Polarity Inducing Latent Semantic Analysis

Wen-tau Yih **Geoffrey Zweig** **John C. Platt**
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{scottyih,gzweig,jplatt}@microsoft.com

Abstract

Existing vector space models typically map synonyms and antonyms to similar word vectors, and thus fail to represent antonymy. We introduce a new vector space representation where antonyms lie on opposite sides of a sphere: in the word vector space, synonyms have cosine similarities close to one, while antonyms are close to minus one.

We derive this representation with the aid of a thesaurus and latent semantic analysis (LSA). Each entry in the thesaurus – a word sense along with its synonyms and antonyms – is treated as a "document," and the resulting document collection is subjected to LSA. The key contribution of this work is to show how to assign signs to the entries in the co-occurrence matrix on which LSA operates, so as to induce a subspace with the desired property.

We evaluate this procedure with the Graduate Record Examination questions of (Mohammad et al., 2008) and find that the method improves on the results of that study. Further improvements result from refining the subspace representation with discriminative training, and augmenting the training data with general newspaper text. Altogether, we improve on the best previous results by 11 points absolute in F measure.

1 Introduction

Vector space representations have proven useful across a wide variety of text processing applications ranging from document clustering to search relevance measurement. In these applications, text is represented as a vector in a multi-dimensional continuous space, and a similarity metric such as cosine similarity can be used to measure the relatedness of different items. Vector space representations have been used both at the document and word levels. At the document level, they are effective for applications including information retrieval (Salton and McGill, 1983; Deerwester et al., 1990), document clustering (Deerwester et al., 1990; Xu et al., 2003), search relevance measurement (Baeza-Yates and Ribeiro-Neto, 1999) and cross-lingual document retrieval (Platt et al., 2010). At the word level, vector representations have been used to measure word similarity (Deerwester et al., 1990; Turney and Littman, 2005; Turney, 2006; Turney, 2001; Lin, 1998; Agirre et al., 2009; Reisinger and Mooney, 2010) and for language modeling (Bellegarda, 2000; Coccaro and Jurafsky, 1998). While quite successful, these applications have typically been consistent with a very general notion of similarity in which basic association is measured, and finer shades of meaning need not be distinguished. For example, latent semantic analysis might assign a high degree of similarity to opposites as well as synonyms (Landauer and Laham, 1998; Landauer, 2002).

Independent of vector-space representations, a number of authors have focused on identifying different kinds of relatedness. At the simplest level, we may wish to distinguish between synonyms and antonyms, which can be further differentiated. For example, in synonymy, we may wish to distinguish hyponyms and hypernyms. Moreover, Cruse (1986) notes that numerous kinds of antonymy are possible, for example *antipodal pairs* like "top-bottom" or *gradable opposites* like "light-heavy." Work in this area includes (Turney, 2001; Lin et al., 2003; Turney and Littman, 2005; Turney, 2006; Curran and Moens, 2002; van der Plas and Tiedemann, 2006; Mohammad et al., 2008; Mohammad et al., 2011). Despite the existence of a large amount of related work in the literature, distinguishing synonyms and antonyms is still considered as a difficult open problem in general (Poon and Domingos, 2009).

In this paper, we fuse these two strands of research in an attempt to develop a vector space representation in which the synonymy and antonymy are naturally differentiated. We follow Schwab et al. (2002) in requiring a representation in which two lexical items in an antonymy relation should lie at opposite ends of an axis. However, in contrast to the logical axes used previously, we desire that antonyms should lie at the opposite ends of a sphere lying in a continuous and automatically induced vector space. To generate this vector space, we present a novel method for assigning both *negative* and *positive* values to the TF-IDF weights used in latent semantic analysis.

To determine these signed values, we exploit the information present in a thesaurus. The result is a vector space representation in which synonyms cluster together, and the opposites of a word tend to cluster together at the opposite end of a sphere.

This representation provides several advantages over the raw thesaurus. First, by finding the items most and least similar to a word, we are able to discover new synonyms and antonyms. Second, as discussed in Section 5, the representation provides a natural starting point for gradient-descent based optimization. Thirdly, as we discuss in Section 6, it is straightforward to embed *new words* into the derived subspace by using information from a large unsupervised text corpus such as Wikipedia.

The remainder of this paper is organized as follows. Section 2 describes previous work. Section 3 presents the classical LSA approach and analyzes some of its limitations. In Section 4 we present our polarity inducing extension to LSA. Section 5 further extends the approach by optimizing the vector space representation with supervised discriminative training. Section 6 describes the proposed method of embedding new words in the thesaurus-derived subspace. The experimental results of Section 7 indicate that the proposed method outperforms previous approaches on a GRE test of closest-opposites (Mohammad et al., 2008). Finally, Section 8 concludes the paper.

2 Related Work

The detection of antonymy has been studied in a number of previous papers. Mohammad et al. (2008) approach the problem by combining information from a published thesaurus with corpus statistics derived from the Google n-gram corpus (Brants and Franz, 2006). Their method consists of two main steps: first, detecting contrasting word categories (e.g. "WORK" vs. "ACTIVITY FOR FUN") and then determining the degree of antonymy. Categories are defined by a thesaurus; contrasting categories are found by using affix rules (e.g., un- & dis-) and WordNet antonymy links. Words belonging to contrasting categories are treated as antonyms and the degree of contrast is determined by distributional similarity. Mohammad et al. (2008) also provides a publicly available dataset for detection of antonymy, which we have adopted. This work has been extended in (Mohammad et al., 2011) to include a study of antonymy based on crowd-sourcing experiments.

Turney (2008) proposes a unified approach to handling analogies, synonyms, antonyms and associations by transforming the last three cases into cases of analogy. A supervised learning method is then used to solve the resulting analogical problems. This is evaluated on a set of 136 ESL questions. Lin et al. (2003) builds on (Lin, 1998) and identifies antonyms as semantically related words which also happen to be found together in a database in pre-identified phrases indicating opposition. Lin et al. (2003) further note that whereas synonyms will tend to translate to the same word in another language, antonyms will not. This observation is used to select antonyms from amongst distributionally similar words. Antonymy is used in (de Simone and Kazakov, 2005) for document clustering and (Harabagiu et al., 2006) to find contradiction.

The automatic detection of synonyms has been more extensively studied. Lin (1998) presents a thorough comparison of word-similarity metrics based on distributional similarity, where this is determined from co-occurrence statistics in dependency triples extracted by parsing a large dataset. Related studies are described in (Curran and Moens, 2002; van der Plas and Bouma, 2005). Later, van der Plas and Tiedemann (2006) extend the use of multilingual data present in Lin et al. (2003) by measuring distributional similarity based on the contexts that a word occurs in *once translated* into a new language. This is used to improve the precision/recall characteristics on synonym pairs. Structured information can be important in determining relatedness, and thesauri and Wikipedia links have been studied in (Milne and Witten, 2008; Jarmasz and Szpakowicz, 2003). Combinations of approaches are studied in (Turney et al., 2003).

Vector-space models and latent semantic analysis in particular have a long history of use in synonym detection, which in fact was suggested in some of the earliest LSA papers. Deerwester et al. (1990) defines a metric for measuring word similarity based on LSA, and it has been used in (Landauer and Dumais, 1997; Landauer et al., 1998) to answer word similarity questions derived from the Test of English as a Foreign Language (TOEFL). Turney (2001) proposes the use of point-wise mutual information in conjunction with LSA, and again presents results on synonym questions derived from the TOEFL. Variants of vector space models are further analyzed in (Turney and Littman, 2005; Turney, 2006; Turney and Pantel, 2010).

3 Latent Semantic Analysis

Latent Semantic Analysis (Deerwester et al., 1990) is a widely used method for representing words and documents in a low dimensional vector space. The method is based on applying singular value decomposition (SVD) to a matrix W which indicates the occurrence of words in documents. To perform LSA, one proceeds as follows. The input is a collection of d documents which are expressed in terms of words from a vocabulary of size n. These documents may be actual documents such as newspaper articles, or simply notional documents such as sentences, or any other collection in which words are grouped together. Next, a $d \times n$ document-term matrix W is formed[1]. At its simplest form, the $ij^{\texttt{th}}$ entry contains the number of times word j has occurred in document i – its *term frequency* or TF value. More conventionally, the entry is weighted by some notion of the importance of word j, for example the negative logarithm of the fraction of documents that contain it, resulting in a TF-IDF weighting (Salton et al., 1975). The similarity between two documents can be computed using the cosine similarity of their corresponding row vectors:

$$\texttt{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\parallel \mathbf{x} \parallel \parallel \mathbf{y} \parallel}$$

Similarly, the cosine similarity of two column vectors can be used to judge the similarity of the corresponding words. Finally, to obtain a subspace representation of dimension k, W is decomposed as

$$W \approx USV^T$$

where U is $d \times k$, V^T is $k \times n$, and S is a $k \times k$ diagonal matrix. In applications, $k \ll n$ and $k \ll d$; for example one might have a $50,000$ word vocabulary and $1,000,000$ documents and use a 300 dimensional subspace representation.

An important property of SVD is that the columns of SV^T – which now represent the words – behave similarly to the original columns of W, in the sense that the cosine similarity between two columns in SV^T approximates the cosine similarity between the corresponding columns in W. This follows from the observation that $W^TW = VS^2V^T$, and the fact that the $ij^{\texttt{th}}$ entry of W^TW is the dot product of the $i^{\texttt{th}}$ and $j^{\texttt{th}}$ columns (words) in W. We will use this observation subsequently in the derivation of polarity-inducing LSA. For efficiency, we normalize the columns of SV^T to unit length, allowing the cosine similarity between two words to be computed with a single dot-product; this also has the property of mapping each word to a point on a multi-dimensional sphere.

A second important property of LSA is that in the word representations which result can by viewed as the result of applying a projection matrix U to the original vectors as:

$$U^TW = SV^T$$

[1](Bellegarda, 2000) constructs the transpose of this, but we have found it convenient in data processing for documents to represent rows.

In Section 5, we will view U simply as a $d \times k$ matrix learned through gradient descent so as to optimize an objective function.

3.1 Limitation of LSA

Word similarity as determined by LSA assigns high values to words which tend to co-occur in documents. However, as noted by (Landauer and Laham, 1998; Landauer, 2002), there is no notion of antonymy; words with low or negative cosine scores are simply unrelated. In comparison, words with high cosine similarity scores are typically *semantically* related, which includes both synonyms and antonyms, as contrasting words often co-occur (Murphy and Andrew, 1993; Mohammad et al., 2008). To illustrate this, we have performed SVD with the aid of the Encarta thesaurus developed by Bloomsbury Publishing Plc. This thesaurus contains approximately 47k word senses and a vocabulary of 50k words and phrases. Each "document" is taken to be the thesaurus entry for a word-sense, including synonyms and antonyms. For example, the word "admirable" induces a document consisting of {admirable, estimable, commendable, venerable, good, splendid, worthy, marvelous, excellent, *unworthy*}. Note that the last word in this set is its antonym. Performing SVD on this set of thesaurus derived "meaning-documents" results in a subspace representation for each word. This form of LSA is similar to the use of Wikipedia in (Gabrilovich and Markovitch, 2007).

Table 1 shows some words, their original thesaurus documents, and the most and least similar words in the LSA subspace. Several properties are apparent:

- The vector-space representation of words is able to identify related words that are not explicitly present in the original thesaurus. For example, "meritorious" for "admirable" – arguably better than any of the words given in the thesaurus itself.
- Similarity is based on co-occurrence, so the co-occurrence of antonyms in the thesaurus-derived documents induces their presence as LSA-similar words. For example, "contemptible" is identified as similar to "admirable." In the case of "mourning," opposites such as "joy" and "elation" actually dominate the list of LSA-similar words.
- The LSA-least-similar words have no relationship at all to the word they are least-similar to. For example, the least-similar word to "considered" is "ready-made-meal."

In the next section, we will present a method for inducing polarity in LSA subspaces, where opposite words will tend to have negative cosine similarities, analogous to the positive similarities of synonyms. Thus, the least-similar words to a given word will be its opposites.

	acrimony	rancor	goodwill	affection
acrimony	1	1	1	1
affection	1	1	1	1

Table 2: The W matrix for two thesaurus entries in its original form. Rows represent documents; columns words.

	acrimony	rancor	goodwill	affection
acrimony	1	1	-1	-1
affection	-1	-1	1	1

Table 3: The W matrix for two thesaurus entries in its polarity-inducing form.

4 Polarity Inducing LSA

We modify LSA so that we may exploit a thesaurus to embed meaningful axes in the induced subspace representation. Words with opposite meaning will lie at opposite positions on a sphere. Recall that the cosine similarity between word-vectors in the original matrix W are preserved in the subspace representation of words. Thus, if we construct the original matrix so that the columns representing antonyms will tend to have negative cosine similarities while columns representing synonyms will tend to have positive similarities, we will achieve the desired behavior.

This can be achieved by negating the TF-IDF entries for the antonyms of a word when constructing W from the thesaurus, which is illustrated in Tables 2 and 3. The two rows in these tables correspond to thesaurus entries for the sense-categories

Word	Thesaurus Entry	LSA Most-Similar Words	LSA Least-Similar Words
admirable	estimable, commendable, venerable, good, splendid, worthy, marvelous, excellent, *unworthy*	commendable, creditable, laudable, praiseworthy, worthy, meritorious, scurvy, contemptible, despicable, estimable	easy-on-the-eye, peacekeeper, peace-lover, conscientious-objector, uninviting, dishy, dessert, pudding, seductive
considered	careful, measured, well-thought-out, painstaking, *rash*	calculated, premeditated, planned, tactical, strategic, thought-through, intentional, fortuitous, purposeful, unpremeditated	ready-made-meal, ready-meal, disposed-to, apt-to, wild-animals, big-game, game-birds, game-fish, rugger, rugby
mourning	grief, bereavement, sorrow, sadness, lamentation, woe, grieving, *exultation*	sorrowfulness, anguish, exultation, rejoicing, jubilation, glee, heartache, travail, joy, elation	muckiness, turn-the-corner, impassibility, filminess, pellucidity, limpidity, sheerness

Table 1: LSA on a thesaurus. Thesaurus entries include antonyms in italics.

"acrimony," and "affection." The thesaurus entries induce two "documents" containing the words and their synonyms and antonyms. The complete set of words is *acrimony, rancor, goodwill, affection*. For simplicity, we assume that all TF-IDF weights are 1. In the original LSA formulation, we have the representation of Table 2. "Rancor" is listed as a synonym of "acrimony," which has "goodwill" and "affection" as its antonyms. This results in the first row. Note that the cosine similarity between every pair of words (columns) is 1.

Table 3 shows the polarity-inducing representation. Here, the cosine similarity between synonymous words (columns) is 1, and the cosine similarity between antonymous words is -1. Since LSA tends to preserve cosine similarities between words, in the resulting subspace we may expect to find meaningful axes, where opposite senses map to opposite extremes. We refer to this as polarity-inducing LSA or PILSA.

In Table 4, we show the PILSA-similar and PILSA-least-similar words for the same words as in Table 1. We see now that words which are least similar in the sense of having the lowest cosine-similarity are indeed opposites. In this table generally the most similar words have similarities in the range of 0.7 to 1.0 and the least similar in the range of -0.7 to -1.0.

5 Discriminative Training

Although the cosine similarity of LSA-derived word vectors are generally very effective in applications such as judging the relevance of words or documents, or detecting antonyms as in our construction, the process of singular value decomposition in LSA does not explicitly try to achieve such goals. In this section, we see that when supervised training data is available, the projection matrix of LSA can be enhanced through a discriminative training technique explicitly designed to create a representation suited to a specific task.

Because LSA is closely related to principle component analysis (PCA), extensions of PCA such as canonical correlation analysis (CCA) and oriented principle component analysis (OPCA) can leverage the labeled data and produce the projection matrix through general eigen-decomposition (Platt et al., 2010). Along this line of work, Yih et al. (2011) proposed a Siamese neural network approach called *S2Net*, which tunes the projection matrix directly through gradient descent, and has shown to outperform other methods in several tasks. Below we describe briefly this technique and explain how we adopt it for the task of antonym detection.

The goal of S2Net is to learn a concept vector representation of the original sparse term vectors. Although such transformation can be non-linear in general, its current design chooses the model form to be a linear projection matrix, which is identical to

Word	PILSA-Similar Words	PILSA-Least-Similar Words
admirable	commendable, creditable, laudable, praiseworthy, worthy, meritorious, estimable, deserving, tiptop, valued	scurvy, contemptible, despicable, lamentable, shameful, reprehensible, unworthy, disgraceful, discreditable, undeserving
considered	calculated, premeditated, planned, tactical, strategic, thought-through, intentional, purposeful, intended, psychological	fortuitous, unpremeditated, unconsidered, off-your-own-bat, unintended, undirected, objectiveless, hit-and-miss, unforced, involuntary
mourning	sorrowful, doleful, sad, miserable, wistful, pitiful, wailing, sobbing, heavy-hearted, forlorn	smiley, happy, blissful, wooden, mirthful, joyful, deadpan, fulfilled, straight-faced, content

Table 4: PILSA on a thesaurus. Thesaurus entries are as in Table 1.

that of LSA, PCA, OPCA or CCA. Given a d-by-1 input vector $\mathbf{f}$, the model of S2Net is a d-by-k matrix $\mathbf{A} = [a_{ij}]_{d \times k}$, which maps $\mathbf{f}$ to a k-by-1 output vector $\mathbf{g} = \mathbf{A}^T \mathbf{f}$. The fact that the transformation can be viewed as a two-layer neural network leads to the method's name.

What differentiates S2Net from other approaches is its loss function and optimization process. In the "parallel text" setting, the labeled data consists of pairs of *similar* text objects such as documents. The objective of the training process is to assign higher cosine similarities to these pairs compared to others. More specifically, suppose the training set consists of m pairs of raw input vectors $\{(\mathbf{f}_{p_1}, \mathbf{f}_{q_1}), (\mathbf{f}_{p_2}, \mathbf{f}_{q_2}), \cdots, (\mathbf{f}_{p_m}, \mathbf{f}_{q_m})\}$. Given a projection matrix $\mathbf{A}$, the similarity score of any pair of objects is $sim_{\mathbf{A}}(\mathbf{f}_{p_i}, \mathbf{f}_{q_j}) = cosine(\mathbf{A}^T \mathbf{f}_{p_i}, \mathbf{A}^T \mathbf{f}_{q_j})$. Let $\Delta_{ij} = sim_{\mathbf{A}}(\mathbf{f}_{p_i}, \mathbf{f}_{q_i}) - sim_{\mathbf{A}}(\mathbf{f}_{p_i}, \mathbf{f}_{q_j})$ be the difference of the similarity scores of $(\mathbf{f}_{p_i}, \mathbf{f}_{q_i})$ and $(\mathbf{f}_{p_i}, \mathbf{f}_{q_j})$. The learning procedure tries to increase Δ_{ij} by using the following logistic loss:

$$L(\Delta_{ij}; \mathbf{A}) = \log(1 + exp(-\gamma \Delta_{ij})),$$

where γ is a scaling factor that adjusts the loss function[2]. The loss of the whole training set is thus:

$$\frac{1}{m(m-1)} \sum_{1 \le i,j \le m, i \ne j} L(\Delta_{ij}; \mathbf{A})$$

Parameter learning (i.e., tuning $\mathbf{A}$) can be done by standard gradient-based methods, such as L-BFGS (Nocedal and Wright, 2006).

The original setting of S2Net can be directly applied to finding synonymous words, where the training data consists of pairs of vectors representing two synonyms. It is also easy to modify the loss function to apply it to the antonym detection problem. We first sample pairs of antonyms from the thesaurus to create the training data. The raw input vector $\mathbf{f}$ of a selected word is its corresponding column vector of the document-term matrix W (Section 3) after inducing polarity (Section 4). When each pair of vectors in the training data represents two antonyms, we can redefine Δ_{ij} by flipping the sign: $\Delta_{ij} = sim_{\mathbf{A}}(\mathbf{f}_{p_i}, \mathbf{f}_{q_j}) - sim_{\mathbf{A}}(\mathbf{f}_{p_i}, \mathbf{f}_{q_i})$, and leave others unchanged. As the loss function encourages Δ_{ij} to be larger, an antonym pair will tend to have a smaller cosine similarity than other pairs. Because S2Net uses a gradient descent technique and a non-convex objective function, it is sensitive to initialization, and we have found that the PILSA projection matrix U (Section 3) provides an excellent starting point. As illustrated in Section 7, learning the word vectors with S2Net produces a significant improvement over PILSA alone.

6 Extending PILSA to Out-of-thesaurus Words

While PILSA is effective at representing synonym and antonym information, in its pure form, it is limited to the vocabulary of the thesaurus. In order to extend PILSA to operate on out-of-thesaurus words,

[2]As suggested in (Yih et al., 2011), γ is set to 10 in our experiments.

we employ a two-stage strategy. We first conduct some lexical analysis and try to match an unknown word to one or more in-thesaurus words in their lemmatized forms. If no such match can be found, we then attempt to find semantically related in-thesaurus words by leveraging co-occurrence statistics from general text data. These two steps are described in detail below.

6.1 Matching via Lexical Analysis

When a target word is not included in a thesaurus, it is quite often that some of its morphological variations are covered. For example, although the Encarta thesaurus does not have the word "corruptibility," it does contain other forms like "corruptible" and "corruption." Replacing the out-of-thesaurus target word with these morphological variations may alter the part-of-speech but typically does not change the meaning[3].

Given an out-of-thesaurus target word, we first apply a morphological analyzer for English developed by Minnen et al. (2001), which removes the inflectional affixes and returns the lemma. If the lemma still does not exist in the thesaurus, we then apply Porter's stemmer (Porter, 1980) and check whether the target word can match any of the in-thesaurus words in their stemmed forms. A simple rule that checks whether removing hyphens from words can lead to a match and whether the target word occurs as part of a compound word in the thesaurus is applied when both morphological analysis and stemming fail to find a match. When there are more than one matched words, the centroid of their PILSA vectors is used to represent the target word. When there is only one matched word, the matched word is treated as the target word.

6.2 Leveraging General Text Data

If no words in the thesaurus can be linked to the target word through the simple lexical analysis procedure, we try to find matched words by creating a *context* vector space model from a large document collection, and then mapping from this space to the PILSA space. We use contexts because of the *distributional hypothesis* – words that occur in the same *contexts* tend to have similar meaning (Harris, 1954). When a word is not in the thesaurus but appears in the corpus, we predict its PILSA vector representation from the context vector space model by using its k-nearest neighbors which are in the thesaurus and *consistent* with each other.

6.2.1 Context Vector Space Model

Given a corpus of documents, we construct the raw context vectors as follows. For each target word, we first create a bag of words by collecting all the terms within a window of [-10,+10] centered at each occurrence of the target word in the corpus. The non-identical terms form a term-vector, where each term is weighted using its TF-IDF value. We then perform LSA on the context-word matrix. The semantic similarity/relatedness of two words can then be determined using the cosine similarity of their corresponding LSA word vectors. In the following text, we refer to this LSA context vector space model as the *corpus* space, in contrast to the PILSA *thesaurus* space.

6.2.2 Embedding Out-of-Vocabulary Words

Given the context space model, we may use a linear regression or a k-nearest neighbors approach to embed out-of-thesaurus words into the thesaurus-space representation. However, as near words in the context space may be synonyms in addition to other semantically related words (including antonyms), such approaches can potentially be noisy. For example, words like "hot" and "cold" may be close to each other in the context space due to their similar usage in text. An affine transform cannot "tear space" and map them to opposite poles in the thesaurus space.

Therefore, we propose a revised k-nearest neighbors approach. Suppose we are interested in an out-of-thesaurus word w. We first find K-nearest in-thesaurus neighbors to w in the context space. We then select a subset of k members of these K words such that the pairwise similarity of each of the k members with every other member is positive. The thesaurus-space centroid of these k items is computed as w's representation. This procedure has the property that the k nearby words used to form the embedding of a non-thesaurus word are selected to be consistent with each other. In practice, we used $K = 10$ and $k = 3$, which requires only around

[3]The rules we use based on lexical analysis are moderately conservative to avoid mistakes like mapping *hopeless* to *hope*.

1000 pairwise computations even done in a brute-force way. To provide a concrete example, if we had the out-of-thesaurus word “sweltering” with in-thesaurus neighbors “hot, cold, burning, scorching, ...” the procedure would return the centroid of “hot, burning, scorching” and exclude “cold.”

7 Experimental Validation

In this section, we present our experimental results on applying PILSA and its extensions to answering the closest-opposite GRE questions.

7.1 Data Resources

The primary thesaurus we use is the Encarta Thesaurus developed by Bloomsbury Publishing Plc[4]. Our version of this has approximately 47k word senses and a vocabulary of 50k words, and contains 125,724 pairs of antonyms. To experiment with the effect of using a different thesaurus, we used WordNet as an information source. Each *synset* in WordNet maps to a row in the document-term matrix; synonyms in a synset are weighted with positive TFIDF values, and antonyms are weighted negative TFIDF values. Entries corresponding to other words in the vocabulary are 0. WordNet provides significantly greater coverage with approximately 227k synsets involving multiple words, and a vocabulary of about 190k words. However, it is also much sparser, with 5.3 words per sense on average as opposed to 10.3 in the thesaurus, and has only 62,821 pairs of antonyms. As general text data for use in embedding out-of-vocabulary words, we used a Nov-2010 dump of English Wikipedia, which contains approximately 917M words.

7.2 Development and Test Data

For testing, we use the closest-opposite questions from GRE tests provided by (Mohammad et al., 2008). Each question contains a target word and five choices, and asks which of the choice words has the most opposite meaning to the target word. Two datasets are made publicly available by Mohammad et al. (2008): the *development set*, which consists of 162 questions, and the *test set*, which has 950 questions[5]. We considered making our own, more extensive, test – for example one which would require the use of sentence context to choose between related yet distributionally different antonyms (e.g. “little, small” as antonyms of “big”) but chose to stick to a previously used benchmark. This allows the direct comparison with previously reported methods.

[4]http://www.bloomsbury.com/

[5]http://www.umiacs.umd.edu/$\sim$saif/WebDocs/LC-data/{devset,testset}.txt

Dimensions	Bloomsbury Prec.	WordNet Prec.
50	0.778	0.475
100	0.850	0.563
200	0.856	0.569
300	**0.863**	**0.625**
400	0.843	**0.625**
500	0.843	0.613
750	0.830	0.613
1000	0.837	0.544
2000	0.784	0.519
3000	0.778	0.494

Table 5: The performance of PILSA vs. the number of dimensions when applied to the closest-opposite questions from the GRE development set. Out of the 162 questions, using the Bloomsbury thesaurus data we are able to answer 153 of them. Using 300 dimensions gives the best precision (132/153 = 0.863). This dimension setting is also optimal when using the WordNet data, which answers 100 questions correctly out of the 160 attempts (100/160 = 0.625).

Some of these questions contain very rarely used target or choice words, which are not included in the thesaurus vocabulary. In order to provide a fair comparison to existing methods, we do not try to randomly answer these questions. Instead, when the target word is out of vocabulary, we skip the whole question. When the target word is in vocabulary but one or more choices are unknown words, we ignore those unknown words and pick the word with the lowest cosine similarity from the rest as the answer. The results of our methods are reported in *precision* (the number of questions answered correctly divided by the number of questions attempted), *recall* (the number of questions answered correctly divided by the number of all questions) and F_1 (the harmonic mean of precision and recall)[6]. We now turn to an in-depth evaluation.

[6]Precision/recall/F_1 were used in (Mohammad et al., 2008) as when their system “did not find any evidence of antonymy between the target and any of its alternatives, then it refrained from attempting that question.” We adopt this convention to provide a fair comparison to their system.

	Dev. Set			Test Set		
	Prec	Rec	F_1	Prec	Rec	F_1
WordNet lookup	0.40	0.40	0.40	0.42	0.41	0.42
WordNet signed-TFIDF w/o LSA	0.41	0.41	0.41	0.43	0.42	0.43
WordNet PILSA	0.63	0.62	0.62	0.60	0.60	0.60
Bloomsbury lookup	0.65	0.61	0.63	0.61	0.56	0.59
Bloomsbury signed TFIDF w/o LSA	0.68	0.64	0.66	0.63	0.57	0.60
Bloomsbury PILSA	0.86	0.81	0.84	0.81	0.74	0.77
Bloomsbury PILSA + S2Net	**0.89**	0.84	0.86	**0.84**	0.77	0.80
Bloomsbury PILSA + S2Net + Embedding	0.88	**0.87**	**0.87**	0.81	**0.80**	**0.81**
(Mohammad et al., 2008)	0.76	0.66	0.70	0.76	0.64	0.70

Table 6: The overall results. PILSA performs LSA on the signed TF-IDF vectors.

7.3 Basic PILSA

When applying PILSA, we need to determine the number of dimensions in the projected space. Evaluated on the GRE development set, Table 5 shows the precision of PILSA, using two different training datasets, Bloomsbury and WordNet, at different dimensions.

The Bloomsbury-based system is able to answer 153 questions, and the best dimension setting is 300, which answers 132 questions correctly and thus archives 0.863 in precision. In contrast, the larger vocabulary in WordNet helps the system answer 160 questions but the quality is not as good. We find dimensions 300 and 400 are equally good, where both answer 100 questions correctly (0.625 in precision)[7]. Because a lower number of dimensions is preferred for saving storage space and computing time, we choose 300 as the number of dimensions in PILSA.

We now compare the proposed methods. All results are summarized in Table 6. When evaluated on the GRE test set, the Bloomsbury thesaurus-based methods (Lines 4–7) attempted 865 questions. The precision, recall and F_1 of the Bloomsbury-based PILSA model (Line 6) are 0.81, 0.74 and 0.77, which are all better than the best reported method in (Mohammad et al., 2008)[8]. In contrast, the WordNet-based methods (Lines 1–3) attempted 936 questions. However, consistent with what we observed on the development set, the WordNet-based model is inferior. Its precision, recall and F_1 on the test set are 0.60, 0.60 and 0.60 (Line 3). Although the quality of the data source plays an important role, we need to emphasize that performing LSA using our polarity inducing construction is in fact a critical step in enhancing the model performance. For example, directly using the antonym sets in the Bloomsbury thesaurus gives 0.59 in F_1 (Line 4), while using cosine similarity on the signed vectors prior to LSA only reaches 0.60 in F_1 (Line 5).

[7]Note that the number of questions attempted is not a function of the number of dimensions.

[8]We take a conservative approach and assume that skipped questions are answered incorrectly. The difference is statistically significant at 99% confidence level using a binomial test.

7.4 Improving Precision with Discriminative Training

Building on the success of the unsupervised PILSA model, we refine the projection matrix. As described in Section 5, we take the PILSA projection matrix as the initial model in S2Net and train the model using 20,517 pairs of antonyms sampled from the Bloomsbury thesaurus. A separate sample of 5,000 antonym pairs is used as the validation set for hyper-parameter tuning in regularization. Encouragingly, we found that the already strong results of PILSA can indeed be improved, which gives 3 more points in both precision (0.84), recall (0.77) and F_1 (0.80).

7.5 Improving Recall with Unsupervised Data

We next evaluate our approach of extending the word coverage with the help of an external text corpus, as well as the lexical analysis procedure. Using the Bloomsbury PILSA-S2Net thesaurus space and the Wikipedia corpus space, our method increases recall by 3 points on the test set. Compared to the in-vocabulary only setting, it attempted 75 more questions ($865 \rightarrow 940$) and had 33 of them correctly answered.

While the accuracy on these questions is much higher than random, the fact that it is substantially below the precision of the original indicates some room for improvement. We notice that the out-of-thesaurus words are either offensive words excluded in the thesaurus (e.g., *moronic*) or some very rarely used words (e.g., *froward*). When the lexical analysis procedure fails to match the target word to some in-thesaurus words, the context vector embedding approach solves the former case, but has difficulty in handling the latter. The main reason is that such words occur very infrequently in a general corpus, which result in significant uncertainty in their semantic vectors. Other than using a much larger corpus, approaches that leverage character n-grams may help. We leave this as future work.

8 Conclusion

In this paper we have tackled the problem of finding a vector-space representation of words where, by construction, synonyms and antonyms are easy to distinguish. Specifically, we have defined a way of assigning sign to the entries in the co-occurrence matrix on which LSA operates, such that synonyms will tend to have positive cosine similarity, and antonyms will tend to have negative similarities. To the best of our knowledge, our method of inducing polarity to the document-term matrix *before* applying LSA is novel and has shown to effectively preserve and generalize the synonymous/antonymous information in the projected space. With this vector space representation, we were able to bring to bear the machinery of discriminative training in order to further optimize the word representations. Finally, by using the notion of closeness in this space, we were able to embed new out-of-vocabulary words into the space. On a standard test set, the proposed methods improved the F measure by 11 points absolute over previous results.

Acknowledgments

We thank Susan Dumais for her thoughtful comments, Silviu-Petru Cucerzan for preparing the Wikipedia data, and Saif Mohammad for sharing the GRE datasets. We are also grateful to the anonymous reviewers for their useful suggestions.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of HLT-NAACL*, pages 19–27.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley.

J. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8).

Thorsten Brants and Alex Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium.

N. Coccaro and D. Jurafsky. 1998. Towards better integration of semantic predictors in statistical language modeling. In *Proceedings, International Conference on Spoken Language Processing (ICSLP-98)*.

D. A. Cruse. 1986. *Lexical Semantics*. Cambridge University Press.

James R. Curran and Marc Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of the ACL-02 workshop on Unsupervised lexical acquisition - Volume 9*, pages 59–66. Association for Computational Linguistics.

Thomas de Simone and Dimitar Kazakov. 2005. Using wordnet similarity and antonymy relations to aid document retrieval. In *Recent Advances in Natural Language Processing (RANLP)*.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(96).

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Zelig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Mario Jarmasz and Stan Szpakowicz. 2003. Rogets thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*.

Thomas Landauer and Susan Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review, 104(2)*, pages 211–240.

T.K. Landauer and D. Laham. 1998. Learning human-like knowledge by singular value decomposition: A progress report. In *Neural Information Processing Systems (NIPS)*.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes, 25*, pages 259–284.

T.K. Landauer. 2002. On the computational basis of learning and cognition: Arguments from lsa. *Psychology of Learning and Motivation*, 41:43–84.

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 768–774, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Milne and Ian H. Witten. 2008. An effective low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*.

G. Minnen, J. Carroll, and D. Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word pair antonymy. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2011. Measuring degrees of semantic opposition. Technical report, National Research Council Canada.

Gregory L. Murphy and Jane M. Andrew. 1993. The conceptual basis of antonymy and synonymy in adjectives. *Journal of Memory and Language*, 32(3):1–19.

Jorge Nocedal and Stephen Wright. 2006. *Numerical Optimization*. Springer, 2nd edition.

John Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual document representations from discriminative projections. In *Proceedings of EMNLP*, pages 251–261.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of HLT-NAACL*, pages 109–117.

Gerard Salton and Michael J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

G. Salton, A. Wong, and C. S. Yang. 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11).

D. Schwab, M. Lafourcade, and V. Prince. 2002. Antonymy and conceptual vectors. In *International Conference on Computational Linguistics (COLING)*.

Peter Turney and Michael Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning, 60 (1-3)*, pages 251–278.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, (37).

Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Recent Advances in Natural Language Processing (RANLP)*.

Peter D. Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *European Conference on Machine Learning (ECML)*.

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

Peter Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *International Conference on Computational Linguistics (COLING)*.

Lonneke van der Plas and Gosse Bouma. 2005. Syntactic contexts for finding semantically similar words. In *Proceedings of the Meeting of Computational Linguistics in the Netherlands 2004 (CLIN)*.

Lonneke van der Plas and Jörg Tiedemann. 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 866–873. Association for Computational Linguistics.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, New York, NY, USA. ACM.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteen Conference on Computational Natural Language Learning (CoNLL)*, pages 247–256, Portland, Oregon, USA.