Improving Relational Similarity Measurement using Symmetries in Proportional Word Analogies

Danushka Bollegala , Tomokazu Goto , Nguyen Tuan Duc , Mitsuru Ishizuka

Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

Abstract

Measuring the similarity between the semantic relations that exist between words is an important step in numerous tasks in natural language processing such as answering word analogy questions, classifying compound nouns, and word sense disambiguation. Given two word pairs (A, B) and (C, D), we propose a method to measure the *relational similarity* between the semantic relations that exist between the two words in each word pair. Typically, a high degree of relational similarity can be observed between proportional analogies (i.e. analogies that exist among the four words, A is to B such as C is to D). We describe eight different types of relational symmetries that are frequently observed in proportional analogies and use those symmetries to robustly and accurately estimate the relational similarity between two given word pairs. We use automatically extracted lexical-syntactic patterns to represent the semantic relations that exist between two words and then match those patterns in Web search engine snippets to find candidate words that form proportional analogies with the original word pair. We define 8 types of relational symmetries for proportional analogies and use those as features in a supervised learning approach. We evaluate the proposed method using the Scholastic Aptitude Test (SAT) word analogy benchmark dataset. Our experimental results show that the proposed method can accurately measure relational similarity between word pairs by exploiting the symmetries that exist in proportional analogies. The proposed method achieves an SAT score of 49.2% on the benchmark dataset, which is comparable to the best results reported on this dataset.

Key words: Relational Similarity, Proportional Analogy, SAT dataset

Email addresses: danushka@iba.t.u-tokyo.ac.jp (Danushka Bollegala), goto@mi.ci.i.u-tokyo.ac.jp (Tomokazu Goto), duc@mi.ci.i.u-tokyo.ac.jp (Nguyen Tuan Duc), ishizuka@i.u-tokyo.ac.jp (Mitsuru Ishizuka).

1 Introduction

Similarity measures can be categorized broadly into two types: *attributional* similarity measures and *relational* similarity measures. For attributional similarity measures, the objective is to compute the similarity between two given words by comparing the attributes of each word. For example, the two words *lion* and *cat* share many attributes such as *being carnivorous mammals*, *having sharp teeth*, and *walk using four legs*. Consequently, they are considered as being attributionally similar. A high degree of attributional similarity can be observed between synonymous words. On the other hand, relational similarity is the correspondence between the semantic relations that exist between two *word pairs*. For example, the relational similarity between the two word pairs (*ostrich, bird*) and (*lion, cat*) is high because ostrich is a large bird whereas lion is a large cat. The semantic relation is a large is common between the two word pairs in this example. In this paper, we propose a method to accurately measure the relational similarity between two word pairs using the symmetric properties in proportional analogies.

Proportional analogies exhibits a high degree of relational similarity. In our previous example, the four words: *ostrich*, *bird*, *lion*, and *cat* forms the proportional analogy *ostrich*:*bird*::*lion*:*cat*. This notation is read ostrich is to bird such as lion is to cat. For notational simplicity, we will write the proportional analogy $A : B :: C : D$ between four words, A, B, C, and D in the form $((A, B), (C, D))$, without using the colon symbols. Although we mainly focus on individual words (i.e. unigrams) in this paper, the discussion can be easily extend to named entities or multiword expressions. We will refer to A, B, C, and D in a proportional analogy as the first, second, third, and the fourth word following that order. Proportional analogies have gained attention in previous work on relational similarity because of their simple analogical representation. Moreover, the ability to recognize proportional analogies has been used as a measure to evaluate the language and reasoning skills in students in examinations such as the Scholastic Aptitude Test (SAT).

An SAT analogy question comprises a question word pair (here on referred to as the source word pair) of concepts/terms and a choice of (usually five) possible candidate answer word pair (here on referred to as target word pairs), only one of which accurately reflects relation that exists between the two words in the source word pair. A typical example is shown below.

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

Here, the relation is a large holds between the two words in the source word pair (e.g. Ostrich and Bird), which is also shared between the two words in the correct answer (e.g. Lion is a large Cat). SAT analogy questions have been used as a benchmark to evaluate relational similarity measures in previous work on relational similarity (Veale, 2004; Turney, 2006b). We propose a method that explicitly takes into consideration the symmetries observed in proportional analogies to solve word analogy questions using the World Wide Web as a knowledge base.

Numerous tasks in natural language processing can benefit from an accurate relational similarity measure. For example, a relational similarity measure can be used to classify noun-modifier pairs depending on the semantic relation that exists between a head noun and its modifier in a noun-modifier pair. 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 (2003) classified noun-modifiers into five classes according to the relations between the noun and the modifier. For example, *flu virus* is classified as an instance of the is the cause of relation because virus is the cause of the flu. Turney (2006b) 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.

An interesting application of relational similarity in information retrieval is to search using implicitly stated analogies (Marx et al., 2002; Veale, 2003). For example, the query "Muslim Church" is expected to return "mosque", and the query "Hindu bible" is expected to return "the Vedas". These queries can be formalized as word pairs: *(Christian, Church)* vs. *(Muslim,x)*, and *(Christian, Bible)* vs. *(Hindu,y)*. We can then find the words *x* and *y* that maximize the relational similarity in each case. Duc et al. (2010) developed a latent relational search engine that can efficiently answer such relational queries using the Web data. They extract lexical patterns that describe the relation between the first two words in a relational query and then match those patterns with the third word to find potential candidates for the fourth word.

Despite the numerous potential applications, accurate measurement of relational similarity remains a major research challenge. First, in the case of proportional analogies, we are only given two word pairs and we must first recognize the relations that are implicitly expressed by the two words in each of those word pairs. For example, given the word pair (*ostrich, bird*), we must first extract the relation that ostrich is a large bird. Second, there can be more than one relation between a given word pair. For example, between the two words *ostrich* and *bird*, aside from the relation is a large, there is also the relation is a flightless. A method to measure relational similarity must first extract all relations between the two words in each word pair before it can compute the similarity between the word pairs. Third, there can be more than one way to express a particular semantic relation in a text. For example, the three lexical patterns – ***X*** *was acquired by* ***Y***, **Y** *completed the* *acquisition of* **X**, and **Y** *buys* **X**– all indicate an acquisition relation between two companies **X** and **Y**. Here, we use the variables **X** and **Y** to indicate respectively the first and second words in a word pair (**X**, **Y**).

We present a method to accurately measure the relational similarity between two word pairs (A, B) and (C, D) which form a proportional analogy, $A : B :: C : D$. Our proposed method utilizes the symmetries that can be observed in proportional analogies to robustly measure relational similarity. For example, if (A, B) and (C, D) are relationally similar, then we can expect the word pairs that are formed by swapping the two words in each word pair (i.e. (B, A) and (D, C)) to be also relationally similar. Therefore, we can estimate the relational similarity between two word pairs using both the original formulation as well as the secondary word-swapped formulation. This redundancy enables us to make more accurate measurement of relational similarity. For example, consider the proportional analogy *((lion,cat),(ostrich,bird))* and the secondary formulation created by swapping the two words in each word pair, *((cat,lion),(bird,ostrich))*. If we represent the first word, *lion*, in the original word pair by **X**, and the second word, *cat*, in the original word pair by **Y**, then the lexical pattern **X** *is a large* **Y** describes the semantic relation that exist between the two words in each word pair in the original formulation, whereas, the lexical pattern *large* **X***'s such as* **Y** describes the same semantic relation in the secondary formulation. We recognize eight different types of such *relational symmetries* in proportional analogies. We systematically combine those eight types of relational symmetries to develop an accurate relational similarity measure. To the best of our knowledge, ours is the first attempt to combine multiple symmetric formulations of word pairs to design a relational similarity measure. The proposed relational similarity measure significantly outperforms numerous baselines and previously proposed relational similarity measures on a benchmark dataset that contains SAT word analogy questions.

2 Related Work

The Structure Mapping Theory (SMT) (Falkenhainer et al., 1989) is based on the premise that an analogy is a mapping of knowledge from one domain (base) into another (target), which conveys that a system of relations known to hold in the base also holds in the target. The target objects need not resemble their corresponding base objects. This structural view of analogy is based on the intuition that analogies are about relations, rather than simple features. Although this approach works best when the base and the target are rich in higher-order causal structures, it can fail when structures are missing or flat (Veale and Keane, 2003).

Turney et al. (2003) combined 13 independent modules by considering the weighted sum of the outputs of each module to solve SAT analogy questions. The best performing individual module was based on the Vector Space Model (VSM). In the

VSM approach (Turney and Littman, 2005), a vector is first created for a word pair (A, B) by counting the frequencies of various lexical patterns in which slot markers **X** and **Y** are substituted respectively by A and B. 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. The numbers of hits for respective queries are 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 that represent the two word pairs. Turney et al. (2003) introduced a dataset containing 374 SAT analogy questions to evaluate relational similarity measures. An SAT analogy question consists of a stem word pair that acts as the question, and five choice word pairs. The choice word pair that has the highest relational similarity with the stem word pair is selected by the system as the correct answer. The average SAT score reported by high school students for word-analogy questions is 57%. The VSM approach achieves a score of 47% on this dataset.

Turney (Turney, 2005, 2006b) proposed Latent Relational Analysis (LRA) by extending the VSM approach in three ways: a) lexical patterns are automatically extracted from a corpus, b) the Singular Value Decomposition (SVD) is used to smooth the frequency data, and c) synonyms are used to explore variants of the word pairs. Similarly, in the VSM approach, LRA represents a word pair as a vector of lexical pattern frequencies. First, using a thesaurus, he finds related words for the two words in a word pair and creates additional word pairs that are related to the original word pairs in the dataset. Second, n-grams of words are extracted from the contexts in which the two words in a word pair cooccur. The most frequent n-grams are selected as lexical patterns to represent a word pair. Then a matrix of word pairs vs. lexical patterns is created for all the word pairs in the original dataset and the additional word pairs. Elements of this matrix correspond to the frequency of a word pair in a lexical pattern. Singular value decomposition is performed on this matrix to reduce the number of columns (i.e. patterns). Finally, the relational similarity between two word pairs is computed as the average cosine similarity over the original word pairs and the additional word pairs derived from them. In fact, LRA achieves a score of 56.4% on SAT word analogy questions.

Both VSM and LRA require numerous search engine queries to create a vector to represent a word pair. For example, with 128 patterns, the VSM approach requires at least 256 queries to create two pattern-frequency vectors for two word pairs before it can compute the relational similarity. In fact, LRA considers synonymous variants of the given word pairs. For that reason, it requires even more search engine queries. Methods that require numerous queries impose a heavy load on search engines. Despite efficient implementations, singular value decomposition of large matrices is time consuming. In fact, LRA takes over 9 days to process the 374 SAT analogy questions (Turney, 2006b). This can be problematic for most systems that require a real-time response. Moreover, in the case of named entities, thesauri of related words are not usually available or are not complete, which becomes a problem when creating the additional word pairs required by LRA.

Veale (2004) proposed a relational similarity measure based on the taxonomic similarity in WordNet. The quality of a proportional analogy $A : B :: C : D$ is evaluated through comparison of the paths in the WordNet, joining *A* to *B* and *C* to *D*. Relational similarity is defined as the similarity between the paths that connect A to B and the paths that connect C to D. However, WordNet does not fully cover named entities such as personal names, organizations and locations, which becomes problematic when using this method to measure relational similarity between words such as named entities, that are not well covered by the WordNet.

Using a relational similarity measure, Turney (2006a) proposed an unsupervised learning algorithm to extract patterns that express implicit semantic relations from a corpus. His method produces a ranked set of lexical patterns that unambiguously describes the relation between the two words in a given word pair. Patterns are ranked according to their pertinence to the input word pair (i.e., the expected relational similarity between a word pair that co-occurs with the pattern and the input word pair). The representative lexical patterns are expected to be ranked at the top of the list. To answer an SAT analogy question, first, ranked lists of patterns are generated for each of the six word pairs (one stem word pair and five choice word pairs). Then each choice is evaluated by taking the intersection of its patterns with the stem's patterns. The shared patterns are scored by the average of their rank in the stem's list and the choice's lists. The algorithm picks the choice with the lowest scoring shared pattern as the correct answer. This method reports an SAT score of 54.6%.

Relational similarity measures have been applied in natural language processing tasks such as generating word analogies (Davidov and Rappoport, 2008b), and classifying noun-modifier compounds based on the relation between the head and the modifier (Turney, 2006b; Nakov and Hearst, 2008; Davidov and Rappoport, 2008a; Nakov and Hearst, 2008). Davidov and Rappoport (2008b) proposed an unsupervised algorithm to discover general semantic relations that pertain between lexical items. They represent a semantic relation with a cluster of patterns. They use the pattern clusters to generate SAT-like word analogy questions for English and Russian languages. The generated questions are then solved by human subjects. They do not evaluate their method for relational similarity between named entities.

Bollegala et al. (2009) 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. 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.

Symmetries that exist in proportional analogies have been explicitly discussed in some previous work. For example, Equation 8 in (Turney, 2006b) uses the relational symmetry between ((A,B),(C,D)) and ((B,A),(D,C)) when building the word-pairs vs. lexical patterns co-occurrence matrix. The symmetry Types 5 and 6 in our method capture this version of relational symmetry. However, we incorporate additional relational symmetry types that are not discussed in (Turney, 2006b). Lepage (2000) presents all eight types of symmetries that exist in proportional analogies as discussed in our work. However, his work is limited to analogies between strings of symbols, whereas our work covers lexical proportional analogies.

We identify several important symmetries that exist in words involved in a proportional analogy. We empirically show that we can improve a relational similarity measure simply by considering a linear combination of those relational symmetries. This improvement is complementary to the techniques used in previous work such as dimensionality reduction, clustering and supervised metric learning. Therefore, we believe that the relational symmetries presented in this paper can potentially contribute to previously proposed relational similarity measures as well.

3 Relational Symmetries in Proportional Analogies

Let us denote a relational similarity measure defined over a proportional analogy $A : B :: C : D$ by $\mathrm{RelSim}((A, B), (C, D))$. One method to determine how relationally similar the word pair (A, B) to (C, D) is by measuring the ability to recall a word (for example say D) using the remaining three words (i.e. A, B, and C). Next, we outline the main steps of this method of estimating relational similarity in a proportional analogy. Further details will be presented in the sections to follow.

First, we use the two words in the word pair (A, B) to find lexical patterns that describe the semantic relation that exist between A and B. In Section 3.1, we present the details of the exact method that we use to extract a set of lexical patterns that describes the semantic relation that exist between two given words. Second, we substitute the word C in each of the extracted lexical patterns in their first slot. Third, we match the set of lexical patterns that we obtain by substituting C in the previous step in a large corpus (e.g. Web data) to find the candidates that match the second slot in the lexical patterns. The method we use to extract a set of candidates for the fourth word, D, is explained in Section 3.2. Here on, we denote the word for which candidates are being extracted by a question mark, “?”, in the remainder of the paper. For example, if we are extracting candidates for D in the proportional analogy, $A : B :: C : D$, then we will denote this fact by $A : B :: C :?$ or alternatively using the bracket notation, $((A, B), (C, ?))$.

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

Fig. 1. A snippet returned for the query *"Google * * * YouTube"*.

The candidates are assigned a score that takes into account two factors: the strength of association between a lexical pattern p extracted for the word pair (A, B), and the strength of association between p and (C, D). The exact scoring formula is detailed in Section 3.3. If D is extracted by the above-mentioned procedure and assigned a high score, then we assume that (A, B) and (C, D) are relationally similar – the value of computed score is considered as the relational similarity, $\mathrm{RelSim}((A, B), (C, D))$, between the two word pairs (A, B) and (C, D). Otherwise, we assume that (A, B) and (C, D) are not relationally similar. The above-mentioned procedure can be repeated by extracting candidates for A, B, C, or D. Moreover, we can swap the two words in each word pair to further induce additional word pairs. This gives us a redundant method to estimate the relational similarity in a proportional analogy using multiple estimates, thereby producing a more robust relational similarity measure. The symmetries that we observe in proportional analogies are explained in Section 3.4.

3.1 Representing Semantic Relations

Given a word pair, (A, B), the first step in our proposed method is to explicitly represent the semantic relation that exists between A and B. For this purpose, we employ the subsequence lexical pattern extraction algorithm proposed by Bollegala et al. (2009). Next, we describe this method.

The context in which two words co-occur provides useful clues about the semantic relations that pertain between those words. We propose the use of text snippets retrieved using a Web search engine as an approximation of the context of two words. Snippets (also known as *dynamic teasers*) are brief summaries provided by most Web search engines along with the search results. Typically, 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 or there are numerous URLs among the search results.

A snippet for a query containing two words captures the local context in which they cooccur. For example, consider the snippet shown in Figure 1, returned by *Yahoo* [1]

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

for the query *"Google * * * YouTube"*. Here, the wildcard operator "*" (asterisk) matches one word or none in a document. The snippet in Figure 1 is extracted from an online newspaper article about the acquisition of YouTube by Google.

To retrieve snippets for a word pair (A, B), we use the following three types of queries: *"A * B"*, *"A * * B"*, and *"A * * * B"* (the quotation marks are also part of the queries). The quotation marks around a query will ensure that the two words appear in the specified order (e.g. *A* before *B* in snippets retrieved for the query *"A * B"*). The queries containing the wildcard operator "*" returns snippets in which the two words, *A* and *B* appear within a window of specified length. We designate such queries as *contextual* queries. Most modern Web search engines support contextual queries. We search for snippets in which the query words co-occur within a maximum window of three words (tokens). This process is intended to approximate the local context of two words in a document.

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

Lexical syntactic patterns have been used in various natural language processing tasks such as extracting hypernyms (Hearst, 1992; Snow et al., 2005), or meronyms (Berland and Charniak, 1999), question answering (Ravichandran and Hovy, 2001), and paraphrase extraction (Bhagat and Ravichandran, 2008). Following these previous works, we present a shallow lexical pattern extraction algorithm to represent the semantic relations between two words. The proposed method requires no language-dependent preprocessing such as part-of-speech tagging or dependency parsing, which can be both time consuming at Web scale, and likely to produce incorrect results because of the fragmented and ill-formed snippets. The pattern extraction algorithm consists of the following three steps.

Step 1: Given a context S (e.g. a snippet), retrieved for a word pair (A, B) according to the procedure described above, we replace the two words A and B, respectively, with two variables ***X*** and ***Y***. Legal abbreviations such as *Inc., Ltd., Corp.*, and titles such as *Mr., Ms., Prof., Dr., Rev.* are considered as occurrences of the query terms. For example, *Google Inc.* is considered as an occurrence of the entity *Google*. We replace all numeric values by num, a marker for digits.

Punctuation marks are not removed.

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

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

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

(iii). When generating a subsequence from a snippet, we allow it to skip words. However, the maximum number of words that can be continuously skipped when generating a subsequence is set to g. Moreover, the total number of all skipped words in a subsequence must not exceed G words. To illustrate the process of skipping words in a context for the purpose of generating subsequence, consider the portion of the snippet in Figure 1 which reads *Google to acquire YouTube for 1.65 billion in stock*. After replacing the two entities Google and YouTube respectively by variables X and Y, and numeric values by num as described in Step 1, we obtain ***X*** *to acquire* ***Y*** *for* num *billion in stock*. From this context, we not only generate subsequences with continuous words such as ***X*** *to acquire* ***Y*** *for*, but also generated subsequences where one or more words are skipped to produce subsequences such as ***X*** *acquire* ***Y*** *for*, or ***X*** *to* ***Y*** *for*.

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

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

Our pattern extraction algorithm has four parameters (ca. *L, g, G* and *T*), which are set to their default values $L = 5$, $g = 2$, $G = 2$, and $T = 5$ as described in (Bollegala et al., 2009). Specifically, using a dataset of relationally similar entity pairs for four semantic relation types, they use a grid search method to determine the optimal values of those parameters. They use the extracted lexical patterns to represent an entity pair, and compute the relational similarity between entity pairs that express the same semantic relations. For parameter tuning we used the development dataset created in Bollegala et al. (2009) that consists of named entities such as personal names, organizations, etc. Then, the values of the above-mentioned parameters are set to those that maximizes the relational similarity between entity pairs that express the same semantic relation. Note that this development dataset does not contain any word pairs that appear in the SAT word analogy dataset that we use for evaluation purposes in this paper.

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 word skips enables us to capture relations between distant words in a snippet. We use a modified version of the *PrefixSpan* algorithm (Pei et al., 2004) to generate subsequences. Next, we briefly summarize the PrefixSpan algorithm.

Prefix-projected Sequential Pattern Mining (PrefixSpan) algorithm is a subsequence pattern mining algorithm that first generates a set of frequent prefix subsequences and then project only their corresponding posfix subsequences to create a projected database of sequential patterns. In each projection step, the existing sequential patterns are grown by exploring only local frequent patterns. The pseudo code for the PrefixSpan algorithm is shown in Algoritm 1. We consider the words in a snippet as a sequence of elements and run Algorithm 1 on each snippet individually. The minimum support threshold θ is set to 2 so that we consider any subsequence that occur at least twice in the entire set of snippets.

Algorithm 1 PrefixSpan.

```
– Main routine
Input: A sequence database $\mathcal{S}$, and the minimum support threshold $\theta$.
Output: The complete set of sequential patterns.
1: Call $\mathrm{PrefixSpan}([], 0, \mathcal{S})$

– $\mathrm{PrefixSpan}(\alpha, l, \mathcal{S}|_{\alpha})$
Input: $\alpha$: a sequential pattern, $l$: the length of $\alpha$, $\mathcal{S}|_{\alpha}$: the projected database, if
    $\alpha \neq []$; otherwise the sequence database $\mathcal{S}$.
Output: The spanned pattern $\alpha'$.
1: Scan $\mathcal{S}|_{\alpha}$ once, find the set of frequent items $b$ such that
    (a) $b$ can be assembled to the last element of $\alpha$ to form a sequential pattern; or
    (b) $[b]$ can be appended to $\alpha$ to form a sequential pattern.
2: For each frequent item $b$, append it to $\alpha$ to form a sequential pattern $\alpha'$, and
    output $\alpha'$.
3: For each $\alpha'$, construct $\alpha'$ - projected database $\mathcal{S}|_{\alpha'}$, and call
    $\mathrm{PrefixSpan}(\alpha', l+1, \mathcal{S}|_{\alpha'})$.
```

The conditions in Step 2 in our pattern extraction procedure are used to prune the search space, thereby reducing the number of generated subsequences in prefixspan. Specifically, in Line 1 of Algorithm 1 in $\mathrm{PrefixSpan}$ subroutine we use the conditions described in Step 2 in our pattern extraction procedure to further limit the candidate subsequences we generate.

All patterns extracted from the snippet shown in Figure 1 are listed in Table 1. In total, we extract 13 lexical patterns from this snippet. On average, the proposed pattern extraction method extracts 8 lexical patterns from a snippet.

Table 1
The set of lexical patterns extracted from the snippet shown in Figure 1.

total words skipped	Lexical Patterns
0	***X** to acquire **Y***, ***X** to acquire **Y*** for
1	***X** acquire **Y***, ***X** acquire **Y*** for, ***X** acquire **Y*** for $num, ***X** to **Y***, ***X** to **Y*** for, ***X** to **Y*** for $num
2	***X Y*** for, ***X Y*** for $num, ***X Y*** for $num billion, ***X** acquire **Y*** $num, ***X** to **Y*** $num

3.2 Extracting Candidates

Let us assume that we have extracted a set of lexical patterns, $\mathcal{P}(A, B)$, that describes the semantic relation that exists between the two words A and B as described in Section 3.1. Moreover, let us denote a lexical pattern (i.e. element) of this set by $p(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ and $\mathbf{Y}$ indicate the place markers for the first (i.e. A) and second (i.e. B) of the two words in the word pair (A, B) for which those lexical patterns are extracted. Using the set of lexical patterns, $\mathcal{P}(A, B)$, we must next retrieve candidates for the fourth word in a proportional analogy. For example, if we assume that we are extracting candidates for D (i.e. $A : B :: C :?$), then we fill the first slot (corresponding to the marker **X**) in each lexical pattern $p(\mathbf{X}, \mathbf{Y})$, by C and the second slot (i.e. corresponding to the marker **Y**) by an asterisk, "*" to construct a pattern, $p(C, *)$. Note that the asterisk would match against none or a single word when issued as a query to a search engine that supports the NEAR operator as used for retrieving snippets from a Web search engine as done in Section 3.1. We then issue the newly formed lexical patterns, $p(C, *)$, as queries to a Web search engine and extract the words that match the asterisk in the retrieved set of snippets. Note that one could extend the matching window by including more than one asterisk as we did when extracting lexical patterns in Section 3.1. However, all the proportional analogies that appear in the SAT benchmark dataset that we use in our experiments (described later in Section 4) consist of single words [2]. Therefore, we limit ourselves to extracting single words as candidates of D in the above-mentioned procedure (hence, the use of a single asterisk in the queries).

For example, let us consider the proportional analogy ((*ostrich*, *bird*), (*lion*, ?)), in which we must extract candidates for the fourth word. Moreover, let us assume that we extracted the lexical pattern, ***X** is a large **Y***, using the two words *ostrich* and *bird*. To extract candidates for D, we first replace **X** in the lexical pattern by *lion* and **Y** in the lexical pattern by * to form the lexical pattern, "*lion is a large* *", which is then issued to a Web search engine as a contextual query. This query would retrieve snippets where for example the "*" is replaced by the word *cat*, which is

[2] a single word is a word that is formed by only one token, such as "lion" or "cat"

the correct word for D to form a proportional analogy in the current example.

3.3 Scoring Candidates

Considering both the ambiguities in lexical patterns and the noisy nature of Web data, it is not guaranteed that we will always retrieve only the correct candidate for D using the extraction procedure mentioned in Section 3.2. In practice, we would often extract a large set of lexical patterns for a word pair (A, B) and those lexical patterns in return retrieve numerous candidates for D. Therefore, we must score each extracted candidate depending on their ability to form a proportional analogy with A, B, and C. Inversely, given two word pairs (A, B) and (C, D), the score assigned to D among that for the other candidates extracted for $((A, B), (C, ?))$ can be considered as an estimate of the relational similarity, $\mathrm{RelSim}((A, B), (C, D))$, between the word pairs (A, B) and (C, D).

We compute the score of D, denoted by $\mathrm{score}(D)$, as follows

$$
\begin{aligned}
&\mathrm{score}(D) \\
&= \mathrm{Prob}((C, D)|\mathcal{P}(A, B)) \times \mathrm{Prob}(\mathcal{P}(A, B)|(A, B)) && (1)\\
&= \sum_{p \in \mathcal{P}(A,B)} \mathrm{Prob}((C, D)|p(\mathbf{X}, \mathbf{Y})) \times \mathrm{Prob}(p(\mathbf{X}, \mathbf{Y})|(A, B)) && (2)\\
&= \sum_{p \in \mathcal{P}(A,B)} \frac{\mathrm{Prob}(p(C, D))}{\mathrm{Prob}(p(C, *))} \times \frac{\mathrm{Prob}(p(A, B))}{\mathrm{Prob}((A, B))} && (3)\\
&\approx \sum_{p \in \mathcal{P}(A,B)} \frac{\frac{h(p(C,D))}{N}}{\frac{h(p(C,*))}{N}} \times \frac{\frac{h(p(A,B))}{N}}{\frac{h(\text{“}A{*}{*}{*}B\text{”})}{N}} && (4)\\
&= \sum_{p \in \mathcal{P}(A,B)} \frac{h(p(C, D))}{h(p(C, *))} \times \frac{h(p(A, B))}{h(\text{“}A * * * B\text{”})}. && (5)
\end{aligned}
$$

Here, the $\mathrm{score}(D)$ is computed as the product of two conditional probabilities: $\mathrm{Prob}((C, D)|\mathcal{P}(A, B))$ and $\mathrm{Prob}(\mathcal{P}(A, B)|(A, B))$. The first conditional probability term measures the likelihood of co-occuring the two words C and D with the set of lexical patterns, $\mathcal{P}(A, B)$, extracted for the word pair (A, B). The second conditional probability term measures the likelihood of extracting the set of lexical patterns, $\mathcal{P}(A, B)$, for the word pair (A, B). Intuitively, if some lexical patterns are likely to be extracted by the word pair (A, B) and those patterns are also likely to co-occur with (C, D), then D will receive a high score under this formula. One can think of the set $\mathcal{P}(A, B)$ of lexical patterns as *bridging* the word pair (A, B) to the word pair (C, D). In Line 2 in the formula we assume the lexical patterns to be mutually independent and partition product terms over individual lexical patterns in $\mathcal{P}(A, B)$. Ideally, not all lexical patterns are mutually independent in practice and a more advanced model must take into account the correlation between lexical patterns. However, the independence assumption greatly simplifies the computation of scores and we empirically show in Section 4 that despite the simplicity of the assumption the proposed scoring formula can accurately estimate the relational similarity between word pairs.

In Line 3 of the scoring formula, we use the product rule of conditional probability to write each term in the summation as a division between two probabilities. The first term inside the summation in Line 3 is the division between the joint probability of pattern p co-occurring with the word pair (C, D) and the marginal probability of the pattern p. Likewise, the second term inside the summation in Line 3 can be written as the division between the joint probability of pattern p co-occurring with the word pair (A, B) and the marginal probability of the word pair (A, B). We estimate each of those probabilities using the number of page counts retrieved from a Web search engine for different queries. We use the notation $h(q)$ to denote the number of page counts returned by a Web search engine for the query q. Here, N denotes the total number of pages (Web documents) on the Web. Although, the exact value of N can be difficult to obtain and might vary considerably over time, it cancels out in the numerator and the denominator in the formula as shown in Line 5, thus not required for computing the scores. $h(p(C, D))$ denotes the number of page counts we obtain for the query we produce by substituting C and D respectively for the slots **X** and **Y** in the lexical pattern p. Likewise, $h(p(A, B))$ denotes the number of page counts we obtain for the query we produce by substituting A and B respectively for the slots **X** and **Y** in the lexical pattern p. To find the number of occurrences of C with a lexical pattern p, we replace **X** with C and **Y** with an asterisk in p and obtain the number of page counts (i.e. $h(p(C, *))$) from a Web search engine. Finally, we obtain the page counts for the co-occurrences of A and B by issuing the contextual query “A *** B” to a Web search engine (i.e. $h(\text{“}A *** B\text{”})$). The finally derived formula shown in Line 5 can be computed only using page counts for four types of queries.

It is noteworthy that page counts is an approximation of co-occurrences between words on the Web. For example, page counts are different from so called *hits* because, even the words in the query appear multiple times in a single page, page counts will be counting such co-occurrences only once, whereas such within page co-occurrences would result in multiple hits. Moreover, page counts are estimated by most Web search engines for efficiency reasons and it is difficult to obtain exact page counts in most cases (especially for high frequent queries). Although a full discussion is beyond the scope of this paper, there have been some recent work that attempt to overcome these difficulties and estimate accurate page counts using Web search engines’ public search interfaces (Matsuo et al., 2007; Bar-Yossef and Gurevich, 2006).

3.4 Relational Symmetries

Let us restate here the main objective of this paper – given two word pairs (A, B) and (C, D) measure the relational similarity, $\mathrm{RelSim}((A, B), (C, D))$ between those word pairs. The score computed in Section 3.3 is one proxy estimate of the relational similarity between (A, B) and (C, D). In addition to scoring D, given a set $\mathcal{P}(A, B)$ of lexical patterns extracted for a word pair (A, B), we can score the remainder of the three words (i.e. A, B, and C) given lexical patterns extracted for (A, B) or (C, D) to estimate the relational similarity $\mathrm{RelSim}((A, B), (C, D))$. Next, we identify eight different types of estimation schemes for the permutations corresponding to the different settings.

Type 1: $((\mathbf{A}, \mathbf{B}), (\mathbf{C}, ?))$ This is case we discussed in Section 3.3, where we use the word pair (A, B) to extract a set of lexical patterns, $\mathcal{P}(A, B)$, and then use those patterns to compute the candidate score, $\mathrm{score}(D)$, for the word D.

Type 2: $((\mathbf{A}, \mathbf{B}), (?, \mathbf{D}))$ This type is similar to Type 1, except for the fact that we compute the score for C (i.e. $\mathrm{score}(C)$) instead for D using the set of lexical patterns $\mathcal{P}(A, B)$ extracted for the word pair (A, B).

Type 3: $((\mathbf{C}, \mathbf{D}), (\mathbf{A}, ?))$ In this type, we interchange the word pairs and measure the relational similarity of (C, D) to (A, B). This operation is based on the assumption that relational similarity is symmetric on the word pairs. Type 3 formation assumes that, the two relational similarity values $\mathrm{RelSim}((A, B), (C, D))$ and $\mathrm{RelSim}((C, D), (A, B))$ are equal. Previous studies in psycholinguistics investigating the similarity between words have found empirical evidence that suggest similarity (in particular semantic similarity between words or visual objects) can be asymmetric in some situations (Medin et al., 1991; Tversky, 1997). However, the level of asymmetry observed in those experiments is less than 5% of the total cases evaluated (Medin et al., 1991). Consequently, much similarity measures proposed in previous work has assumed similarity to be symmetric notion. Type 3 formation introduced here follows previous work on relational similarity measures assumes relational similarity to be symmetric.

Type 4: $((\mathbf{C}, \mathbf{D}), (?, \mathbf{B}))$ This type is similar to the Type 3 above except for the fact that we compute the score for A instead for B. We extract a set of lexical patterns, $\mathcal{P}(C, D)$, for the word pair (C, D) and then use those patterns to compute the score, $\mathrm{score}(A)$ as described in Section 3.3.

Type 5: $((\mathbf{B}, \mathbf{A}), (\mathbf{D}, ?))$ We swap the two words in each word pair to create two new word pairs and then measure the relational similarity between those word pairs. For example, we swap the two words in the word pair (A, B) to create a new word pair (B, A) and we swap the two words in the word pair (C, D) to create a new word pair (D, C). Note that in general the set of lexical patterns

$\mathcal{P}(A, B)$ that we would obtain if we use the word pair (A, B) is not necessarily identical to the set of lexical patterns $\mathcal{P}(B, A)$ that we would obtain if we use the word pair (B, A). For example, given the word pair (*ostrich*, *bird*), we can expect to find the lexical pattern ***X is a large Y***, whereas for the word pair (*bird*, *ostrich*) we would extract the lexical pattern ***Xs such as Y***. However, the semantic relation that is described by a word pair does not change if we swap the two words in that word pair. This enables us to estimate the relational similarity between two word pairs using an additional redundancy. Note that we must swap the two words in *both* word pairs simultaneously. Otherwise, we are not guaranteed to find the same set of lexical patterns with each word pair even though the two word pairs might be highly relationally similar. In Type 5, we start with the word pair (B, A) and extract a set of lexical patterns, $\mathcal{P}(B, A)$, and then use those patterns to compute the score, $\mathrm{score}(C)$, using the formula described in Section 3.3.

Type 6: $((\mathbf{B}, \mathbf{A}), (?, \mathbf{C}))$ This type is similar to Type 5 above except for the fact that we are computing the score for D instead for C.

Type 7: $((\mathbf{D}, \mathbf{C}), (\mathbf{B}, ?))$ We combine the symmetry in relational similarity described under Type 3 with the word swapping method described under Type 4, to form a novel type. Specifically, we swap the two words in each word pair as well as inter-changing the two word pairs. After conducting this operation we compute the score, $\mathrm{score}(A)$ using the formula described in Section 3.3.

Type 8: $((\mathbf{D}, \mathbf{C}), (?, \mathbf{A}))$ This type is similar to type 7 above except for the fact that we compute the score for B instead for A.

The above mentioned eight types of analogies provide us eight different ways to estimate the relational similarity between the two word pairs (A, B) and (C, D). In the remainder of the paper, we refer collectively refer to those eight types as *relational symmetry types* or in short as *symmetry types*. It remains an interesting question as how to combine these different types of symmetries to robustly estimate the relational similarity between two word pairs. One simple approach is to assume that all eight types *equally* represent the original proportional analogy (i.e. Type 1) and take the arithmetic mean of their scores (i.e. divide the sum of scores in all types 1-8 by 8). However, it is not obvious as to how the different types of symmetries capture the concept of relational similarity as compared to the original version (i.e. type 1). In Section 4.2, we propose a method to combine the different symmetry types using a supervised learning approach.

4 Evaluation

4.1 Benchmark Dataset

Attributional similarity measures are frequently evaluated by comparing the similarity ratings assigned by a set of human annotators to a set of word pairs (Miller and Charles, 1998; Rubenstein and Goodenough, 1965; Finkelstein et al., 2002). Previous work on attributional similarity have often employed correlation metrics such as the Pearson correlation coefficient (on the Miller-Charles dataset (Miller and Charles, 1998)) or the Spearman correlation coefficient (on the WordSimilarity-353 dataset (Finkelstein et al., 2002)). A high correlation with human ratings of similarity indicates a good attributional similarity measure. However, such a direct comparison against human ratings cannot be performed for relational similarity measures because a human rated dataset is not available for relational similarity. Instead, *all* previous work on relational similarity evaluate a relational similarity measure by the ability of that measure to correctly answer Scholastic Aptitude Test (SAT) word analogy questions. Turney et al. (2003) first proposed the use of SAT word analogy questions as a benchmark dataset for evaluating relational similarity measures. We use the SAT benchmark dataset to compare the different types of relational symmetries discussed in Section 3.4. Moreover, by using the SAT dataset we can directly compare our results against previous work on relational similarity (Section 4.2).

A typical SAT word analogy question was shown in Section 1. For the ease of explanation, we repeat it below.

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

An SAT word analogy question is defined by a word pair, which acts as the question (often referred to as the *stem word pair* or the *source word pair* in the literature). The examinees are required to select the most *analogous* word pair from four or five candidate word pairs (often referred to as the *target word pairs* in the literature) to the source word pair. To be able to correctly answer an SAT question, examinees must not only understand the meaning of individual words but also must correctly infer the relations that exist between two words and compare that against the relations that exist between the two words in the source word pair.

To evaluate a relational similarity measure using SAT questions the following procedure is taken. First the relational similarity between the source word pair (let us assume to be (A, B)) and each of the target word pairs (let us assume the i-th target word pair to be (C_i, D_i)) is measured. Let us denote the relational similarity between the source word pair and the i-th target word pair by $\mathrm{RelSim}((A, B), (C_i, D_i))$. Next, we select the target word pair with the maximum relational similarity to the source word pair as the correct answer to that question. Using the above notation, the correct candidate i^* is given by,

$$i^* = \operatorname*{argmax}_i \mathrm{RelSim}((A, B), (C_i, D_i)). \tag{6}$$

An SAT question has only one correct answer. In the official SAT examination an examinee is awarded one point for a correct answer, whereas one point is penalized for an incorrect answer. Therefore, examinees are encouraged not to randomly guess answers but to skip questions when they are uncertain about the correct answer. An algorithm that answers SAT questions can be evaluated for its performance by comparing the correctly answered questions to the total number of questions in the dataset or the total number of question attempted by the algorithm. In our experiments, we do not skip any questions and answer all questions in the test dataset. Therefore, we use accuracy as our evaluation measure. Accuracy is defined as follows.

$$\mathrm{Accuracy} = \frac{\text{no. of correctly answered questions}}{\text{total no. of questions in the test dataset}} \tag{7}$$

Note that in a setting in which an algorithm attempts all questions in a test dataset and the outcome for a question can be only of the two – correct or incorrect, typical precision (ratio of the correctly answered questions to the total number of question attempted) and recall (ratio of the correctly answered questions to the total questions in the dataset) measures will become equal (hence the F-score will also be equal to precision and recall) to the accuracy measure as defined in Equation 7. Therefore, we only report accuracy scores in our experimental results. Moreover, the total number of questions in the SAT dataset is 374. Therefore, the denominator in Equation 7 is set to 374.

4.2 Combining Different Relational Symmetries

In Section 3.4, we introduced eight different types of scoring functions based on relational symmetries that exist in proportional analogies. Any one of those scoring functions can be used as a measure of relational similarity between the source word pair and a target word pair in an SAT question. Therefore, we can evaluate the different types of symmetries discussed in Section 3.4 for their ability to capture the notion of relational similarity by employing the corresponding scoring functions to answer SAT word analogy questions.

Moreover, we study the combined effect of the relational symmetries proposed in the paper for the purpose of solving word analogy questions. Given two word pairs (A, B) and (C, D), we construct an eight dimensional feature vector to represent the two word pairs using the eight symmetry types proposed in the paper. We then model the problem of detecting analogical word pairs as a binary classification task. Specifically, given a word pair (A, B) that corresponds to a question in an SAT word analogy question, we assign a positive (+1) label to the feature vector representing $((A, B), (C, D))$, in which (C, D) is the correct answer for the question denoted by (A, B). Otherwise, we assign a negative (-1) label to the feature vector that represents $((A, B), (C, D))$. We then train a two-class (binary) Support Vector Machine (SVM) (Vapnik, 1998) with the radial basis function (RBF) kernel, to classify positive vs. negative instances.

In the test phase, to answer an SAT question we use the following procedure. First, for each candidate answer word pair (C, D) of an SAT question (A, B), we compute the eight relational symmetry types described in Section 3.4. Next we represent $((A, B), (C, D))$ by a feature vector in which, each feature corresponds to a particular relational symmetry type, exactly as we did during the training phase. Then we use the trained SVM model to classify each of the feature vectors individually and select the candidate word pair that is furthest on the positive side of the decision hyper-plane as the correct answer. If all candidates are classified as negative instances, then we select the candidate that is closest to the decision hyper-plane (i.e. least negative instance) as the correct answer.

Note that because an SAT question contains multiple incorrect candidate answers, and a single correct candidate answer, the number of negative training instances is larger (ca. 4 times more) than that of the negative training instances. Typically when such an imbalance of positive vs. negative training data exists in a training corpus, it affects classification accuracy and some form of a data balancing technique must be used (Kubat and Matwin, 1997; Zheng et al., 2004). Although we varied the amount of negative examples in our preliminary experiments, we did not observe any significant difference in the classification accuracy. We therefore used all negative instances to train the binary SVM. We performed 10-fold cross-validation using the 374 SAT word analogy questions. The results reported in the paper are the average accuracy over the 10 folds. The SVM cost parameter C (margin violation penalty) and the RBF kernel parameter γ are set to their optimal values using a grid search procedure. We use a portion of training data as development data in each fold to tune those two parameters. On average over the 10-folds, C takes the value 10 and γ takes the value 0.1.

In Table 2, we compare the performance of individual relational symmetry types $1-8$ with the combination of those symmetry types (denoted by **Combined**) we learnt

Table 2
Performance of individual symmetry types and their combination via. supervised learning.

Method	Accuracy
Combined	49.2
Type 7	43.7
Type 8	43.4
Type 3	35.9
Type 6	35.3
Type 2	34.1
Type 1	33.7
Type 5	29.4
Type 4	29.0

using the SVM using the accuracy scores computed using Formula 7. To solve SAT word analogy questions using a single symmetry type we use the approach presented in Equation 6. Specifically, we compare each candidate answer word pair (C, D) in an SAT question denoted by a word pair (A, B) using the symmetry scores computed as described in Section 3.3. Next, for each relational symmetry type T, we select the candidate word pair that reports the highest score with T for the SAT word analogy question that we are solving. In Table 2, we arrange the different methods compared in the descending order of their accuracies computed using Equation 7. Moreover, we perform Fisher exact test (Fisher, 1922) between all pairs of methods ($9 \times 9 = 81$ in total) to test for the statistical significance of the accuracies reported in Table 2.

From Table 2, we see that the combined approach reports the highest classification accuracy. The performance reported by the combined approach is statistically significantly different from all the other methods compared in Table 2 according to the Fisher exact test under $p < 0.01$ critical level. Among the individual relational symmetry types, type 7 reports the best accuracy for this task. However, the difference in accuracy between type 7 and 8 is not significantly different. The better performance of types 7 and 8 among the individual relational symmetry types can be attributed to two reasons. First, recall that type 7 is the case $((D, C), (B, ?))$, in which we compute the score for A, and type 8 is the case $((D, C), (?, A))$, in which we compute the score for B. Note that (A, B) is the source word pair in an SAT question and is always guaranteed to have one or more semantic relations between A and B. Therefore, we are likely to extract numerous lexical patterns that describe the semantic relations that exist between A and B. On the other hand, not all candidate word pairs in an SAT questions express a semantic relation. Therefore, by first extracting lexical patterns for a candidate word pair, and then matching those lexical patterns against the source word pair, the incorrect matches due to the ambiguity

Table 3
Performance on the SAT dataset.

Algorithm	score	95% confidence interval
Random guessing	20.0%	16.1 – 24.5%
Jiang & Conrath Turney (2006b)	27.3%	23.1 – 32.4%
Lin (Turney, 2006b)	27.3%	23.1 – 32.4%
Leacock & Chodrow (Turney, 2006b)	31.3%	26.9 – 26.5%
Hirst & St.-Onge (Turney, 2006b)	32.1%	27.6 – 37.4%
Resnik (Turney, 2006b)	33.2%	28.7 – 38.5%
PMI-IR (Turney, 2006b)	35.0%	30.2 – 40.1%
SVM (Bollegala et al., 2008)	40.1%	35.9 – 45.3%
LSA+Predictation (Mangalath et al., 2004)	42.0%	37.2 – 47.4%
WordNet (Veale, 2004)	43.0%	38.0 – 48.2%
Bicici and Yuret (2006)	44.0%	39.0 – 49.3%
VSM (Turney and Littman, 2005)	47.1%	42.2 – 52.5%
Combined	49.2%	44.3 – 54.6%
Pair-Classifier (Turney, 2008)	52.1%	46.9 – 57.3%
RELSIM (Bollegala et al., 2009)	51.1%	46.1 – 56.5%
Pertinence (Turney, 2006a)	53.5%	48.5 – 58.9%
LRA (Turney, 2006b)	56.1%	51.6 – 61.2%
Human	57.0%	52.0 – 62.3%

of the relations represented by lexical patterns is greatly reduced. Second, note that the symmetry types 7 and 8 consider the situation where we have swapped the two words in each word pair. For this operation to be valid we must have a high degree of relational similarity between the two original word pairs (A, B) and (C, D). Therefore, if we can accurately recall A or B by using the lexical patterns extracted for (D, C), then there is strong evidence to the fact that (A, B) and (C, D) are indeed highly relationally similar. Among other relational symmetry types shown in Table 2, the performance among types 3, 6, 2, and 1 are not statistically significantly different according to the Fisher exact test under $p < 0.01$ level. Types 5 and 4 have the worst performance and the difference between those two types is not statistically significant according to the Fisher exact test under $p < 0.01$ level.

4.3 Comparison against Previous Work

In Table 3, we compare the relational symmetry-based relational similarity measuring approach (shown as **Combined**) against the previous work on relational similarity using the SAT benchmark dataset. We select the combination of symmetry types using SVM as the **Combined** method. We perform a Binomial Exact Test for each method compared in Table 3 and show the 95% confidence interval.

Note that there are five choices in a typical SAT question out of which only one is correct. Therefore, a random guessing strategy would obtain a 20% score on the SAT benchmark dataset. This strategy can be considered as a lower-bound baseline for comparisons and is shown by **Random guessing** in Table 3. The average SAT score obtained by high school students for the word analogy questions is 57.0% and is shown by **Human** in Table 3. However, **Human** score must not be interpreted as a upper bound for this task because many high school students obtain perfect SAT scores.

The **Combined** method reports an SAT score of 49.2% and is ranked 5-th in Table 3 among the compared relational similarity measures. The WordNet-based relational similarity measure proposed by Veale (2004) uses the manually created word ontology, WordNet, to find the relations between the two words in each word pair. The relations are expressed as the paths that connect one word to another in the WordNet ontology. Finally, relational similarity is computed by the number of common paths shared between the two word pairs. Although the **Combined** method does not use any manually compiled language resources such as the WordNet, and use lexical patterns extracted from snippets retrieved from a web search engine, it reports a higher accuracy than the WordNet-based approach (score of 43.0%). This is an interesting result because it suggests the possibility of measuring the relational similarity between word pairs which contain words that are not listed in the WordNet. For example, most named entities such as people, locations or organizations are not listed in the WordNet. Therefore, we believe the **Combined** method can complement WordNet-based relational similarity measures when measuring the relational similarity between word pairs that contain named entities. A future research direction is to explore the possibilities of integrating WordNet-based relational similarity with the symmetry types proposed in this work.

Nakov and Hearst (2008) proposed a linguistically motivated method to measure the relational similarity between word pairs, in which all words are nouns. They extract lexical patterns using a Web search engine to retrieve numerous contexts that provide clues regarding the semantic relations that exist between two words. Next, part-of-speech tagging and shallow parsing are conducted on the extracted contexts. They extract words that participate in numerous syntactic relations with a noun such as verbs, prepositions, and coordinate conjunctions. The extracted words are weighted using the tf-idf weighting scheme to down-weight common features.

Finally, the relational similarity between two word pairs is computed using the Dice measure between their corresponding feature vectors. They limit their evaluation to a subset consisting of 184 questions from the SAT benchmark dataset, in which all words are nouns. An accuracy score of 71.3% is reported by their method on this subset of word analogy questions. Because Table 3 summarizes previous work that evaluate using the entire SAT dataset, we have not included this method in Table 3. On the other hand, the relational symmetries presented in this paper are not limited to word pairs that consist only of nouns.

The methods that report accuracy scores higher than the **Combined** method in Table 3 are Pair-Classifier (Turney, 2008), RELSIM (Bollegala et al., 2009), Pertinence (Turney, 2006a), and LRA (Turney, 2006b). RELSIM is a fully supervised distance metric learning approach that use the SAT dataset as training data to learn a relational similarity measure. RELSIM first clusters numerous lexical patterns extracted for word pairs, and then use those clusters as features in an information theoretic distance metric learning approach. Both clustering and the metric learning steps are computationally costly and become problematic when we have a large number of lexical patterns. Pair-Classifer method (Turney, 2008) trains a supervised classifier to detect numerous types of semantic relations such as synonyms, hypernyms, and antonyms using the word pairs in SAT dataset as training data.

Pertinence and LRA methods use singular value decomposition (SVD) to perform dimensionality reduction before computing relational similarity. Because multiple lexical patterns can express the same semantic relation, dimensionality reduction techniques such as pattern clustering (Bollegala et al., 2009) or SVD have shown to produce improved results in relational similarity measurement.

On the other hand, the **Combined** method does not use any dimensionality reduction techniques when measuring relational similarity. Although we did not use dimensionality reduction in the current work because we wanted to empirically evaluate the benefit of using symmetry types when measuring relational similarity, a potential future research direction is to investigate efficient dimensionality reduction methods with symmetry types proposed in this paper to further improve the relational similarity measurement.

4.4 Computational Cost in Measuring Relational Similarity

A direct comparison of the computational costs involved with the different methods proposed for computing relational similarity is difficult because of several reasons. First, the different methods presented in Table 3 use different resources for computing relational similarity. Some methods use pre-compiled dictionaries such as the WordNet (Veale, 2004), some require locally indexed corpora (Turney, 2005), whereas some methods use Web search engines to retrieve contexts in which two words co-occur (Nakov and Hearst, 2008; Bollegala et al., 2009). Second, those methods are implemented using different programming languages and the architectures of the machines on which experiments are conducted vary. Third, the implementations of those methods are not publicly available. Therefore, we resort to an estimation of computational cost based on the number of web search engine queries that each method would require if we were to apply those methods at Web scale.

First, Let us consider the scenario where we compute the relational similarity between two pairs of words using any one of the relational symmetries proposed in this paper. As a working example, consider computing type 1 symmetry for two word pairs (A, B) and (C, D). We first send the the three queries *"A * B"*, *"A * * B"*, and *"A * * * B"* to retrieve snippets for the word pair (A, B). Let us assume that we extracted n number of lexical patterns from all the snippets retrieved from those three queries. Next, to compute the relational symmetry type 1 using Equation 1, we require additional $3n$ web search queries: n queries to compute $h(p(A, B))$ with each pattern p coupled with the word pair (A, B), n queries to compute $h(p(C, D))$ with each pattern p coupled with the word pair (C, D), and n queries to compute $h(p(C, *))$ with each pattern p coupled with the word C. Note that we can obtain *h("A * * * B")* when we issue the query *"A * * * B"* to retrieve snippets for the word pair (A, B). Therefore, the total number of web search engine queries that we require to compute type 1 symmetry is $3n + 3$. In our experiments, we select the most frequent 10 lexical patterns among all patterns extracted for the word pair (A, B). Therefore, $n = 10$ in our experiments, which implies that we require 33 search engine queries on average to compute a particular relational symmetry type. Assuming that an SAT word analogy question contains five candidate answers, we must compute the relational symmetry type with each of the candidate word pairs. However, the question word pair (A, B) is common to all symmetry types, which means we can re-use the counts $h(p(A, B))$ and *h("A * * * B")*. We must however issue n queries to obtain $h(p(C, D))$ counts and another n queries to obtain $h(p(C, *))$ counts for each candidate word pair (C, D). Therefore, the total number of search engine queries we require to solve an SAT question is $(3n + 3) + (5 \times 2n) = 13n + 3$. As mentioned earlier, in our experiments, we select only the most frequent 10 lexical patterns from the snippets retrieved for a word pair (i.e. $n = 10$). Therefore, the total number of web search engine queries required to solve a single SAT question using a particular relational symmetry type is 133.

On the other hand, to solve an SAT question using LRA, we must first create the co-occurrence matrix between word pairs and lexical patterns. Let us denote the number of patterns used by LRA by n and the total number of unique word pairs in the SAT dataset by d. In LRA, for each word in a word pair its most similar synonyms are obtained from a thesaurus to create additional 3 word pairs. Therefore, the total number of word pairs generated from a single pair of words is $(1+3) = 4$. Therefore, the dimensions of the co-occurrence matrix is $4d \times n$. LRA uses the

Wumpus search engine [3] and retrieve contexts in which two words in a word pair co-occur using phras queries. Next, the retrieved contexts are analyzed to obtain co-occurrences of lexical patterns with word pairs. The total number of queries required by LRA to solve a single SAT question is $4 \times 6 = 24$ (stem word pair plus five candidate word pairs per question). Although this value is less than the 133 web search engine queries required by the proposed method, the amount of processing cost of large number of contexts to obtain co-occurrences between word pairs and lexical patterns might be significant in practice. Moreover, in the case of web search engines we have access only to the top ranked snippets and might not be able to obtain all co-occurrences of word pairs with lexical patterns using only those snippets.

When we combine all 8 symmetry types, the search engine queries required by the proposed method increases to $8 \times 133 = 1064$. Actually, the amount of queries required by a combined approach involving all eight relational symmetry types is less than this number because certain queries can be shared across different symmetry types. For example, types 1 and 2 both use (A, B) as the question word pair. Therefore, all queries related to (A, B) as well as the lexical patterns extracted for (A, B) can be shared between those two symmetry types. Therefore, 1064 must be considered as an upper bound on the number of queries required by a combined approach using all eight symmetry types.

From a computational time complexity point-of-view, the SVD step in LRA can be conducted efficiently using a truncated singular value decomposition algorithm (Brand, 2006) in time complexity $\mathcal{O}(ndr)$, where n is the number of rows in the matrix (i.e. the total number of word pairs), d is the number of columns in the matrix (i.e. the total number of lexical patterns), and r is the number of dimensions (i.e. columns in the case of LRA) after SVD. On the other hand, to compute a relational symmetry type using the Equation 1, we require the summation of n terms. The computational time complexity of this process is linear in the number of lexical patterns extracted for a word pair (A, B). Therefore, the proposed relational symmetries are desirable from the points of both the number of search engine queries as well as the overall computational time complexity.

The relational metric learning method proposed by Bollegala et al. (2009) uses information theoretic metric learning (ITML) (Davis et al., 2007) algorithm to learn a Mahalanobis distance metric to measure the relational similarity between word pairs. It is a fully supervised approach. During training, the distance between relationally similar word pairs (e.g. question word pair in a particular SAT question and its correct candidate answer) is minimized while the distance between relationally dissimilar word pairs (e.g. question word pair in a particular SAT question and its incorrect candidates) is maximized. This method requires sequential lexical pattern clustering algorithm to reduce the dimensionality of the feature space prior

[3] `http://www.wumpus-search.org/`

Table 4
Correspondence of relational symmetry types in four different dataset variants. Symmetry types $T1 - T8$ are shown for the original dataset (Version 1) and each of its transformations Versions 2-4.

Version 1	Version 2	Version 3	Version 4
$(A,B),(C,?)$ **T1**	$(C,D),(A,?)$ **T3**	$(B,A),(D,?)$ **T5**	$(D,C),(B,?)$ **T7**
$(A,B),(?,D)$ **T2**	$(C,D),(?,B)$ **T4**	$(B,A),(?,C)$ **T6**	$(D,C),(?,A)$ **T8**
$(C,D),(A,?)$ **T3**	$(A,B),(C,?)$ **T1**	$(D,C),(B,?)$ **T7**	$(B,A),(D,?)$ **T5**
$(C,D),(?,B)$ **T4**	$(A,B),(?,D)$ **T2**	$(D,C),(?,A)$ **T8**	$(B,A),(?,C)$ **T6**
$(B,A),(D,?)$ **T5**	$(D,C),(B,?)$ **T7**	$(A,B),(C,?)$ **T1**	$(C,D),(A,?)$ **T3**
$(B,A),(?,C)$ **T6**	$(D,C),(?,A)$ **T8**	$(A,B),(?,D)$ **T2**	$(C,D),(?,B)$ **T4**
$(D,C),(B,?)$ **T7**	$(B,A),(D,?)$ **T5**	$(C,D),(A,?)$ **T3**	$(A,B),(C,?)$ **T1**
$(D,C),(?,A)$ **T8**	$(B,A),(?,C)$ **T6**	$(C,D),(?,B)$ **T4**	$(A,B),(?,D)$ **T2**

to learning the Mahalanobis distance metric. The computational time complexity of the sequential clustering algorithm is $\mathcal{O}(n \log(n))$, in which n denotes the total number of lexical patterns extracted for all word pairs in the SAT dataset. Although this computational time complexity is much less than the cubic time complexity in LRA, it is still greater than the linear time complexity of our relational symmetries.

In an online setting in which we must measure relational similarity between two word pairs incrementally, methods that require minimum additional processing costs are desirable. Although SVD is required as an intermediate step in LRA, it is possible to quickly update the SVD result of a matrix when new rows are added without having to perform SVD on the new matrix from the scratch (Brand, 2006). Likewise, we can compute the relational symmetries proposed in this paper for any two word pairs independent of whether those word pairs were encountered before. Therefore, both LRA as well as the proposed method can be used in online relational measurement tasks.

4.5 Invariance of Relational Symmetries under Dataset Transformations

In Section 3.4, we introduced eight types of symmetries commonly observed in propositional analogies. As we show experimentally in Section 4.2, the combination of those symmetry types using supervised machine learning was able to accurately solve word analogy questions in the SAT benchmark dataset. In fact, such symmetric transformations can also be applied to the dataset itself to generate auxiliary word analogy questions. In this Section, we discuss how the relational symmetries proposed in this paper would perform on different word analogy datasets that stem from symmetrical transformations in word pairs.

Let us consider an SAT word analogy question in which, (A, B) denotes the question word pair and the five candidate word pairs are (C, D), (E, F), (G, H), and (I, J). Let us refer to this original untransformed dataset as Dataset Version 1. Without loss of generality let us further assume that (C, D) is the correct answer to (A, B). We will use $T1 - T8$ to denote the symmetry types computed for the two word pairs $((A, B), (C, D))$. A second version of the word analogy dataset can be generated by swapping (C, D) with (A, B). In other words, the original SAT question can be transformed into (C, D) vs. (A, B), (E, F), (G, H), and (I, J). Likewise two additional versions of the original (Version 1) dataset can be generated as follows. Version 3: (B, A) vs. (D, C), (F, E), (H, G), and (J, I). Version 4: (D, C) vs. (B, A), (F, E), (H, G), and (J, I).

Although the four versions of the dataset are very different from each other, it turns out that the symmetry types $T1-T8$ in the original dataset (Version 1) are sufficient to represent all transformed word pairs in the other versions. The correspondence between symmetry types and dataset versions are show in Table 4. For example, from Table 4 we see that the first symmetry type $T1$ which occurs for the word pairs $((A, B), (C, ?))$ in the original dataset (version 1), appears as the third symmetry type in Version 2. In fact, all four versions of the datasets consist of some permutation of the eight symmetry types. Therefore, in the combined approach, in which we use all eight types of relational symmetries to represent two word pairs, we would obtain the exact performance (accuracy) across all versions of the transformed dataset. This invariance property of the proposed method to symmetric transformations of word pairs makes it a robust relational similarity measure.

5 Conclusion

In this paper, we studied the problem of measuring the relational similarity between two given word pairs. First, we represent the semantic relations that exist between the two words in each word pair using lexical patterns extracted from snippets retrieved from a Web search engine for those words. Second, we proposed eight types of symmetries that can be used to measure the relational similarity between two words. The different symmetry types can be seen as redundant estimates that can be combined to accurately estimate the relational similarity between two word pairs. We used the eight symmetry types proposed in the paper as features to represent two word pairs by a feature vector. We then train a binary support vector machine (SVM) with radial basis function (RBF) kernel to learn the optimal combination of those eight symmetry types to solve SAT word analogy questions. Our experimental results show that the proposed method obtains a level of accuracy that is comparable to some of the previously proposed relational similarity measures, at the same time requiring much less web search engine queries. Moreover, the symmetry types provides a robust feature representation of two word pairs that is invariant against symmetric transformations on proportional analogies.

References

Bar-Yossef, Z., Gurevich, M., 2006. Random sampling from a search engine's index. In: Proceedings of 15th International World Wide Web Conference.

Berland, M., Charniak, E., 1999. Finding parts in very large corpora. In: Proc. of ACL'99. pp. 57–64.

Bhagat, R., Ravichandran, D., 2008. Large scale acquisition of paraphrases for learning surface patterns. In: Proc. of ACL'08: HLT. pp. 674–682.

Bicici, E., Yuret, D., 2006. Clustering word pairs to answer analogy questions. In: Proc. of TAINN'06.

Bollegala, D., Matsuo, Y., Ishizuka, M., 2008. Www sits the sat: Measuring relational similarity on the web. In: Proc. of ECAI'08. pp. 333–337.

Bollegala, D., Matsuo, Y., Ishizuka, M., 2009. Measuring the similarity between implicit semantic relations from the web. In: WWW 2009. pp. 651 – 660.

Brand, M., 2006. Fast low-rank modifications of the thin singular value decomposition. Linear Algebra and its Applications 415 (1), 20 – 30.

Davidov, D., Rappoport, A., 2008a. Classification of semantic relationships between nominals using pattern clusters. In: Proc. of the ACL'08.

Davidov, D., Rappoport, A., 2008b. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In: Proc. of ACL'08-HLT. pp. 692–700.

Davis, J. V., Kulis, B., Jain, P., Sra, S., Dhillon, I. S., 2007. Information-theoretic metric learning. In: IProc. of CML'07. pp. 209–216.

Duc, N., Bollegala, D., Ishizuka, M., 2010. Using relational similarity between word pairs for latent relational search on the web. In: Proc. of Int'l Conf. on Web Intelligence (WI'10).

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

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

Fisher, R. A., 1922. On the interpretation of χ^2 from contingency tables, and the calculation of p. Journal of the Royal Statistical Society 85 (1), 87 – 94.

Hearst, M., 1992. Automatic acquisition of hyponyms from large text corpora. In: Proc. of 14th COLING. pp. 539–545.

Kubat, M., Matwin, S., 1997. Addressing the curse of imbalanced training sets: one-sided selection. In: ICML 1997. pp. 179 – 186.

Lepage, Y., 2000. Languages of analogical strings. In: ACL'00. pp. 488 – 494.

Mangalath, P., Quesada, J., Kintsch, W., 2004. Analogy-making as predictation using relational information and lsa vectors. In: Proc. of Int'l Conf. on Research in Computational Linguistics.

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

Matsuo, Y., Tomobe, H., Nishimura, T., 2007. Robust estimation of google counts for social network extraction. In: AAAI'07. pp. 1395 – 1401.

Medin, D., Goldstone, R., Gentner, D., 1991. Respects for similarity. Psychological Review 6(1), 1–28.

Miller, G., Charles, W., 1998. Contextual correlates of semantic similarity. Language and Cognitive Processes 6(1), 1–28.

Nakov, P., Hearst, M., 2008. Solving relational similarity problems using the web as a corpus. In: Proc. of ACL'08-HLT. pp. 452–460.

Natase, V., Szpakowicz, S., 2003. Exploring noun-modifier semantic relations. In: Proc. of fifth int'l workshop on computational semantics (IWCS-5). pp. 285–301.

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

Ravichandran, D., Hovy, E., 2001. Learning surface text patterns for a question answering system. In: Proc. of ACL '02. pp. 41–47.

Rubenstein, H., Goodenough, J., 1965. Contextual correlates of synonymy. Communications of the ACM 8, 627–633.

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

Turney, P., 2005. Measuring semantic similarity by latent relational analysis. In: Proc. of IJCAI'05. pp. 1136–1141.

Turney, P., 2006a. Expressing implicit semantic relations without supervision. In: Proc. of Coling/ACL'06. pp. 313–320.

Turney, P., 2006b. Similarity of semantic relations. Computational Linguistics 32 (3), 379–416.

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

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

Turney, P. D., 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In: COLING'08. pp. 905 – 912.

Tversky, A., 1997. Features of similarity. Psychological Review 84(4), 327–352.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, Chichester, GB.

Veale, T., 2003. The analogical thesaurus. In: Proc. of 15th Innovative Applications of Artificial Intelligence Conference (IAAI'03). pp. 137–142.

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

Veale, T., Keane, M. T., 2003. The competence of structure mapping on hard analogies. In: Proc. of IJCAI'03.

Zheng, Z., Wu, X., Srihari, R., June 2004. Feature selection for text categorization on imbalanced data. ACM SIGKDD Explorations Newsletter 6 (1), 80 – 89.