remainder of the experiments described in this paper. The selected set contains $48,253$ lexical patterns. Top ranked 10 patterns are shown in Table I alongside with their frequencies in the corpus.

C. Training

For given two pairs of words (A, B) and (C, D), we create a feature vector using the patterns selected in Section III-B. First, we record the frequency of occurrence of each selected pattern in snippets for each word pair. We call this the *pattern frequency*. It is a local frequency count, analogous to *term frequency* in information retrieval [10]. Secondly, we combine the two pattern frequencies of a pattern (i.e., frequency of occurrence in snippets for (A, B) and that in snippets for (C, D) using various feature functions to compute the feature values for training. The different feature functions experimented in the paper are explained in Section IV.

We model the problem of identifying the correct answer for a given SAT question as a candidate ranking problem. To illustrate our proposed method let us assume that (A, B) is the question word pair and $\mathcal{S} = \{(C_i, D_i)_{i=1}^{5}\}$ is the set of candidate answer word pairs. Moreover, let us assume that (C_j, D_j) is the correct answer (i.e. j-th candidate) for this question. We form four partial ordering preferences for this question as follows, $((A, B), (C_j, D_j)) \succ ((A, B), (C_i, D_i))$, for $i \neq j$. Here, we use the notation $\succ$ to denote that a ranking function *prefers* the word pair (C_j, D_j) as the correct answer to the question word pair (A, B) over the other candidates (C_i, D_i). It is noteworthy that we do not induce any partial ordering preferences between the incorrect candidates. This setting is analogous to the problem of learning a ranking function to order results for a query in a search engine. The problem of rank learning has a wide variety of applications and a complete discussion is beyond the scope of the current paper. Ranking support vector machines (SVM) [11] can be

used to learn a ranking function from the set of induced partial ordering preferences. Once the ranking SVM model is trained, it can be applied to solve a SAT word analogy question.

IV. Experiments and Results

For the experiments in this paper we used the 374 SAT college-level multiple-choice analogy questions dataset which was first proposed by Turney et al. [5]. We compute the total score for answering SAT questions as follows,

$$\text{score} = \frac{100 \times \text{no. of correctly answered questions}}{\text{total no. of questions}}. \quad (1)$$

A. Feature Functions

In previous sections we described a method to represent a word pair using a pattern frequency vector. However, we are interested in measuring relational similarity between *two* word pairs. Therefore, we must somehow represent two word pairs using a single feature vector. It is not obvious how to construct a single feature vector to represent two word pairs using pattern frequencies that appear in two separate feature vectors representing each word pair.

Bollegala et al. [1] defined several functions to combine features from two vectors to construct a feature vector for two word pairs. However, they only evaluated symmetric functions because previous results in psychological experiments investigating similarity between words suggest that although similarity is an asymmetric phenomenon, the degree of asymmetry is less than 5%. However, this does not limit us to use only symmetric feature functions. We extend the set of feature functions proposed in [1] to include asymmetric feature functions and evaluate their effect empirically. Next, we describe each of the feature functions we investigate in this paper.

Let us assume the frequency of a pattern v in two word-pairs (A, B) and (C, D) to be f_{AB} and f_{CD}, respectively. We compute the value assigned to the feature corresponding to pattern v in the feature vector that represents the two word pairs (A, B) and (C, D) using the following four symmetric feature functions.

1) $|f_{AB} - f_{CD}|$: The absolute value of the difference of pattern frequencies is considered as the feature value.
2) $(f_{AB} - f_{CD})^2$: The square of the difference of pattern frequencies is considered as the feature value.
3) $f_{AB}+f_{CD}$: The sum of pattern frequencies is considered as the feature value.
4) $f_{AB} \times f_{CD}$: The product of the pattern frequencies is considered as the feature value.
5) JS divergence: Ideally, if two word pairs are analogous we would expect to see similar distributions of patterns in each word pair. Consequently, the *closeness* between the pattern distributions can be regarded as an indicator of relational similarity. We define a feature function based on Jensen-Shannon divergence [12] as a measure of the closeness between pattern distributions. Jensen-Shannon (JS) divergence $D_{JS}(P||Q)$, between two probability distributions P and Q is given by,

$$D_{JS}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M). \quad (2)$$

Here, $M = (P + Q)/2$ and D_{KL} is Kullback-Leibler divergence, which is given by,

$$D_{KL}(P||Q) = \sum_v P(v) \log \frac{P(v)}{Q(v)}. \quad (3)$$

Here, $P(v)$ denotes the normalized pattern frequency of a pattern v in the distribution P. Pattern frequencies are normalized s.t. $\sum_v P(v) = 1$ by dividing the frequency of each pattern by the sum of frequencies of all patterns. We define the contribution of each pattern towards the total JS-divergence in Formula 2 as its feature value, $JS(v)$. Substituting Formula 3 in 2 and collecting the terms under summation, we derive $JS(v)$ as,

$$JS(v) = \frac{1}{2}(p \log \frac{2q}{p+q} + q \log \frac{2p}{p+q}). \quad (4)$$

Here, p and q respectively denote the normalized pattern frequencies of f_{AB} and f_{CD}.

All of the above-described feature functions are symmetric in their arguments (i.e. we obtain the same feature value even if we reverse the two feature vectors given as arguments for the feature function). Next, we define asymmetric feature functions.

1) f_{AB}/f_{CD}: The ratio between the frequency of pattern v in the two vectors is taken as the feature value. If f_{CD} is zero, then we set the feature value to zero.
2) f_{CD}/f_{AB}: Same as above, except we consider the reversed ratio. If f_{AB} is zero, then we set the feature value to zero.
3) $f_{AB} - f_{CD}$: We subtract frequencies of patterns in each word pair and take it as the feature value.
4) $f_{CD} - f_{AB}$: Same as above, except we consider the reversed difference.
5) KL divergence, $D_{KL}(f_{AB}||f_{CD})$: Each term within the summation in the definition for KL divergence given by Equation 3 can be considered as the contribution of a pattern v towards the total divergence between the two distributions. This is an asymmetric measure. We use it to compute an asymmetric feature function. Here, we have used the notation $D_{KL}(\cdot||\cdot)$ to denote the contribution of a pattern v towards the total divergence. Note that f_{AB} and f_{CD} are *not* distributions.
6) Reverse KL divergence, $D_{KL}(f_{CD}||f_{AB})$: Same as the above, except we consider the reverse of the two pattern frequencies.

There are $5 + 6 = 11$ different feature functions. We use those feature functions to construct feature vectors for given two pairs of words. Next, we train a ranking support vector machine as described in Section III-C. We use five popular kernels in our experiments: linear, quadratic (degree = 2), cubic (degree = 3), Radial Basis Functions (RBF), and the Sigmoid

kernel. To evaluate the effect of different kernels with different feature functions, we use each feature function separately with each kernel function to train and test on SAT word analogy questions. We use svmlight[3] as the SVM implementation in our experiments. We do not tune any of the parameters in SVM including the kernel parameters. All parameters are set to their default values in svmlight. To avoid any bias towards the difference in the range of absolute values of features, we normalize each feature to range $[0, 1]$ by dividing from the maximum value of that feature. There are 374 word analogy questions in the SAT dataset. Each question typically has 5 candidate answer pairs, whereas for a small number of questions there are only 4 candidate answer pairs. There is only one correct answer word pair for each SAT word analogy question. We randomly select 50 questions for testing and the remaining $374 - 50 = 324$ questions are used as training data. Experimental results for are presented in Table II.

From Table II, we see that the combination of multiplicative feature function ($f_{AB} \times f_{CD}$) with RBF kernel produces the maximum SAT score of 58% on the test dataset. Among all asymmetric feature functions, the KL divergence ($D_{KL}(f_{AB}||f_{CD})$) reports the maximum SAT score when used with the cubic kernel. The performance of $f_{AB} \times f_{CD}$ and $D_{KL}(f_{AB}||f_{CD})$ are comparable across different kernel functions, except in Sigmoid kernel where KL divergence perform poorly. Considering the fact that SVMs are sensitive to the kernel parameters, we believe that the poor performance observed with the Sigmoid kernel is a result of the sub-optimal kernel parameter values. All kernels perform consistently well when used with multiplicative feature function even without any parameter tuning. Therefore, we can conclude that multiplicative feature function is robust against different kernel choices when used in SVMs. It is noteworthy that both subtraction and division of feature values perform poorly with all kernel functions. Considering the fact that random guessing on SAT dataset yields an SAT score of 20% the performance of those asymmetric feature functions is not significantly different from the random baseline. KL divergence-based feature function clearly outperforms all other asymmetric feature functions when used with any of the kernels considered in this experiment. This shows that although asymmetric feature functions are useful to detect relational similarity, naively combining features to produce asymmetric feature functions is not desirable. It is particularly interesting to note that the KL divergence ($D_{KL}(f_{AB}||f_{CD})$) outperforms its counterpart, the reverse KL divergence ($D_{KL}(f_{CD}||f_{AB})$). This result shows that in SAT word analogy questions we must take into consideration the fact that we are comparing a single question word pair with multiple candidate answer pairs. Lexical patterns that occur in a question word pair (i.e. (A, B)) define the relation that is considered by that question. In $D_{KL}(f_{AB}||f_{CD})$, the log ratios are weighted by the probabilities of patterns that occur in the question word pair. Therefore, divergences from the patterns that occur frequently in the question word pair are weighted higher than those that occur less frequently in question word pair. From Table II, we can see that quadratic and cubic kernels outperform the linear kernel in the best symmetric (multiplicative) and asymmetric (KL divergence) feature functions. Unlike the linear kernel, which considers each feature independently, polynomial kernels consider combinations of features. Therefore, the superior performance reported by polynomial kernels against the linear kernel suggests that the lexical patterns that we use as features are not independent. This can be explained considering the fact that there exist multiple lexical patterns that represent the same semantic relation.

[3]http://www.cs.cornell.edu/People/tj/svm_light/

TABLE II
EFFECT OF FEATURE FUNCTIONS AND KERNEL FUNCTIONS.

Feature function	linear	quadratic	cubic	RBF	sigmoid
$\lvert f_{AB} - f_{CD} \rvert$	24	28	28	20	18
$(f_{AB} - f_{CD})^2$	22	20	24	24	18
$f_{AB} + f_{CD}$	26	32	36	10	12
$f_{AB} \times f_{CD}$	44	50	46	58	50
JS divergence	24	26	28	24	22
f_{AB}/f_{CD}	22	12	14	16	14
f_{CD}/f_{AB}	26	22	14	16	14
$f_{AB} - f_{CD}$	26	26	24	20	22
$f_{CD} - f_{AB}$	26	26	24	20	22
$D_{KL}(f_{AB}\|\|f_{CD})$	46	50	52	36	20
$D_{KL}(f_{CD}\|\|f_{AB})$	22	20	16	16	8

TABLE III
COMPARISON AGAINST PREVIOUS WORK.

	Algorithm	score		Algorithm	score
1.	Phrase Vectors	38.2	2.	Thesaurus Paths	25
3.	Synonym	20.7	4.	Antonym	24
5.	Hypernym	22.7	6.	Hyponym	24.9
7.	Meronym:substance	20	8.	Meronym:part	20.8
9.	Meronym:member	20	10.	Holonym:substance	20
11.	Holonym:member	20	12.	Similarity:dict	18
13.	Similarity:wordsmyth	29.4	14.	Combined [5]	45
15.	Binary SVM [1]	40.1	16.	**Proposed** (RankSVM)	46.9
17.	WordNet [7]	42.8	18.	VSM [6]	47.1
19.	Pertinence [13]	53.5	20.	LRA [4]	56.1

B. Comparison against previous work

Table III summarizes various relational similarity measures proposed in previous work. All algorithms in Table III are evaluated on the same SAT analogy questions. Score (given by Formula 1) is the percentage of correctly answered questions to the total number of questions (374) in the dataset. An SAT question typically contain 5 choices. Therefore, a random guessing algorithm would obtain a score of 20. The score reported by average senior high-school student is about 57 [6]. We performed 5-fold cross validation on SAT questions to evaluate the performance of the proposed method using the multiplicative feature function and the RBF kernel, which we found to be the best combination in Table II. We do not tune any parameters in the RBF kernel nor the SVM. The results reported in Table II must not be directly compared against the SAT scores in Table III because in Table II we do not perform a cross-validation over the entire 374 questions instead use a randomly selected test set of 50 questions for efficiency reasons when training with a wide range of kernel functions and feature functions. The first 13 algorithms were proposed by

Turney et al. [5], in which they combined these modules using a weight optimization method. For given two word pairs, the phrase vector (row 1) algorithm creates a vector of manually created pattern-frequencies for each word-pair and compute the cosine of the angle between the vectors. Algorithms in rows 2-11 use WordNet to compute various relational similarity measures based on different semantic relations defined in WordNet. **Similarity:dict** (row 12) and **Similarity:wordsmith** (row 13) respectively use `Dictionary.com` and `Wordsmyth.net` to find the definition of words in word-pairs and compute the relational similarity as the overlap of words in the definitions. The proposed method outperforms all those 13 individual modules reporting a score of 46.9, which is comparable to the combined approach which has an SAT score of 45. The Binary SVM approach [1] models the SAT word analogy question solving as a binary classification approach, where a binary classifier is trained using the word pairs in the SAT dataset. Specifically, from each SAT question, a positive training instance is created by combining the question word pair with the word pair that corresponds to the correct answer. Likewise, negative training instances are created by combining the question word pair with word pairs that correspond to incorrect answers in a question. This method uses the multiplicative feature function as used by the proposed method. However, linear kernel is used instead of the RBF kernel. This method reports an SAT score of 40.1 which is less than the SAT score of the proposed ranking SVM approach. Because each SAT question describes a different semantic relation we must not directly compare positive or negative instances generated from different SAT questions as done by the binary SVM method. On the other hand, the proposed ranking SVM-based approach does not compare partial orderings generated from different SAT questions. We believe that the improved performance of the ranking approach can be partially attributable to this difference. Moreover, the number of lexical patterns used in our proposed method is much larger $(48,253)$ than the Binary SVM method which uses a relatively small set of $9,980$. Therefore, we can represent a semantic relation that exists between two words using a richer feature representation. The proposed method outperforms the WordNet-based relational similarity measure [7]. One limitation of the WordNet-based relational similarity measure is that it cannot compute the relational similarity between word pairs where at least one of the four words is not in the WordNet. Because named entities are not well-covered by WordNet, we believe that the proposed relational similarity measure can be useful when computing relational similarity between pairs that involve named entities. In future work, we intend to evaluate the proposed method using named entities. The SAT score reported by the proposed method is comparable to the VSM method. However, the proposed method is outperformed by both pertinence and LRA methods. One reason for this might be that we only consider co-occurrences of patterns in snippets, whereas VSM, pertinence and LRA use co-occurrences of patterns in a large corpus. Therefore, it is interesting to explore the possibility of supervised learning approaches to measuring relational similarity using co-occurrences in a larger corpus. Although there are multiple incorrect answers in an SAT question, they are not ranked by their degree of relational similarity in the SAT benchmark dataset. Therefore, all partial orderings involving those incorrect answers are considered equally during training. If we can obtain some ranking information for the incorrect answers, then we can use that to further guide the rank learning process. One possibility is to rank each incorrect answer according to the number of times it was selected by examinees. If many examinees select an incorrect answer, then it is likely that it is relationally similar to the question word pair although it is not the correct answer to the question.

V. Conclusion

We proposed a ranking-based approach to detect relationally similar word pairs to a given word pair. We generated pairwise partial ordering constraints from SAT word analogy questions and train a a ranking support vector machine to identify relationally similar word pairs. Our proposed method achieved an SAT score of 46.9 on a benchmark dataset of 374 SAT word analogy questions. Considering the fact that the average SAT score on word analogy questions by high school students is only 57, this is a strongly encouraging result. In future, we intend to explore other rank learning algorithms for this task.

References

[1] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Www sits the sat: Measuring relational similarity on the web," in *ECAI 2008: Proceedings, 18th European Conference on Artificial Intelligence, July 21-25, 2008, Patras, Greece: Including Prestigious Applications of Intelligent*, 2008, pp. 333 – 337.

[2] 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.

[3] V. Natase and S. Szpakowicz, "Exploring noun-modifier semantic relations," in *Proc. of fifth int'l workshop on computational semantics (IWCS-5)*, 2003, pp. 285–301.

[4] P. Turney, "Similarity of semantic relations," *Computational Linguistics*, vol. 32, no. 3, pp. 379–416, 2006.

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

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

[7] T. Veale, "Wordnet sits the sat: A knowledge-based approach to lexical analogy," in *Proc. of 16th European Conference on Artificial Intelligence (ECAI'04)*, 2004, pp. 606–612.

[8] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring the similarity between implicit semantic relations from the web," in *WWW 2009*, 2009, pp. 651 – 660.

[9] 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.

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

[11] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. of KDD'02*, 2002.

[12] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 2002.

[13] P. Turney, "Expressing implicit semantic relations without supervision," in *Proc. of Coling/ACL'06*, 2006, pp. 313–320.