Generating Lexical Analogies Using Dependency Relations

Andy Chiu, Pascal Poupart, and Chrysanne DiMarco
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{pachiu,ppoupart,cdimarco}@uwaterloo.ca

Abstract

A lexical analogy is a pair of word-pairs that share a similar semantic relation. Lexical analogies occur frequently in text and are useful in various natural language processing tasks. In this study, we present a system that generates lexical analogies automatically from text data. Our system discovers semantically related pairs of words by using dependency relations, and applies novel machine learning algorithms to match these word-pairs to form lexical analogies. Empirical evaluation shows that our system generates valid lexical analogies with a precision of 70%, and produces quality output although not at the level of the best human-generated lexical analogies.

1 Introduction

Analogy discovery and analogical reasoning are active research areas in a multitude of disciplines, including philosophy, psychology, cognitive science, linguistics, and artificial intelligence. A type of analogy that is of particular interest in natural language processing is *lexical analogy*. A lexical analogy is a pair of word-pairs that share a similar semantic relation. For example, the word-pairs (*dalmatian*, *dog*) and (*trout*, *fish*) form a lexical analogy because dalmatian is a subspecies of dog just as trout is a subspecies of fish, and the word-pairs (*metal*, *electricity*) and (*air*, *sound*) form a lexical analogy because in both cases the initial word serves as a conductor for the second word. Lexical analogies occur frequently in text and are useful in various natural language processing tasks. For example, understanding metaphoric language such as "*the printer died*" requires the recognition of implicit lexical analogies, in this case between (*printer*, *malfunction*) and (*person*, *death*). Lexical analogies also have applications in word sense disambiguation, information extraction, question-answering, and semantic relation classification (see (Turney, 2006)).

In this study, we present a novel system for generating lexical analogies directly from a text corpus without relying on dictionaries or other semantic resources. Our system uses dependency relations to characterize pairs of semantically related words, then compares the similarity of their semantic relations using two machine learning algorithms. We also present an empirical evaluation that shows our system generates valid lexical analogies with a precision of 70%. Section 2 provides a list of definitions, notations, and necessary background materials. Section 3 describes the methods used in our system. Section 4 presents our empirical evaluation. Section 5 reviews selected related work. Finally, Section 6 concludes the paper with suggested future work and a brief conclusion.

2 Definitions

A *word-pair* is a pair of entities, where each entity is a single word or a multi-word named entity. The *underlying relations* of a word-pair (w_1, w_2) are the semantic relations[1] between w_1 and w_2. For exam-

[1] Here 'semantic relations' include both *classical relations* such as synonymy and meronymy, and *non-classical relations* as defined by Morris and Hirst (2004).

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 561–570, Prague, June 2007.

ple, the underlying relations of (*poet*, *poem*) include *produces*, *writes*, *enjoys*, and *understands*. A *lexical analogy* is a pair of word-pairs that share at least one identical or similar underlying relation.

A key linguistic formalism we use is *dependency grammar* (Tesnière, 1959). A dependency grammar describes the syntactic structure of a sentence in a manner similar to the familiar phrase-structure grammar. However, unlike phrase-structure grammars which associate each word of a sentence to the syntactic phrase in which the word is contained, a dependency grammar associates each word to its syntactic superordinate as determined by a set of rules. Each pair of depending words is called a *dependency*. Within a dependency, the word being depended on is called the *governor*, and the word depending on the governor is called the *dependent*. Each dependency is also labelled with the syntactic relation between the governor and the dependent. Dependency grammars require that each word of a sentence have exactly one governor, except for one word called the *head word* which has no governor at all. A proposition p that is governor to exactly one word w_1 and dependent of exactly one word w_2 is often *collapsed* (Lin and Pantel, 2001); that is, the two dependencies involving p are replaced by a single dependency between w_1 and w_2 labelled p.

The dependency structure of a sentence can be concisely represented by a *dependency tree*, in which each word is a node, each dependent is a child of its governor, and the head word is the root. A *dependency path* is an undirected path through a dependency tree, and a *dependency pattern* is a dependency path with both ends replaced by slots (Lin and Pantel, 2001). Figure 1 illustrates various dependency structures of the sentence, *rebels fired rockets at a military convoy*, after each word is lemmatized.

3 Methods

We consider lexical analogy generation as a sequence of two key problems: *data extraction* and *relation-matching*. Data extraction involves the identification and extraction of pairs of semantically related words, as well as features that characterize their relations. Relation-matching involves matching word-pairs with similar features to form lexical analogies. We describe our methods for solving these two problems in the following subsections.

3.1 Data Extraction

Extracting Word-Pairs

To identify semantically related words, we rely on the assumption that highly syntactically related words also tend to be semantically related — a hypothesis that is supported by works such as Levin's (1993) study of English verbs. As such, the dependency structure of a sentence can be used to approximate the semantic relatedness between its constituent words. Our system uses a dependency parser to parse the input text into a set of dependency trees, then searches through these trees to extract dependency paths satisfying the following constraints:

1. The path must be of the form *noun-verb-noun*.
2. One of the nouns must be the subject of the clause to which it belongs.

Each of these paths is then turned into a word-pair by taking its two nouns. The path constraints that we use are suggested by the *subject-verb-object* (SVO) pattern commonly used in various relation extraction algorithms. However, our constraints allow significantly more flexibility than the SVO pattern in two important aspects. First, our constraints allow an arbitrary relation between the verb and the second noun, not just the object relation. Hence, word-pairs can be formed from a clause's subject and its location, time, instrument, and other arguments, which are clearly semantically related to the subject. Secondly, searching in the space of dependency trees instead of raw text data means that we are able to find semantically related words that are not necessarily adjacent to each other in the sentence.

It is important to note that, although these constraints improve the precision of our system and tend to identify effectively the most relevant word-pairs, they are not strictly necessary. Our system would be fully functional using alternative sets of constraints tailored for specific applications, or even with no constraints at all.

Using the sentence in Figure 1 as an example, our system would extract the dependency paths "*rebel* $\xleftarrow{subj}$ *fire* $\xrightarrow{obj}$ *rocket*" and "*rebel* $\xleftarrow{subj}$ *fire* $\xrightarrow{at}$ *convoy*", and would thus generate the word-pairs (*rebel*, *rocket*) and (*rebel*, *convoy*).

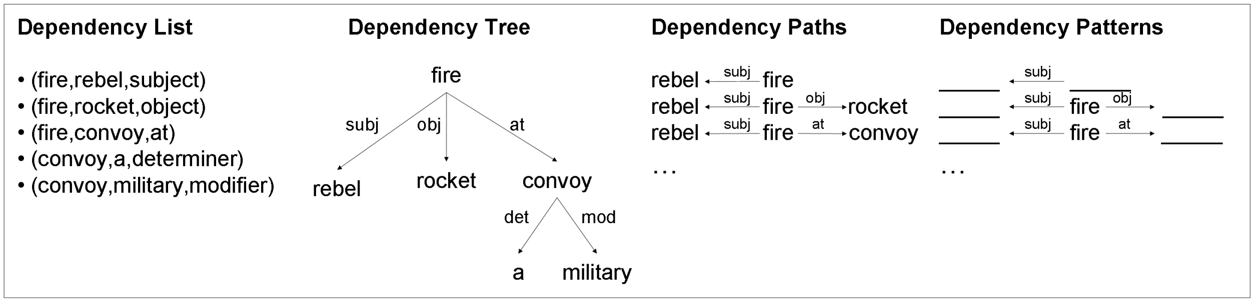

Figure 1: Dependency structures of "*rebels fired rockets at a military convoy*" after lemmatization

Extracting Features

Recall that each word-pair originates from a dependency path. The path, and in particular the middle verb, provides a connection between the two words of the word-pair, and hence is a good indication of their semantic relation. Therefore, for each word-pair extracted, we also extract the dependency pattern derived from the word-pair's dependency path as a feature for the word-pair. We further justify this choice of feature by noting that the use of dependency patterns have previously been shown to be effective at characterizing lexico-syntactic relations (Lin and Pantel, 2001; Snow et al., 2004).

Using Figure 1 as an example again, the dependency patterns "___ $\overset{subj}{\leftarrow}$ *fire* $\overset{obj}{\rightarrow}$ ___" and "___ $\overset{subj}{\leftarrow}$ *fire* $\overset{at}{\rightarrow}$ ___" would be extracted as a feature of (*rebel*, *rocket*) and (*rebel*, *convoy*), respectively.

Filtering

Word-pairs and features extracted using only dependency relations tend to be crude in several aspects. First, they contain a significant amount of noise, such as word-pairs that have no meaningful underlying relations. Noise comes from grammatical and spelling mistakes in the original input data, imperfect parsing, as well as the fact that dependency structure only approximates semantic relatedness. Secondly, some of the extracted word-pairs contain underlying relations that are too general or too obscure for the purpose of lexical analogy generation. For example, consider the word-pair (*company*, *right*) from the sentence "*the company exercised the right to terminate his contract*". The two words are clearly semantically related, however the relation (*have* or *entitled-to*) is very general and it is difficult to construct satisfying lexical analogies from the word-pair. Lastly, some features are also subject to the same problem. The feature "___ $\overset{subj}{\leftarrow}$ *say* $\overset{obj}{\rightarrow}$ ___", for example, has very little characterization power because almost any pair of words can occur with this feature.

In order to retain only the most relevant word-pairs and features, we employ a series of refining filters. All of our filters rely on the occurrence statistics of the word-pairs and features. Let $\mathcal{W} = \{wp_1, wp_2, ..., wp_n\}$ be the set of all word-pairs and $\mathcal{F} = \{f_1, f_2, ..., f_m\}$ the set of all features. Let $\mathcal{F}_{wp}$ be the set of features of word-pair wp, and let $\mathcal{W}_f$ be the set of word-pairs associated with feature f. Let $O(wp)$ be the total number of occurrences of word-pair wp, $O(f)$ be the total number of occurrences of feature f, and $O(wp, f)$ be the number of occurrences of word-pair wp with feature f. The following filters are used:

1. Occurrence filter: Eliminate word-pair wp if $O(wp)$ is less than some constant K_{f_1}, and eliminate feature f if $O(f)$ is less than some constant K_{f_2}. This filter is inspired by the simple observation that valid word-pairs and features tend to occur repeatedly.

2. Generalization filter: Eliminate feature f if $|\mathcal{W}_f|$ is greater than some constant K_{f_3}. This filter ensures that features associated with too many word-pairs are not kept. A feature that occurs with many word-pairs tend to describe overly general relations. An example of such a feature is "___ $\overset{subj}{\leftarrow}$ *say* $\overset{obj}{\rightarrow}$ ___", which in our experiment occurred with several thousand word-pairs while most features occurred with less than a hundred.

3. Data sufficiency filter: Eliminate word-pair wp if $|\mathcal{F}_{wp}|$ is less than some constant K_{f_4}. This filter ensures that all word-pairs have sufficient features to be compared meaningfully.

4. Entropy filter: Eliminate word-pair wp if its *normalized entropy* is greater than some constant K_{f_5}. We compute a word-pair's entropy by considering it as a distribution over features, in a manner that is analogous to the *feature entropy* defined in (Turney, 2006). Specifically, the normalized entropy of a word-pair wp is:

$$-\frac{\sum_{f\in\mathcal{F}_{wp}} p(f|wp)\log\left(p(f|wp)\right)}{\log|\mathcal{F}_{wp}|}$$

where $p(f|wp) = \frac{O(wp,f)}{O(wp)}$ is the conditional probability of f occurring in the context of wp. The normalized entropy of a word-pair ranges from zero to one, and is at its highest when the distribution of the word-pair's occurrences over its features is the most random. The justification behind this filter is that word-pairs with strong underlying relations tend to have just a few dominant features that characterize those relations, whereas word-pairs that have many non-dominant features tend to have overly general underlying relations that can be characterized in many different ways.

3.2 Relation-Matching

Central to the problem of relation-matching is that of a *relational similarity function*: a function that computes the degree of similarity between two word-pairs' underlying relations. Given such a function, relation-matching reduces to simply computing the relational similarity between every pair of word-pairs, and outputting the pairs scoring higher than some threshold K_{th} as lexical analogies. Our system incorporates two relational similarity functions, as discussed in the following subsections.

Latent Relational Analysis

The baseline algorithm that we use to compute relational similarity is a modified version of Latent Relational Analysis (LRA) (Turney, 2006), that consists of the following steps:

1. Construct an n-by-m matrix A such that the ith row maps to word-pair wp_i, the jth column maps to feature f_j, and $A_{i,j} = O(wp_i, f_j)$.

2. Reduce the dimensionality of A to a constant K_{svd} using Singular Value Decomposition (SVD) (Golub and van Loan, 1996). SVD produces a matrix $\hat{A}$ of rank K_{svd} that is the best approximation of A among all matrices of rank K_{svd}. The use of SVD to compress the feature space was pioneered in Latent Semantic Analysis (Deerwester et al., 1990) and has become a popular technique in feature-based similarity computation. The compressed space is believed to be a semantic space that minimizes artificial surface differences.

3. The relational similarity between two word-pairs is the cosine measure of their corresponding row vectors in the reduced feature space. Specifically, let $\hat{A}_i$ denote the ith row vector of $\hat{A}$, then the relational similarity between word-pairs wp_{i_1} and wp_{i_2} is:

$$\frac{\hat{A}_{i_1}\cdot\hat{A}_{i_2}}{\left\|\hat{A}_{i_1}\right\|_2 + \left\|\hat{A}_{i_2}\right\|_2}$$

The primary difference between our algorithm and LRA is that LRA also includes each word's synonyms in the computation. Synonym inclusion greatly increases the size of the problem space, which leads to computational issues for our system as it operates at a much larger scale than previous work in relational similarity. Turney's (2006) extensive evaluation of LRA on SAT verbal analogy questions, for example, involves roughly ten thousand relational similarity computations[2]. In contrast, our system typically requires millions of relational similarity computations because every pair of extracted word-pairs needs to be compared. We call our algorithm LRA-S (LRA Without Synonyms) to differentiate it from the original LRA.

Similarity Graph Traversal

While LRA has been shown to perform well in computing relational similarity, it suffers from two

[2]The study evaluated 374 SAT questions, each involving 30 pairwise comparisons, for a total of 11220 relational similarity computations.

limitations. First, the use of SVD is difficult to interpret from an analytical point of view as there is no formal analysis demonstrating that the compressed space really corresponds to a semantic space. Secondly, even LRA-S does not scale up well to large data sets due to SVD being an expensive operation — computing SVD is in general $O(mn \cdot \min(m, n))$ (Koyuturk et al., 2005), where m, n are the number of matrix rows and columns, respectively.

To counter these limitations, we propose an alternative algorithm for computing relational similarity — Similarity Graph Traversal (SGT). The intuition behind SGT is as follows. Suppose we know that wp_1 and wp_2 are relationally similar, and that wp_2 and wp_3 are relationally similar. Then, by transitivity, wp_1 and wp_3 are also likely to be relationally similar. In other words, the relational similarity between two word-pairs can be reinforced by other word-pairs through transitivity. The actual algorithm involves the following steps:

1. Construct a *similarity graph* as follows. Each word-pair corresponds to a node in the graph. An edge exists from wp_1 to wp_2 if and only if the cosine measure of the two word-pairs' feature vectors is greater than or equal to some threshold K_{sgt}, in which case, the cosine measure is assigned as the *strength* of the edge.

2. Define a *similarity path* of length k, or k-*path*, from wp_1 to wp_2 to be a directed acyclic path of length k from wp_1 to wp_2, and define the *strength* $s(p)$ of a path p to be the product of the strength of all of the path's edges. Denote the set of all k-paths from wp_1 to wp_2 as $\mathcal{P}(k, wp_1, wp_2)$, and denote the sum of the strength of all paths in $\mathcal{P}(k, wp_1, wp_2)$ as $\mathcal{S}(k, wp_1, wp_2)$.

3. The relational similarity between word-pairs wp_{i_1} and wp_{i_2} is:

$$\begin{gathered}\alpha_1 \mathcal{S}(1, wp_1, wp_2) + \\ \alpha_2 \mathcal{S}(2, wp_1, wp_2) + \\ \cdots \\ \alpha_{K_l} \mathcal{S}(K_l, wp_1, wp_2)\end{gathered}$$

   where K_l is the maximum path length to consider, and α_1, $\ldots$, α_{K_l} are weights that are learned using least-squares regression on a small set of hand-labelled lexical analogies.

A natural concern for SGT is that relational similarity is not always transitive, and hence some paths may be invalid. For example, although (*teacher*, *student*) is relationally similar to both (*shepherd*, *sheep*) and (*boss*, *employee*), the latter two word-pairs are not relationally similar. The reason that this is not a problem for SGT is because truly similar word-pairs tend to be connected by many transitive paths, while invalid paths tend to occur in isolation. As such, while a single path may not be indicative, a collection of many paths likely signifies a true common relation. The weights in step 3 ensure that SGT assigns a high similarity score to two word-pairs only if there are sufficiently many transitive paths (which are sufficiently strong) between them.

Analogy Filters

As a final step in both LSA-R and SGT, we filter out lexical analogies of the form (w_1,w_2) and (w_1,w_3), as such lexical analogies tend to express the near-synonymy between w_2 and w_3 more than they express the relational similarity between the two word-pairs. We also keep only one *permutation* of each lexical analogy: (w_1,w_2) and (w_3,w_4), (w_3,w_4) and (w_1,w_2), (w_2,w_1) and (w_4,w_3), and (w_4,w_3) and (w_2,w_1) are different permutations of the same lexical analogy.

4 Evaluation

Our evaluation consisted of two parts. First, we evaluated the performance of the system, using LRA-S for relation-matching. Then, we evaluated the SGT algorithm, in particular, how it compares to LRA-S.

4.1 System Evaluation

Experimental Setup

We implemented our system in Sun JDK 1.5. We also used MXTerminator (Reynar and Ratnaparkhi, 1997) for sentence segmentation, MINIPAR (Lin, 1993) for lemmatization and dependency parsing, and MATLAB[3] for SVD computation. The experiment was conducted on a 2.1 GHz processor, with

[3]http://www.mathworks.com

the exception of SVD computation which was carried out in MATLAB running on a single 2.4 GHz processor within a 64-processor cluster. The input corpus consisted of the following collections in the Text Retrieval Conference Dataset[4]: AP Newswire 1988–1990, LA Times 1989–1990, and San Jose Mercury 1991. In total, 1196 megabytes of text data were used for the experiment. Table 1 summarizes the running times of the experiment.

Process	Time
Sentence Segmentation	20 min
Dependency Parsing	2232 min
Data Extraction	138 min
Relation-Matching	65 min

Table 1: Experiment Running Times

The parameter values selected for the experiment are listed in Table 2. The filter parameters were selected mostly through trial-and-error — various parameter values were tried and filtration results examined. We used a threshold value $K_{th} = 0.80$ to generate the lexical analogies, but the evaluation was performed at ten different thresholds from 0.98 to 0.80 in 0.02 decrements.

K_{f_1}	K_{f_2}	K_{f_3}	K_{f_4}	K_{f_5}	K_{svd}
35	10	100	10	0.995	600

Table 2: Experiment Parameter Values

Evaluation Protocol

An objective evaluation of our system is difficult for two reasons. First, lexical analogies are by definition subjective; what constitutes a 'good' lexical analogy is debatable. Secondly, there is no gold standard of lexical analogies to which we can compare. For these reasons, we adopted a subjective evaluation protocol that involved human judges rating the quality of the lexical analogies generated. Such a manual evaluation protocol, however, meant that it was impractical to evaluate the entire output set (which was well in the thousands). Instead, we evaluated random samples from the output and interpolated the results.

[4]http://trec.nist.gov/

In total, 22 human judges participated in the evaluation. All judges were graduate or senior undergraduate students in English, Sociology, or Psychology, and all were highly competent English speakers. Each judge was given a survey containing 105 lexical analogies, 100 of which were randomly sampled from our output, and the remaining five were sampled from a control set of ten human-generated lexical analogies. All entries in the control set were taken from the Verbal Analogy section of the Standard Aptitude Test[5] and represented the best possible lexical analogies. The judges were instructed to grade each lexical analogy with a score from zero to 10, with zero representing an invalid lexical analogy (i.e., when the two word-pairs share no meaningful underlying relation) and ten representing a perfect lexical analogy. To minimize inter-judge subjectivity, all judges were given detailed instructions containing the definition and examples of lexical analogies. In all, 1000 samples out of the 8373 generated were graded, each by at least two different judges.

We evaluated the output at ten threshold values, from 0.98 to 0.80 in 0.02 decrements. For each threshold, we collected all samples down to that threshold and computed the following metrics:

1. Coverage: The number of lexical analogies generated at the current threshold over the number of lexical analogies generated at the lowest threshold (8373).

2. Precision: The proportion of samples at the current threshold that scored higher than three. These are considered valid lexical analogies. Note that this is significantly more conservative than the survey scoring. We want to ensure very poor lexical analogies were excluded, even if they were 'valid' according to the judges.

3. Quality: The average score of all samples at the current threshold, divided by ten to be in the same scale as the other metrics.

4. Goodness: The proportion of samples at the current threshold that scored within 10% of the average score of the control set. These are considered human quality.

[5]http://www.collegeboard.com/

Note that recall was not an evaluation metric because there does not exist a method to determine the true number of lexical analogies in the input corpus.

Result

Table 3 summarizes the result of the control set, and Figure 2:Left summarizes the result of the lexical analogies our system generated. Table 4 lists some good and some poor lexical analogies our system generated, along with some of their shared features.

Coverage	Precision	Quality	Goodness
N/A	1.00	0.97	0.90

Table 3: Result of the Control Set

As Figure 2 shows, our system performed fairly well, generating valid lexical analogies with a precision around 70%. The quality of the generated lexical analogies was reasonable, although not at the level of human-generation. On the other hand, a small portion (19% at the highest threshold) of our output was of very high quality, comparable to the best human-generated lexical analogies.

Our result also showed that there was a correspondence between the score our system assigned to each generated lexical analogy and its quality. Precision, quality, and goodness all declined steadily toward lower thresholds: precision 0.70–0.66, quality 0.54–0.49, and goodness 0.19–0.14.

Error Analysis

Despite our aggressive filtration of irrelevant word-pairs and features, noise was still the most significant problem in our output. Most low-scoring samples contained at least one word-pair that did not have a meaningful and clear underlying relation; for examples, (*guy*, *ball*) and (*issue*, *point*). As mentioned, noise originated from mistakes in the input data, errors in sentence segmentation and parsing, as well as mismatches between dependencies and semantic relatedness. An example of the latter involved the frequent usage of the proposition "*of*" in various constructs. In the sentence "*the company takes advantage of the new legislation*", for example, the dependency structure associates *company* with *advantage*, whereas the semantic relation clearly lies between *company* and *legislation*. All three of our evaluation metrics (precision, quality, and goodness) were negatively affected by noise.

Polysemic words, as well as words which were heavily context-dependent, also posed a problem. For example, one of the lexical analogies generated in the experiment was (*resolution*, *house*) and (*legislation*, *senate*). This lexical analogy only makes sense if "*house*" is recognized as referring to the House of Representatives, which is often abbreviated as "*the House*" in news articles. Polysemy also negatively affected all three of our evaluation metrics, although to a lesser extent for precision.

Finally, our system had difficulties differentiating semantic relations of different granularity. The underlying relations of (*relation*, *country*) and (*tie*, *united states*), for example, are similar, yet they do not form a good lexical analogy because the relations are at different levels of granularity (countries in general in the former, and a particular country in the latter). Undifferentiated granularity affected quality and goodness, but it did not have a significant effect on precision.

4.2 SGT Evaluation

To evaluate how SGT compares to LRA-S, we repeated the experiment using SGT for relation-matching. We set K_l (maximum path length) to 3, and K_{sgt} (cosine threshold) to 0.2; these values were again determined largely through trial-and-error. To train SGT, we used 90 lexical analogies graded by human judges from the previous experiment. In order to facilitate a fair comparison to LRA-S, we selected K_{th} values that allowed SGT to generate the same number of lexical analogies as LRA-S did at each threshold interval.

Running on the same 2.1 GHz processor, SGT finished in just over eight minutes, which is almost a magnitude faster than LRA-S' 65 minutes. SGT also used significantly less memory, as the similarity graph was efficiently stored in an adjacency list. The sets of lexical analogies generated by the two algorithms were quite similar, overlapping approximately 50% at all threshold levels.

The significant overlap between SGT and LRA-S' outputs allowed us to evaluate SGT using the samples collected from the previous surveys instead of conducting a new round of human grading. Specifically, we identified previously graded samples that

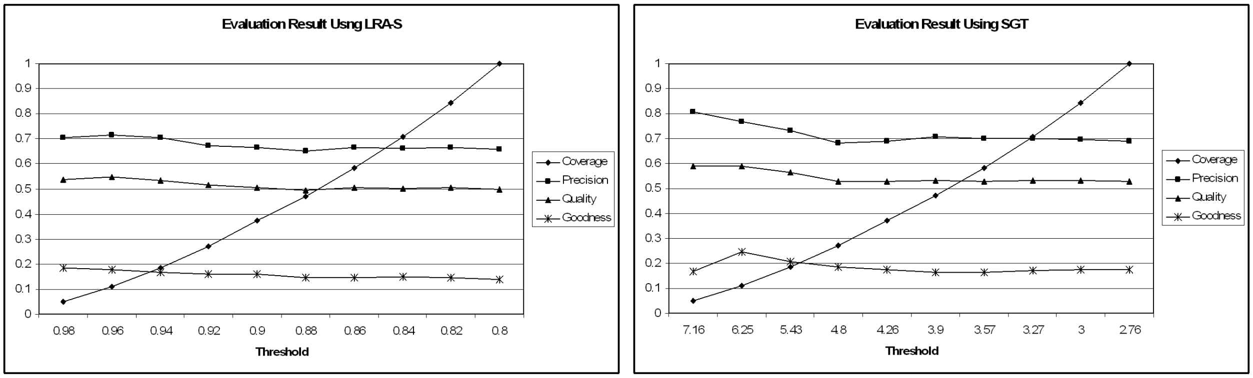

Figure 2: System Evaluation Results

Good Examples	Shared Features
(*vietnam, cambodia*) and (*iraq, kuwait*)	$\_\_\_ \xleftarrow{subj} \textit{invade} \xrightarrow{obj} \_\_\_,\ \_\_\_ \xleftarrow{subj} \textit{pull out} \xrightarrow{of} \_\_\_$
(*building, office*) and (*museum, collection*)	$\_\_\_ \xleftarrow{subj} \textit{house} \xrightarrow{obj} \_\_\_,\ \_\_\_ \xleftarrow{subj} \textit{consolidate} \xrightarrow{obj} \_\_\_$
(*stock market, rally*) and (*student, march*)	$\_\_\_ \xleftarrow{subj} \textit{stage} \xrightarrow{obj} \_\_\_$
(*researcher, experiment*) and (*doctor, surgery*)	$\_\_\_ \xleftarrow{subj} \textit{perform} \xrightarrow{obj} \_\_\_$
(*gainer, loser*) and (*decline, advance*)	$\_\_\_ \xleftarrow{subj} \textit{outnumber} \xrightarrow{obj} \_\_\_$
(*book, shelf*) and (*picture, wall*)	$\_\_\_ \xleftarrow{with} \textit{line} \xrightarrow{subj} \_\_\_,\ \_\_\_ \xleftarrow{subj} \textit{remain} \xrightarrow{on} \_\_\_$
(*blast, car*) and (*sanction, economy*)	$\_\_\_ \xleftarrow{subj} \textit{damage} \xrightarrow{obj} \_\_\_,\ \_\_\_ \xleftarrow{by} \textit{destroy} \xrightarrow{subj} \_\_\_$
Poor Examples	**Shared Features**
(*president, change*) and (*bush, legislation*)	$\_\_\_ \xleftarrow{subj} \textit{veto} \xrightarrow{obj} \_\_\_$
(*charge, death*) and (*lawsuit, federal court*)	$\_\_\_ \xleftarrow{subj} \textit{file} \xrightarrow{in} \_\_\_$
(*relation, country*) and (*tie, united states*)	$\_\_\_ \xleftarrow{obj} \textit{severe} \xrightarrow{subj} \_\_\_$
(*judge, term*) and (*member, life*)	$\_\_\_ \xleftarrow{subj} \textit{sentence} \xrightarrow{to} \_\_\_$
(*issue, point*) and (*stock, cent*)	$\_\_\_ \xleftarrow{subj} \textit{be} \xrightarrow{down} \_\_\_,\ \_\_\_ \xleftarrow{subj} \textit{be} \xrightarrow{up} \_\_\_$

Table 4: Examples of Good and Poor Lexical Analogies Generated

had also been generated by SGT, and used these samples as the evaluation data points for SGT. At the lowest threshold (where 8373 lexical analogies were generated), we were able to reuse 533 samples out of the original 1000 samples. Figure 2:Right summarizes the performance of the system using SGT for relation-matching.

As the figure shows, SGT performed very similarly to LRA-S. Both SGT's precision and quality scores were slightly higher than LRA-S, but the differences were very small and hence were likely due to sample variation. The goodness scores between the two algorithms were also comparable. In the case of SGT, however, the score fluctuated instead of monotonically decreased. We attribute the fluctuation to the smaller sample size.

As the samples were drawn exclusively from the portion of SGT's output that overlapped with LRA-S' output, we needed to ensure that the samples were not strongly biased and that the reported result was not better than SGT's actual performance. To validate the result, we conducted an additional experiment involving a single human judge. The judge was given a survey with 50 lexical analogies, 25 of which were sampled from the overlapping portion of SGT and LRA-S' outputs, and 25 from lexical analogies generated only by SGT. Table 5 summarizes the result of this experiment. As the table demonstrates, the results from the two sets were comparable with small differences. Moreover, the differences were in favour of the SGT-only portion. Therefore, either there was no sampling bias at all, or the sampling bias negatively affected the result. As such, SGT's actual performance was at least as good as reported, and may have been slightly higher.

	Precision	Quality	Goodness
Overlap	0.76	0.56	0.28
SGT-Only	0.88	0.62	0.2

Table 5: Overlap vs. SGT-Only

We conclude that SGT is indeed a viable alternative to LRA-S. SGT generates lexical analogies that are of the same quality as LRA-S, while being significantly faster and more scalable. On the other hand, an obvious limitation of SGT is that it is a supervised algorithm requiring manually labelled training data. We claim this is not a severe limitation because there are only a few variables to train (i.e., the weights), hence only a small set of training data is required. Moreover, a supervised algorithm can be advantageous in some situations; for example, it is easier to tailor SGT to a particular input corpus.

5 Related Work

The study of analogy in the artificial intelligence community has historically focused on computational models of analogy-making. French (2002) and Hall (1989) provide two of the most complete surveys of such models. Veale (2004; 2005) generates lexical analogies from WordNet (Fellbaum, 1998) and HowNet (Dong, 1988) by dynamically creating new type hierarchies from the semantic information stored in these lexicons. Unlike our corpus-based generation system, Veale's algorithms are limited by the lexicons in which they operate, and generally are only able to generate *near-analogies* such as (*Christian*, *Bible*) and (*Muslim*, *Koran*). Turney's (2006) Latent Relational Analysis is a corpus-based algorithm that computes the relational similarity between word-pairs with remarkably high accuracy. However, LRA is focused solely on the relation-matching problem, and by itself is insufficient for lexical analogy generation.

6 Conclusion and Future Work

We have presented a system that is, to the best of our knowledge, the first system capable of generating lexical analogies from unstructured text data. Empirical evaluation shows that our system performed fairly well, generating valid lexical analogies with a precision of about 70%. The quality of the generated lexical analogies was reasonable, although not at the level of human performance. As part of the system, we have also developed a novel algorithm for computing relational similarity that rivals the performance of the current state-of-the-art while being significantly faster and more scalable. One of our immediate tasks is to complement dependency patterns with additional features. In particular, we expect semantic features such as word definitions from machine-readable dictionaries to improve our system's ability to differentiate between different senses of polysemic words, as well as different granularities of semantic relations. We also plan to take advantage of our system's flexibility and relax the constraints on dependency paths so as to generate more-varied lexical analogies, e.g., analogies involving verbs and adjectives.

A potential application of our system, and the original inspiration for this research, would be to use the system to automatically enrich ontologies by spreading semantic relations between lexical analogues. For example, if words w_1 and w_2 are related by relation r, and (w_1, w_2) and (w_3, w_4) form a lexical analogy, then it is likely that w_3 and w_4 are also related by r. A dictionary of lexical analogies therefore would allow an ontology to grow from a small set of seed relations. In this way, lexical analogies become bridges through which semantic relations flow in a sea of ontological concepts.

Acknowledgments

We thank the reviewers of EMNLP 2007 for valuable comments and suggestions. This work was supported in part by the Ontario Graduate Scholarship Program, Ontario Innovation Trust, Canada Foundation for Innovation, and the Natural Science and Engineering Research Council of Canada.

References

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Dong Zhen Dong. 1988. What, how and who? *Proceedings of the International Symposium on Electronic Dictionaries*. Tokyo, Japan.

Christine Fellbaum, editor. 1998. *WordNet — An Electronic Lexical Database*. MIT Press.

Robert French. 2002. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200–205.

Gene Golub and Charles van Loan. 1996. *Matrix Computations*. Johns Hopkins University Press, third edition.

Rogers Hall. 1989. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120.

Mehmet Koyuturk, Ananth Grama, and Naren Ramakrishnan. 2005. Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):447–461.

Beth Levin 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Dekang Lin. 1993. Principle-based parsing without overgeneration. *Proceedings of the 31st Annual Meeting on ACL*, pp 112–120. Columbus, USA.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Jane Morris and Graeme Hirst. 2004. Non-classical lexical semantic relations. *Proceedings of the Computational Lexical Semantics Workshop at HLT-NAACL 2004*, pp 46–51. Boston, USA.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp 16–19. Washington, USA.

Gerard Salton, A. Wong, and C.S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 13(11):613–620.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Proceedings of the 2004 Neural Information Processing Systems Conference*. Vancouver, Canada.

Lucien Tesnière. 1959. *Éléments de Syntaxe Structurale*. Librairie C. Klincksieck, Paris.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Tony Veale, Jer Hayes, and Nuno Seco. 2004. The Bible is the Christian Koran: Discovering simple analogical compounds. *Proceedings of the Workshop on Computational Creativity in 2004 European Conference on Case-Based Reasoning*. Madrid, Spain.

Tony Veale. 2005. Analogy generation with HowNet. *Proceedings of the 2005 International Joint Conference on Artificial Intelligence*. Edinburgh, Scotland.