IEICE Transactions on Information and Systems,
vol.E93-D, no.6, pp.1378–1385, 2010.

Spectral Methods for Thesaurus Construction

Nobuyuki Shimizu (shimizu@r.dl.itc.u-tokyo.ac.jp)
Information Technology Center, The University of Tokyo

Masashi Sugiyama (sugi@cs.titech.ac.jp)
Tokyo Institute of Technology

Hiroshi Nakagawa
Information Technology Center, The University of Tokyo

Abstract

Traditionally, popular synonym acquisition methods are based on the distributional hypothesis, and a metric such as Jaccard coefficients is used to evaluate the similarity between the contexts of words to obtain synonyms for a query. On the other hand, when one tries to compile and clean a thesaurus, one often already has a modest number of synonym relations at hand. Could something be done with a half built thesaurus alone? We propose the use of spectral methods and discuss their relation to other network-based algorithms in natural language processing (NLP), such as PageRank and Bootstrapping. Since compiling a thesaurus is very laborious, we believe that adding the proposed method to the toolkit of thesaurus constructors would significantly ease the pain in accomplishing the task.

Keywords

synonym acquisition, synonym extraction, thesaurus, spectral clustering, graph Laplacian

1 Introduction

Since the usage of thesauri is known to improve the performance of various tasks in natural language processing (NLP) [1] and information retrieval (IR) [2, 3], they are regarded as one of the most important resources in these fields. However, thesauri are one of the most laborious resources to create and maintain.

Imagine you are constructing a thesaurus and and you have just created a thesaurus of a modest size. As we build thesauri from a scratch, all thesauri are rather small and poorly designed at some point of their development. You wonder what other synonyms are

missing from the thesaurus, and you would like a system that suggests you a next synonym candidate you should consider adding to the thesaurus. While the traditional methods for synonym acquisition collects statistics from a large corpus and compare contexts of words for similarity, they do not use the thesaurus that you just compiled. What if you follow synonym relations? Given a thesaurus entry, you may find two words that are synonymous with the entry. Then maybe these words are likely to be synonymous to each other as well. In the situation such as above, it is not clear what path you should follow to find a synonym candidate. If you have enough entries in the thesaurus, it may be better to observe the network of synonyms, where a node represents a word and an edge is a synonym relation, and guess missing links in the network. While there are studies on graph clustering involving synonyms, to the best of the authors' knowledge, there has been no research on thesaurus expansion based on link analysis of synonym networks. On the other hand, if the network is not dense enough to suggest a synonym candidate, it may be better to use more traditional synonym acquisition methods based on the distributional hypothesis.

The objective of this paper is to explore the trade-off between network-based distance measures to more traditional corpus-based synonym acquisition measures, and shed a light on the conditions in which one is superior to the other. Our contribution is as follows. First, we demonstrate that Graph Laplacian Embedding (GLE), a network based method, performs quite well in assigning true synonym candidates a coordinate that is close to that of the synonymous query, solely using the existing synonym network that we intend to expand. This is surprising, as two synonyms of a word are not generally synonymous to each other. To compare GLE with corpus based methods, we then vary the density of the existing synonymous connection from the query words to the rest of words in the network, and see how the performance of a network based method deteriorates as the graph gets sparser and less informative. Even then, we find that GLE works better than corpus based method until two thirds of connections are removed. This shows the effectiveness of utilizing existing synonymous network as we expand a thesaurus. At the same time, we discuss a number of network-based algorithms for NLP, and how the proposed spectral methods provide a unified view. Specifically, we advocate the use of non-principal eigenvectors of a transition matrix and give the interpretations of these vectors.

The remainder of this paper is organized as follows: in Section 2, we review related work on synonym acquisition and network-based methods in NLP. In Section 3, we discuss the baseline and the spectral methods. We then review traditional corpus-based synonym acquisition methods in Section 4. The experimental settings are described in Section 5. Finally in Section 6, we discuss the results and conclude in Section 7.

2 Related Work

The topic of lexical similarity enjoys a long history of research; some of them are based on dictionaries such as WordNet [4, 5], some of them on similarity/distance metrics and

contextual features of a word [6, 7]. Although both approaches yield lexical similarity, synonym acquisition is typically done without a dictionary, since the objective of synonym acquisition is to construct or expand a thesaurus in a domain where such language resources do not exist.

To acquire synonyms without dictionaries, methods usually assume the distributional hypothesis [8] which states that semantically similar words share similar contexts. Based on this hypothesis, a synonym acquisition method roughly implements the following procedure. First, given a target word, we extract useful features from the contexts of the target. Features often include surrounding words or dependency structure. Second, to evaluate the similarity of words, we choose a similarity/distance metric, and calculate the similarity/distance between the contexts of two given words. Many studies [9, 10, 11, 12] investigated a variety of distance and similarity metrics on synonym acquisition performance. Among the ones considered in the literature, the examples of metrics known for higher performance in synonym acquisition include cosine similarity, Jaccard coefficient and vector-based Jaccard coefficient. Another notable metric is skew divergence, a metric based on a Kullback-Leibler divergence. While not a synonym acquisition, other notable related research include [13], who extended WordNet hyponym-hypernym relations.

As our proposed method is based on the eigenvectors of the graph Laplacian or transition matrix, we list a few applications of eigenvectors in NLP as well. Some applications of eigenvectors are explicit in the form of PageRank. PageRank [14] is based on the power method, an iterative algorithm for computing the principal eigenvector of a transition matrix. It finds a number of applications, such as word sense disambiguation [15] and automatic extractive summarization [16].

While there are many applications of the principal eigenvector, applications of the non-principal eigenvectors of a transition matrix are few and far in between, with the exception of [17]. As they are clearly useful, we believe they warrant more applications. In our experiments, we show that the number of eigenvectors affects the performance of the system.

3 Network-based Methods

We introduce two network-based methods in this section. In the first subsection, we introduce Squared Affinity Matrix (SAM), which forms the baseline. Next we explain Graph Laplacian Embedding (GLE) in the second subsection and provide a unified view of graph structure defined by the graph Laplacian and transition matrix. As we advocate using non-principal eigenvectors of a transition matrix, we state what they mean in detail.

The following is the notation common to all network-based models. Let n be the number of words we are considering, and each word x_i is represented by a vector $\mathbf{x}_i$ where i ranges from 1 to n. This includes the words in the thesaurus constructed so far and other words under consideration for inclusion to the thesaurus. The set of words in an arbitrary feature space are represented as a weighted undirected graph $G = (V, E)$ where the nodes $x_i, x_j \in V$ of the graph are words, and an edge $e_{i,j} \in E$ is formed between

every synonymous pair of words. Also, let W be an n by n sample-sample affinity matrix. This symmetric matrix represents the synonym relations in the thesaurus. If the words x_i, x_j are known to be synonymous so far, then $W_{i,j} = 1$. Otherwise $W_{i,j} = 0$. Note that both network-based methods are unsupervised. They simply define a distance over nodes in the network; they are not at all extensible to words outside the thesaurus, and no supervised learning takes place.

Squared Affinity Matrix and Cubed Affinity Matrix

A simple method for predicting is to compute $Y = WW$, and find words x_i, x_j such that $Y_{i,j}$ is non-zero. This is like saying my friend's friend is my friend; two words that share a synonym is predicted to be synonymous with each other. As the elements of W are either zero or one, we initially treated the addition operator defined on the elements to be a binary OR operation. Unfortunately, this operation does not take accounts of the number of friends. In order to take accounts of the degree of friendship, we instead use the ordinary addition defined over real number and normalize the resulting figure to be between 0 and 1.

The computational complexity and required storage size of this approach in its naive implementation is $O(n^3)$ and $O(n^2)$, respectively. However, since W is sparse, its computational cost and memory space is much less in practice. We also consider Cubed Affinity Matrix (CAM) by computing $Y = WWW$. Once we obtain Y, given a query i, we sort the value $Y_{i,j}$ for each candidate j and evaluate the ranked list for synonym retrieval performance.

Graph Laplacian Embedding (GLE)

The next approach we take is based on Graph Laplacian Embedding (GLE) [18, 19].

In this approach, we aim to reduce the dimension d of $\mathbf{x}_i$ to r ($1 \leq r \leq d$), such that the respective projections of $\mathbf{x}_i$ and $\mathbf{x}_j$, denoted by $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, are close to each other if $W_{i,j}$ is 1. The objective function is stated as follows:

$$\min_{\phi(\mathbf{x}_1),\ldots,\phi(\mathbf{x}_n)} \frac{1}{2} \sum_{i,j=1}^{n} W_{i,j} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2,$$

which can be equivalently expressed after a few lines of calculation as

$$\min_{\Phi=(\phi(\mathbf{x}_1),\ldots,\phi(\mathbf{x}_n))} \mathrm{trace}(\Phi L \Phi^\top),$$

where $\Phi = (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))$ is an r by n matrix with i-th column representing the projected vector $\phi(\mathbf{x}_i)$. L is an n by n graph Laplacian matrix, $L = D - W$, and D is an n by n diagonal matrix with i-th diagonal element $\sum_{j=1}^{n} W_{i,j}$.

Under constraints that $\Phi D \Phi^\top = I$ (to avoid rank degeneracy; I denotes the identity matrix) and $\Phi D \mathbf{1} = 0$ (to remove a trivial solution; $\mathbf{1}$ denotes the vector with all ones),

a solution of the above minimization problem can be analytically obtained as $\phi(\mathbf{x}_i) = (\psi_1(\mathbf{x}_i), \ldots, \psi_r(\mathbf{x}_i))^\top$, where $\psi_k(\mathbf{x}_i)$ is the i-th element of the k-th eigenvector $\mathbf{\Psi}_k$ and $\mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_r$ ($\in \mathbb{R}^n$) are r eigenvectors associated with the r smallest *positive* eigenvalues (λ) of the following n-dimensional *sparse* generalized eigenproblem:

$$L\mathbf{\Psi} = \lambda D\mathbf{\Psi}. \tag{1}$$

Finally, given a query i for each candidate j, we compute the Euclidean distance between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ and evaluate the ranked list of the candidates for synonym retrieval performance. Given a fixed r, the computational complexity and memory requirement of the algorithm in its naive implementation is both $O(n^2)$. However, since L is sparse, its computational cost and required memory space is much less in practice (e.g., ARPACK). Thus GLE is scalable to large datasets both in terms of computation time and storage space and therefore would be suitable for the current task of synonym prediction.

In the next two subsections, we show the equivalence of GLE to other methods (Normalized Cuts, Random Walk), to provide the unifying view of the graph based NLP methods.

GLE and Normalized Cuts

To provide insights into what $\phi(\mathbf{x}_i)$ represents, we explain the relationship between the eigenvector $\mathbf{\Psi}_1$ (which is often called the Fiedler vector in spectral graph theory) and clustering based on normalized cuts. Suppose you would like to bi-partition a connected graph into two sub-graphs A and B by removing the edges between them. The total weight of edges to be removed is called *cut*: $cut(A, B) = \sum_{i \in A, j \in B} W_{i,j}$.

Although clustering based on the minimum cut of a graph often produces reasonable partitions, this clustering criterion tends to cut a small set of edges to isolated nodes in the graph. To balance the size of partitions, Shi and Malik [20] have proposed the following *normalized cut* criterion.

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)},$$

where $assoc(A, V) = \sum_{i \in A, j \in V} W_{i,j}$ is the total weight of nodes in A to all nodes in the graph. After a little algebra, one arrives at

$$\min_{\mathbf{\Psi}} \frac{\mathbf{\Psi}^\top L \mathbf{\Psi}}{\mathbf{\Psi}^\top D \mathbf{\Psi}}$$

subject to $\mathbf{\Psi} \in \{1, -b\}^n$, $b = \frac{\sum_{i \in A} D_{i,i}}{\sum_{j \in B} D_{j,j}}$ and $\mathbf{\Psi}^\top D\mathbf{1} = 0$. Once we relax $\mathbf{\Psi}$ to take on real values, the objective is reduced to (1). Since a graph Laplacian L is known to have zero as its minimum eigenvalue, the second minor eigenvector $\mathbf{\Psi}_1$ corresponds to the relaxed solution for the partition assignments by normalized cuts. Thus graph Laplacian embedding $\phi(\mathbf{x}_i)$ has this soft partition assignments in its first coordinate. Note that other eigenvectors besides the second minor eigenvector could also be useful—after all, the third minor eigenvector and so on also reduce the objective, and they could represent the second best bi-partitioning, and so forth.

GLE and Random Walk

We may also view the generalized eigendecomposition problem (1) in terms of random walk. By normalizing the matrix W, we obtain the stochastic matrix $P = D^{-1}W$ whose every row sums to 1. This matrix can be interpreted as defining a random walk on the graph, and thus the transition probability from the vertex x_i to x_j in one time step:

$$Pr\{v(t+1) = x_j | v(t) = x_i\} = P_{i,j},$$

where $v(t)$ indicates the location of vertex at time t. Next we relate this random walk view to the graph Laplacian. The solutions to the spectral problem

$$P\mathbf{\Psi} = \lambda\mathbf{\Psi} \tag{2}$$

corresponds to (1) via the following proposition [21] : *If* $(\lambda, \mathbf{\Psi})$ *is a solution of* (2)*, then* $((1-\lambda), \mathbf{\Psi})$ *is a solution of* (1). This result allows us to analyze the graph Laplacian embedding using the eigenvectors of P [22]. Let $p(t, x_j|x_i)$ be the probability of random walk landing at a vertex x_j at time t given a starting vertex x_i at time $t = 0$. If the graph is connected, P is an irreducible and aperiodic Markov chain. It has the largest eigenvalue equal to 1 and the remaining eigenvalues are strictly smaller than 1 in absolute value. Then, regardless of the starting position x_i, there is a stationary distribution $\phi_0(x_j)$ such that

$$\lim_{t\to\infty} p(t, x_j|x_i) = \frac{D_{j,j}}{\sum_k D_{k,k}} = \phi_0(x_j).$$

$\phi_0(x_j)$ corresponds to the eigenvector associated with the largest eigenvalue (equal to 1) of P. We now consider the following distance between nodes x_i and x_j at time t.

$$dist_t^2(x_i, x_j) = \sum_{k=1}^{n} (p(t, x_k|x_j) - p(t, x_k|x_i))^2 w(x_k)$$

with specific choice of $w(x_k) = 1/\phi_0(x_k)$. This choice allows us to put more weights on low density points. Nadler et al. [22] call this *diffusion distance*. They also defined the *diffusion map* at time t, $\phi_t(x_i) = ((1-\lambda_1)^t\psi_1(x_i), \ldots, (1-\lambda_r)^t\psi_r(x_i))$ and showed that the *diffusion distance* is equal to the *Euclidean distance* in the diffusion map space with all $(n-1)$ eigenvectors.

$$\begin{aligned} dist_t^2(x_i, x_j) &= \sum_{k=1}^{n-1} (1-\lambda_k)^{2t} (\psi_k(x_i) - \psi_k(x_j))^2 \\ &= \|\phi_t(x_i) - \phi_t(x_j)\|^2. \end{aligned}$$

GLE is equivalent to the diffusion map at time $t = 0$. This is what we use in our experiments.

The analysis also shows the relation of our method (GLE) to PageRank. While PageRank [14] computes the principal eigenvector of a transition matrix, we mainly make use of non-principal eigenvectors, which corresponds to the diffusion distance.

4 Corpus-based Methods

In this section, we describe the corpus-based synonym acquisition methods based on the distributional hypothesis, explaining the preprocessing of data and features, as well as various metrics we used to find the lexical similarity between target words.

4.1 Features

Since effectiveness of grammatical dependencies for synonym acquisition is well-known [23, 7], we use a syntactic parser called RASP Toolkit 2 (RASP2) [24] to extract contextual features for synonym acquisition. RASP2 analyzes the sentence and outputs the extracted dependency structure as n-ary *grammatical relations* [25]. After we identify contextual features of a word using the parser, for each pair of word x and contextual feature φ, we compute the raw co-occurrence count $N(x, \varphi)$ from one whole year of New York Times articles (1997)[1]. New York Times articles (1997) consist of approximately 202 thousand documents and 131 million words.

While one whole news paper corpus may yield more than 100,000 of contextual feature types, many of them just occurs once or twice. Since a large number of dimensions render similarity calculation impractical, we apply simple frequency cutoff to reduce the number of contextual features and synonym candidates we consider. Specifically, we remove any word x such that $\sum_{\varphi} N(x, \varphi) < 20$ and any feature φ such that $\sum_{x} N(x, \varphi) < 20$ from the co-occurrence data. Once we apply this cut-off, we are left with 27,688 word types and 83,029 features. To assign weights to the features, we use pointwise mutual information: $\mathrm{PMI}(x, \varphi) = \log(P(x, \varphi)/(P(x)P(\varphi)))$. Then the weight of a feature is: $\mathrm{wgt}(x, \varphi) = \max(\mathrm{PMI}(x, \varphi), 0)$. As negative values are reported to lower the performance [10], we bound PMI by 0.

4.2 Similarity and Distance Metrics

As a baseline, we compare three similarity metrics (cosine similarity, Jaccard coefficient, vector-based Jaccard coefficient (Jaccardv)) and two distance metrics (Jensen-Shannon divergence (JS), skew divergence (SD99)).

Before we describe the metrics, we define some notations: Let $\mathbf{x}_i$ be the feature vector corresponding to word x_i, i.e., $\mathbf{x}_i = [\mathrm{wgt}(x_i, \varphi_1) \ldots \mathrm{wgt}(x_i, \varphi_d)]^\top$, where φ_j is a contextual feature. Let $F(x)$ be the set of contextual features that co-occur with word x, that is, $F(x) = \{\varphi | N(x, \varphi) > 0\}$.

Cosine Similarity:

$$\frac{\mathbf{x}_1^\top \mathbf{x}_2}{||\mathbf{x}_1|| \cdot ||\mathbf{x}_2||}.$$

[1]New York Times we use is a portion of the English Gigaword corpus obtainable from Linguistic Data Consortium. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T05

Jaccard Coefficient:

$$\frac{\sum_{\varphi \in F(x_1) \cap F(x_2)} \min(\mathrm{wgt}(x_1, \varphi), \mathrm{wgt}(x_2, \varphi))}{\sum_{\varphi \in F(x_1) \cup F(x_2)} \max(\mathrm{wgt}(x_1, \varphi), \mathrm{wgt}(x_2, \varphi))}.$$

Vector-based Jaccard Coefficient (Jaccardv):

$$\frac{\mathbf{x}_1^\top \mathbf{x}_2}{||\mathbf{x}_1|| + ||\mathbf{x}_2|| - \mathbf{x}_1^\top \mathbf{x}_2}.$$

Jensen-Shannon Divergence (JS):

$$\frac{1}{2}\{KL(p_1||m) + KL(p_2||m)\}.$$

To define Jensen-Shannon divergence, we need to map a word x_i to a probability distribution: $p_i(\varphi) = N(x_i, \varphi)/\sum_{\varphi'} N(x_i, \varphi')$ and $m = (p_1 + p_2)/2$. Jensen-Shannon divergence (JS) is a symmetric version of the Kullback-Leibler (KL) divergence which measures the distance between two probability distributions. Although the KL divergence suffers from the so-called zero-frequency problem, this version naturally avoids it.

Skew Divergence (SD99):

$$KL(p_1||\alpha p_2 + (1-\alpha)p_1).$$

The skew divergence is an adaptation of KL divergence, which avoids the zero-frequency problem by mixing the original distribution with the target distribution. The parameter α is set to 0.99 in our experiments [26].

Again, given a query i for each candidate j, we compute the similarity/distance metric between i and j. We then use the ranked list of the candidates for the evaluation.

5 Experimental Settings

To provide a good overall perspective before explaining experimental settings, we summarize the two approaches in the previous section.

Corpus-based methods exploit a corpus to fill the affinity matrix $W(= Y)$, and it uses only the affinity matrix by itself without a further modification. The advantage of these methods is that the matrix is not as sparse as the affinity matrix for the network-based methods, which contain handcrafted synonymous relations.

On the other hand, network-based methods use the handcrafted affinity matrix containing known synonymous relations. As the existence of edges in such a network gives away the answer to the queries, we assume that there are no such edges given before the experiments, carefully constructing the training set and test set partitions. This prevents us from simply using the affinity matrix and forces us to look at least one step further in

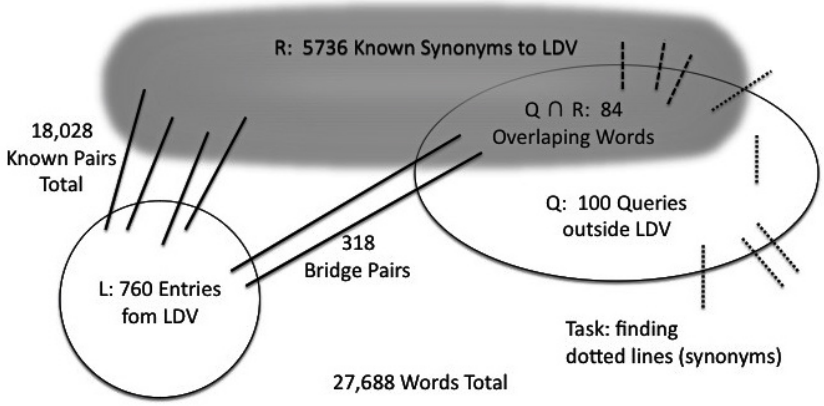

Figure 1: Overlap between Test Set and Thesaurus

the affinity matrix, be it SAM, CAM, or GLE. However, being hand-made, the entries of the affinity matrix are of much higher quality despite being sparse. This higher quality allows us to exploit the synonymous relations to improve the performance.

We now go on to clarify how we create the affinity matrix for the respective approaches.

5.1 Thesaurus and the Test Set

As a starting point of thesaurus construction, we combine a portion of three thesauri into one synonym network. We also choose the Longman Defining Vocabulary (LDV) as a set of query words whose synonyms are known.

For each word in LDV, we consult three existing thesauri: Roget's Thesaurus, Collins COBUILD Thesaurus, and WordNet. We look up each LDV as a noun to obtain the union of synonyms. After we remove words marked "idiom", "informal" or "slang" and phrases comprised of two or more words, the union is used as the reference set of query words. We omit the LDV word for which we find no noun synonyms in any of the reference thesauri. From the remaining 771 LDV words, we select 760 query words that had at least one synonym in the corpus.

We consider the training set for the network-based methods to be a bipartite graph, where in one partition L, there are 760 nodes (LDV entries in the thesaurus) and in the other partition R, 5736 nodes (words known to be synonymous to the entries). Between these partitions, there are 18,028 edges that represent synonymous relations. Thus, the affinity matrix for network-based methods is very sparse, with only 18,028 entries filled in it. To construct the test set outside LDV, we pseudo-randomly select 100 words with 5 or more synonyms and treat them as test queries, Q. Of these, we find 84 queries be also in R. As these queries are in R, they are adjacent to some words in L. We find that we have 318 edges between 84 words in $Q \cap R$ and 760 words in L. Let us call them bridge pairs. The objective is to find 1010 words synonymous with the queries in Q, with the condition that these 1010 words being outside L. Let this set of 1010 words be S. Since

there is no overlap between S and L, the synonyms to the queries Q are not given away by the training set.

As there are some connections between 84 test queries and 5736 candidates in the synonym network, these words with a connection to the synonym network have a chance of synonym identification using GLE. Other words not connected to the network can only be retrieved using the corpus-based methods. There are 26,928 words in the corpus outside L, all of which is a synonym candidate to a word in Q. Note that, with this setting, words in Q are possibly synonymous with another word in Q. The relations between the training and test set is shown in Figure 1.

As the training set for the corpus-based methods includes no handcrafted synonym relations, the corpus-based methods are free to use all words (27,688 word types) and associate them with features (83,029 of them) extracted from news texts. The affinity matrix for them is induced from this information instead of pre-built thesaurus.

The performance of corpus-based methods depends solely on the quality and quantity of news corpus; the performance of the network-based methods depends on density of the handcrafted synonymous relations. To show the trade-off between the corpus based methods and GLE, we randomly partition these 318 bridge pairs into 10 sets and see how the network-based methods perform as we remove the overlap between the thesaurus and the test set. This reduces the density of the existing synonymous connection from the query words to the words in the corpus, and the performance of GLE deteriorates as the graph gets sparser and less informative.

When computing GLE, we solved a simplified problem $L\mathbf{\Psi} = \lambda\mathbf{\Psi}$, not (1) since the results were almost the same. In addition, we used another trick to improve the performance, which is to normalize the embedding by the following operation: $\phi(\mathbf{x}_i) := \phi(\mathbf{x}_i)/||\phi(\mathbf{x}_i)||$

5.2 Evaluation Measure

To evaluate we make use of Mean Average Precision, Average Rank of Last Synonym and TOP1 [27]. Every evaluation measure considered in this section analyzes a ranked list of synonym candidates, sorted by predicted similarity in ascending order.

The Average Precision (APR) measure evaluates the precision at every recall where it is defined, and finds the threshold that produces the maximum precision for each of these recall values. Average precision is the average over all of the recall values greater than 0. In our experiments, we measure average precision on each query, and report the mean of each query's average precision as the final metric. A perfect prediction translates to APR of 1.0. The lowest possible APR is 0.0.

Average Rank of Last Synonym (RKL) measures how far down the ranked list we must go to find the last true synonym. Note that the lower the figure, the better the system is for this evaluation measure. If a query word has N synonyms in the ranked list, then the highest obtainable RKL value for the query word is N. RKL near 26,927 (the number of synonym candidates, in the current setting) indicates poor performance of a synonym acquisition system.

TOP1 measures how likely we have correct synonyms at the top of the ranked list. To

Table 1: Comparison of Corpus-based Methods

Metric	APR	RKL	TOP1
Euclidean	0.00044	25556.89	0.0000
Manhattan	0.00041	25697.15	0.0000
Cosine	0.06954	12100.68	0.1900
Jaccard	**0.07491**	12630.72	**0.2700**
Jaccardv	0.07293	12313.08	0.2200
JS	0.00072	25685.60	0.0000
SD99	0.04848	**11920.39**	0.1800

calculate TOP1 of a metric, we first score each queries as follows. Given a query and a metric, if the word closest to the query word is a synonym, then the score is 1. If there are ties, all of the tied cases must be synonyms of the query. Otherwise, the score is 0. TOP1 is an average of the scores above over all queries. Its value ranges from 1.0 to 0.0. To achieve 1.0, perfect TOP1 prediction, a synonym acquisition system must place a true synonym at the top of the ranked list in every query.

6 Results

Table 1 shows how various corpus-based methods perform in terms of APR, RKL and TOP1. As the figures in bold format indicate, Jaccard performed best in two evaluation measures APR and TOP1, and SD99 performed best in RKL. We take these metrics and compare them to the performance of network-based methods as the number of bridge pairs is increased from 31 to 318, by the increments of 10%. The performance of network-based methods are shown in Table 2. As we examine a larger value for the parameter of Graph Laplacian Embedding, which is the dimension of the projected space r, the APR kept increasing, with the largest examined r being 600. TOP1, on the other hand, peaked between 300 and 600.

The figures for Graph Laplacian Embedding are in bold if they are above the baseline performance provided by Jaccard and SD99. We notice that as the number of the bridge pairs increase, around 60%, with 190 nodes in the intersection, Graph Laplacian Embedding starts to give a clear edge over Jaccard in terms of TOP1. RKL of network-based methods are lower than those of corpus-based ones due to the sparseness of the affinity matrix compared to the feature vectors. As the candidate words include all words in the corpus, many completely unreachable from the thesaurus, this is expected. For SAM, Table 2 shows two variants: one with (OR) uses OR operation for addition, and the other (+) uses the ordinary addition operation and rescales the resulting number to fit between 0 and 1 afterwards. While the simple heuristics of the squared affinity matrix (+) generally outperforms in terms of APR, the Graph Laplacian Embedding method is clearly better than SAM in terms of RKL. On the other hand, CAM (+) which compute $Y = WWW$ does not perform well at all.

Table 2: Comparison of Network-based and Corpus-based Methods

Network-based	SAM (OR)			SAM (+)			CAM (+)			GLE (r=600)		
# bridge pairs (# connected queries)	APR	RKL	TOP1	APR	RKL	TOP1	APR	RKL	TOP1	APR	RKL	TOP1
31 (23)	0.026	23582	0.05	0.048	26657	0.05	0.027	26658	0.00	0.040	21217	0.04
62 (39)	0.048	23149	0.09	**0.080**	26388	0.10	0.041	26389	0.00	0.073	21190	0.09
94 (50)	0.058	21991	0.09	**0.120**	26120	0.15	0.046	26123	0.00	**0.102**	20460	0.13
126 (63)	**0.082**	21335	0.11	**0.158**	25852	0.19	0.061	25590	0.00	**0.141**	19929	0.21
158 (68)	**0.086**	20901	0.09	**0.194**	25852	0.22	0.060	25326	0.00	**0.156**	19952	0.22
190 (72)	**0.100**	20507	0.12	**0.233**	25586	0.24	0.066	25066	0.00	**0.187**	19264	0.24
222 (76)	**0.109**	19895	0.11	**0.265**	25317	**0.29**	0.070	25066	0.00	**0.218**	18805	**0.32**
254 (79)	**0.108**	19182	0.11	**0.289**	24779	**0.35**	0.068	24266	0.00	**0.247**	18173	**0.39**
286 (81)	**0.108**	18857	0.07	**0.307**	24242	**0.36**	0.065	23738	0.00	**0.262**	18043	**0.41**
318 (84)	**0.114**	18213	0.07	**0.326**	23703	**0.42**	0.067	23208	0.00	**0.276**	17467	**0.42**
Corpus-based	Jaccard			SD99								
	APR	RKL	TOP1	APR	RKL	TOP1						
	0.074	12630	**0.27**	0.048	**11920**	0.18						

The bold font indicates the portion of the table where the network-based methods outperform the corpus-based methods.

Table 3: Comparison of Network-based Methods on Connected Network

Network-based # connected queries (# connected words)	SAM (+) APR	SAM (+) RKL	SAM (+) TOP1	GLE (r=600) APR	GLE (r=600) RKL	GLE (r=600) TOP1
23 (4959)	0.228	4742	0.217	0.194	**2674**	0.173
39 (4975)	0.241	4468	0.256	0.224	**2839**	0.230
50 (4986)	0.275	4493	0.300	0.237	**2865**	0.260
63 (4999)	0.284	4526	0.301	0.253	**2755**	0.333
68 (5004)	**0.324**	4566	0.323	0.261	**2921**	0.323
72 (5008)	**0.364**	4391	0.333	0.292	**2528**	0.333
76 (5012)	**0.397**	4231	0.381	0.327	**2294**	0.421
79 (5015)	**0.413**	4012	0.443	0.355	**2310**	0.493
81 (5017)	**0.429**	3856	0.444	0.367	**2151**	0.506
84 (5020)	**0.440**	3721	0.500	0.375	**2105**	0.500

The bold font indicates that the method shows statistically significant performance compared to the other method.

In order to remove the random effects caused by the unreachable words in the network, we examine the performance of network-based methods using the connected portion of the network. After all, if the network is not connected, we know that the network-based methods are unworkable and we may opt to use corpus-based methods. Table 3 illustrates the difference between SAM (+) and GLE in this case. To evaluate the statistical significance, we employed paired sample t-test. The bold font indicates that SAM shows a statistically significant performance advantage in terms of APR. On the other hand, GLE has significant advantage in terms of RKL throughout. While GLE appears slightly better at TOP1, none of the figures were statistically significant. Notice that when the graph is restricted to reachable nodes, the figure for RKL using GLE is simply a half of the all connected nodes. This allows thesaurus constructor to examine only a half entries of existing thesaurus find missing synonymous pairs.

7 Discussion

We observed that when network-based methods find synonyms for a query, they tend to find a few of them at the same time. Perhaps due to the large number of contextual features, Jaccard finds synonyms more evenly across queries. In addition, some words seem to be distinctly easier for Graph Laplacian Embedding than Jaccard. For example, for the query word "pedigree", Graph Laplacian Embedding with 31 bridge pairs finds 6 synonyms at the top. The 6 synonyms are "parentage", "bloodline", "genealogy", "extraction", "ancestry" and "lineage". Jaccard finds none of them. This shows that when synonym relations surrounding the target query word are obvious to thesaurus constructors, but words themselves are rare, network-based methods work much better

than Jaccard and other corpus-based synonym acquisition methods.

Another observation we made is the dismal performance of SAM when the addition operation is a binary OR, and significantly higher performance of SAM when the addition is over real numbers. These two observations indicate that the synonym relations tend to form a cluster within the network, perhaps forming a dense or nearly complete sub-graph, displaying a modular property of a network. Furthermore, Cubed Affinity Matrix (CAM), instead of Squared Affinity Matrix, also shows a dismal performance in this dataset. These observations indicate that the synonym relations tends to form a cluster within the network, perhaps forming a dense or nearly complete sub-graph, displaying a modular property of a network.

This explains the performance of GLE, which examines wider range of connections than SAM. As GLE reaches beyond one edge, it is capable of identifying the words located far to be synonyms, outperforming in terms of RKL. On the other hand, modularity of the network strongly suggests that a word nearby is synonymous, and reaching farther increases the inclusion of noises, thereby decreasing the overall performance, especially apparent in terms of reduced APR. While SAM ($Y = WW$) performs well, as we move to CAM ($Y = WWW$), it seems to quickly forget the nearby synonyms and becomes much like a popularity vote instead of similarity one. Considering the difficulty of extending SAM to CAM, GLE appears to be quite successful at using wider portion of the network, reducing RKL. We postulate that this is due to the GLE's ability to model diffusion, keeping the essence of a similarity metric.

The thesaurus of the size we used in the experiment is modest, with less than 20,000 synonym pairs altogether. Since the study indicates perhaps only a half as much data is required for network-based methods to be effective, they warrants an application for those who are constructing a thesaurus. As recall is important for manual thesaurus construction, GLE as well as SAM and the traditional corpus-based methods finds its own place in this task.

8 Conclusion

We proposed a new approach that allows us to automatically expand existing small thesaurus by suggesting synonym candidates for possible inclusions into the thesaurus. While more traditional corpus-based methods use contextual features obtained from the corpus outside the thesaurus to represent a word, to induce a similar vector representation of a word, our proposed method uses the structure of a synonym network that the thesaurus has.

Our experiments found that, with a modest size of an existing thesaurus, expanding it is much easier with the proposed method based on Graph Laplacian Embedding than the traditional synonym acquisition methods. This proves that if you can expect a reasonable overlap between the existing synonym network and synonym candidates of the queries as well as the bridge pairs, the proposed Graph Laplacian Embedding is the method of choice for anyone who constructs a thesaurus and in need of a method with a high recall. We

have also given unifying interpretations of eigenvectors to allow intuitive interpretations of our method. Since the use of non-principal eigenvectors is still quite limited in NLP, we hope that our exposition proves useful for many researchers.

Acknowledgement

We would like to acknowledge and thank the anonymous reviewers. Especially the section on the experiments is significantly improved thanks to their comments.

References

[1] G. Grefenstette, Explorations in Automatic Thesuarus Discovery, Kluwer Academic Publisher, 1994.

[2] Y. Jing and B. Croft, "An association thesaurus for information retrieval," Proc. of Recherche d'Informations Assistée par Ordinateur (RIAO), pp.146–160, 1994.

[3] S. Buttcher, C. Clarke, and G. Cormack, "Domain-specific synonym expansion and validation for biomedical information retrieval (multitext experiments for trec 2004)," The Thirteenth Text Retrieval Conference (TREC 2004), 2004.

[4] C. Fellbaum, WordNet: an electronic lexical database, MIT Press, 1998.

[5] P. Resnik, "Using information content to evaluate semantic similarity," Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, pp.448–453, 1995.

[6] D. Hindle, "Noun classification from predicate-argument structures," Proc. of the 28th Annual Meeting of the ACL, pp.268–275, 1990.

[7] D. Lin, "Automatic retrieval and clustering of similar words," Proc. of the 36th Annual Meeting of the ACL, pp.786–774, 1998.

[8] Z. Harris, Distributional Structure, Oxford University Press, 1985.

[9] L. Lee, "Measures of distributional similarity," Proc. of the 37th Annual Meeting of the ACL, pp.23–32, 1999.

[10] J.R. Curran and M. Moens, "Improvements in automatic thesaurus extraction. in workshop on unsupervised lexical acquisition," Proc. of ACL SIGLEX, pp.231–238, 2002.

[11] J. Weeds, D. Weir, and D. McCarthy, "Characterising measures of lexical distributional similarity," Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING), pp.1015–1021, 2004.

[12] M. Geffet and I. Dagan, "Feature vector quality and distributional similarity," Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING), 2004.

[13] R. Snow, D. Jurafsky, and A.Y. Ng, "Semantic taxonomy induction from heterogenous evidence," Proceedings of Coling-ACL, pp.801–808, 2006.

[14] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," Computer Networks and ISDN Systems, 30(1-7), 1998.

[15] R. Mihalcea, "Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling," Proceedings of HLT-EMNLP, pp.411–418, 2005.

[16] R. Mihalcea, "Language independent extractive summarization," Proceedings of the ACL Interactive Poster and Demonstration Sessions, pp.49–52, 2005.

[17] M. Komachi, T. Kudo, M. Shimbo, and Y. Matsumoto, "Graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms," Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pp.1011–1020, 2008.

[18] A.Y. Ng, M.I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," Advances in Neural Information Processing Systems 14, ed. T.G. Dietterich, S. Becker, and Z. Ghahramani, Cambridge, MA, MIT Press, 2002.

[19] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," Advances in Neural Information Processing Systems 14, ed. T.G. Dietterich, S. Becker, and Z. Ghahramani, Cambridge, MA, MIT Press, 2002.

[20] J. Shi and J. Malik, "Normalized cuts and image segmentation," IEEE Transactions Pattern Analysis and Machine Intelligence, Volume: 22, Issue: 8, pp.888–905, 2000.

[21] M. Maila and J. Shi, "A random walks view of spectral segmentation," AI and STATISTICS 2001(AISTATS), 2001.

[22] B. Nadler, S. Lafon, R. Coifman, and I.G. Kevrekidis, "Diffusion maps - a probabilistic interpretation for spectral embedding and clustering algorithms," Lecture Notes in Computational Science And Engineering VOL 58, pp.238–260, 2007.

[23] G. Ruge, "Automatic detection of thesaurus relations for information retrieval applications," Foundations of Computer Science: Potential - Theory - Cognition, LNCS, Berlin, Germany, pp.499–506, Springer Verlag, 1997.

[24] T. Briscoe, J. Carroll, and R. Watson, "The second release of the rasp system," Proc. of the COLING/ACL 2006 Interactive Presentation Sessions, pp.77–80, 2006.

[25] T. Briscoe, J. Carroll, J. Graham, and A. Copestake, "Relational evaluation schemes," Proc. of the Beyond PARSEVAL Workshop at the Third International Conference on Language Resources and Evaluation, pp.4–8, 2002.

[26] L. Lee, "On the effectiveness of the skew divergence for statistical language analysis," Artificial Intelligence and Statistics 2001, pp.65–72, 2001.

[27] R. Caruana, T. Jachims, and L. Backstrom, "Kdd-cup 2004: results and analysis," ACM SIGKDD Explorations Newslatter, pp.95–108, 2004.