Encoding Prior Knowledge with Eigenword Embeddings

Dominique Osborne
Department of Mathematics and Statistics
University of Strathclyde
Glasgow, G1 1XH, UK
dominique.osborne.13@uni.strath.ac.uk

Shashi Narayan and Shay B. Cohen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LE, UK
{snaraya2,scohen}@inf.ed.ac.uk

Abstract

Canonical correlation analysis (CCA) is a method for reducing the dimension of data represented using two views. It has been previously used to derive word embeddings, where one view indicates a word, and the other view indicates its context. We describe a way to incorporate prior knowledge into CCA, give a theoretical justification for it, and test it by deriving word embeddings and evaluating them on a myriad of datasets.

1 Introduction

In recent years there has been an immense interest in representing words as low-dimensional continuous real-vectors, namely word embeddings. Word embeddings aim to capture lexico-semantic information such that regularities in the vocabulary are topologically represented in a Euclidean space. Such word embeddings have achieved state-of-the-art performance on many natural language processing (NLP) tasks, e.g., syntactic parsing (Socher et al., 2013), word or phrase similarity (Mikolov et al., 2013b), dependency parsing (Bansal et al., 2014), unsupervised learning (Parikh et al., 2014) and others. Since the discovery that word embeddings are useful as features for various NLP tasks, research on word embeddings has taken on a life of its own, with a vibrant community searching for better word representations in a variety of problems and datasets.

These word embeddings are often induced from large raw text capturing distributional co-occurrence information via neural networks (Bengio et al., 2003; Mikolov et al., 2013b; Mikolov et al., 2013c) or spectral methods (Deerwester et al., 1990; Dhillon et al., 2015). While these general purpose word embeddings have achieved significant improvement in various tasks in NLP, it has been discovered that further tuning of these continuous word representations for specific tasks improves their performance by a larger margin. For example, in dependency parsing, word embeddings could be tailored to capture similarity in terms of context within syntactic parses (Bansal et al., 2014) or they could be refined using semantic lexicons such as WordNet (Miller, 1995), FrameNet (Baker et al., 1998) and the Paraphrase Database (Ganitkevitch et al., 2013) to improve various similarity tasks (Yu and Dredze, 2014; Faruqui et al., 2015; Rothe and Schütze, 2015). This paper proposes a method to encode prior semantic knowledge in spectral word embeddings (Dhillon et al., 2015).

Spectral learning algorithms are of great interest for their speed, scalability, theoretical guarantees and performance in various NLP applications. These algorithms are no strangers to word embeddings either. In latent semantic analysis (LSA, (Deerwester et al., 1990; Landauer et al., 1998)), word embeddings are learned by performing SVD on the word by document matrix. Recently, Dhillon et al. (2015) have proposed to use canonical correlation analysis (CCA) as a method to learn low-dimensional real vectors, called Eigenwords. Unlike LSA based methods, CCA based methods are scale invariant and can capture multiview information such as the left and right contexts of the words. As a result, the eigenword embeddings of Dhillon et al. (2015) that were learned using the simple linear methods give accuracies comparable to or better than state of the art when compared with highly non-linear deep learning based approaches (Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et al., 2013b; Mikolov et al., 2013c).

The main contribution of this paper is a technique

to incorporate prior knowledge into the derivation of canonical correlation analysis. In contrast to previous work where prior knowledge is introduced in the off-the-shelf embeddings as a post-processing step (Faruqui et al., 2015; Rothe and Schütze, 2015), our approach introduces prior knowledge in the CCA derivation itself. In this way it preserves the theoretical properties of spectral learning algorithms for learning word embeddings. The prior knowledge is based on lexical resources such as WordNet, FrameNet and the Paraphrase Database.

Our derivation of CCA to incorporate prior knowledge is not limited to eigenwords and can be used with CCA for other problems. It follows a similar idea to the one proposed by Koren and Carmel (2003) for improving the visualization of principal vectors with principal component analysis (PCA). Our derivation represents the solution to CCA as that of an optimization problem which maximizes the distance between the two view projections of training examples, while weighting these distances using the external source of prior knowledge. As such, our approach applies to other uses of CCA in the NLP literature, such as the one of Jagarlamudi and Daumé (2012), who used CCA for transliteration, or the one of Silberer et al. (2013), who used CCA for semantically representing visual attributes.

2 Background and Notation

For an integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$. We assume the existence of a vocabulary of words, usually taken from a corpus. This set of words is denoted by $H = \{h_1, \ldots, h_{|H|}\}$. For a square matrix A, we denote by $\mathrm{diag}(A)$ a diagonal matrix B which has the same dimensions as A such that $B_{ii} = A_{ii}$ for all i. For vector $v \in \mathbb{R}^d$, we denote its ℓ_2 norm by $||v||$, i.e. $||v|| = \sqrt{\sum_{i=1}^{d} v_i^2}$. We also denote by v_j or $[v]_j$ the jth coordinate of v. For a pair of vectors u and v, we denote their dot product by $\langle u, v \rangle$.

We define a word embedding as a function f from H to $\mathbb{R}^m$ for some (relatively small) m. For example, in our experiments we vary m between 50 and 300. The word embedding function maps the word to some real-vector representation, with the intention to capture regularities in the vocabulary that are topologically represented in the corresponding Euclidean space. For example, all vocabulary words that correspond to city names could be grouped together in that space.

Research on the derivation of word embeddings that capture various regularities has greatly accelerated in recent years. Various methods used for this purpose range from low-rank approximations of co-occurrence statistics (Deerwester et al., 1990; Dhillon et al., 2015) to neural networks jointly learning a language model (Bengio et al., 2003; Mikolov et al., 2013a) or models for other NLP tasks (Collobert and Weston, 2008).

3 Canonical Correlation Analysis for Deriving Word Embeddings

One recent approach to derive word embeddings, developed by Dhillon et al. (2015), is through the use of canonical correlation analysis, resulting in so-called "eigenwords." CCA is a technique for multi-view dimensionality reduction. It assumes the existence of two views for a set of data, similarly to co-training (Yarowsky, 1995; Blum and Mitchell, 1998), and then projects the data in the two views in a way that maximizes the correlation between the projected views.

Dhillon et al. (2015) used CCA to derive word embeddings through the following procedure. They first break each document in a corpus of documents into n sequences of words of a fixed length $2k + 1$, where k is a window size. For example, if $k = 2$, the short document "Harry Potter has been a best-seller" would be broken into "Harry Potter has been a" and "Potter has been a best-seller." In each such sequence, the middle word is identified as a pivot.

This leads to the construction of the following training set from a set of documents: $\{(w_1^{(i)}, \ldots, w_k^{(i)}, w^{(i)}, w_{k+1}^{(i)}, \ldots, w_{2k}^{(i)}) \mid i \in [n]\}$. With abuse of notation, this is a multiset, as certain words are expected to appear in certain contexts multiple times. Each $w^{(i)}$ is a pivot word, and the rest of the elements are words in the sequence called "the context words." With this training set in mind, the two views for CCA are defined as following.

We define the first view through a sparse "context matrix" $C \in \mathbb{R}^{n \times 2k|H|}$ such that each row in the matrix is a vector, consisting of $2k$ one-hot vectors, each of length $|H|$. Each such one-hot vector corre-

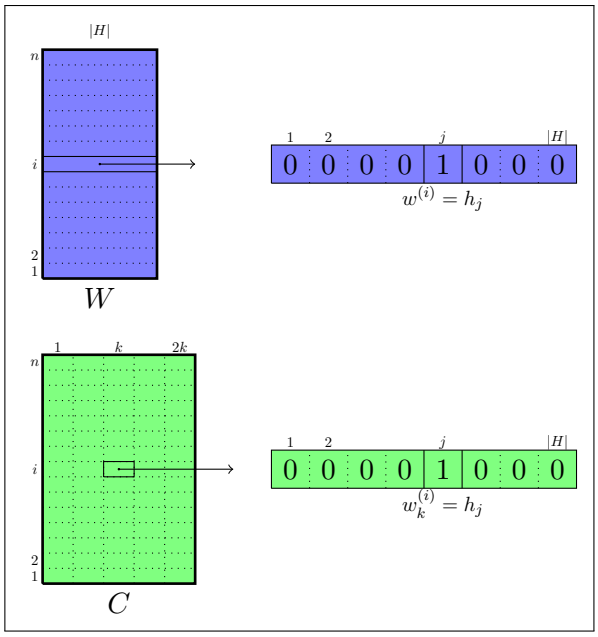

Figure 1: The word and context views represented as matrix W and C. Each row in W is a vector of length $|H|$, corresponding to a one-hot vector for the word in the example indexed by the row. Each row in C is a vector of length $2k|H|$, divided into sub-vectors each of length $|H|$. Each such sub-vector is a one-hot vector for one of the $2k$ context words in the example indexed by the row.

sponds to a word that fired in a specific index in the context. In addition, we also define a second view through a matrix $W \in \mathbb{R}^{n \times |H|}$ such that $W_{ij} = 1$ if $w^{(i)} = h_j$. We present both views of the training set in Figure 1.

Note that now the matrix $M = W^\top C$ is in $\mathbb{R}^{|H| \times (2k|H|)}$ such that each element M_{ij} gives the count of times that h_i appeared with the corresponding context word and context index encoded by j.

Similarly, we define a matrix $D_1 = \text{diag}(W^\top W)$ and $D_2 = \text{diag}(C^\top C)$. Finally, to get the word embeddings, we perform singular value decomposition (SVD) on the matrix $D_1^{-1/2} M D_2^{-1/2}$. Note that in its original form, CCA requires use of $W^\top W$ and $C^\top C$ in their full form, and not just the corresponding diagonal matrices D_1 and D_2; however, in practice, inverting these matrices can be quite intensive computationally and can lead to memory issues. As such, we approximate CCA by using the diagonal matrices D_1 and D_2.

From the SVD step, we get two projections $U \in \mathbb{R}^{|H| \times m}$ and $V \in \mathbb{R}^{2k|H| \times m}$ such that

$$D_1^{-1/2} M D_2^{-1/2} \approx U \Sigma V^\top$$

where $\Sigma \in \mathbb{R}^{m \times m}$ is a diagonal matrix with $\Sigma_{ii} > 0$ being the ith largest singular value of $D_1^{-1/2} M D_2^{-1/2}$. In order to get the final word embeddings, we calculate $D_1^{-1/2} U \in \mathbb{R}^{|H| \times m}$. Each row in this matrix corresponds to an m-dimensional vector for the corresponding word in the vocabulary. This means that $f(h_i)$ for $h_i \in H$ is the ith row of the matrix $D_1^{-1/2} U$. The projection V can be used to get "context embeddings." See more about this in Dhillon et al. (2015).

This use of CCA to derive word embeddings follows the usual distributional hypothesis (Harris, 1957) that most word embeddings techniques rely on. In the case of CCA, this hypothesis is translated into action in the following way. CCA finds projections for the contexts and for the pivot words which are most correlated. This means that if a word co-occurs in a specific context many times (either directly, or transitively through similarity to other words), then this context is expected to be projected to a point "close" to the point to which the word is projected. As such, if two words occur in a specific context many times, these two words are expected to be projected to points which are close to each other.

For the next section, we denote $X = W D_1^{-1/2}$ and $Y = C D_2^{-1/2}$. To refer to the dimensions of X and Y generically, we denote $d = |H|$ and $d' = 2k|H|$. In addition, we refer to the column vectors of U and V as $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$.

Mathematical Intuition Behind CCA The procedure that CCA follows finds a projection of the two views in a shared space, such that the correlation between the two views is maximized at each coordinate, and there is minimal redundancy between the coordinates of each view. This means that CCA solves the following sequence of optimization problems for $j \in [m]$ where $a_j \in \mathbb{R}^{1 \times d}$ and $b_j \in \mathbb{R}^{1 \times d'}$:

$$\begin{aligned} \arg\max_{a_j, b_j} \quad & \text{corr}(a_j W^\top, b_j C^\top) \\ \text{such that} \quad & \text{corr}(a_j W^\top, a_k W^\top) = 0, \quad k < j \\ & \text{corr}(b_j C^\top, b_k C^\top) = 0, \quad k < j \end{aligned}$$

where corr is a function that accepts two vectors and return the Pearson correlation between the pairwise elements of the two vectors. The approximate solution to this optimization problem (when using diagonal D_1 and D_2) is $\hat{a}_i^\top = D_1^{-1/2} u_i$ and $\hat{b}_i^\top = D_2^{-1/2} v_i$ for $i \in [m]$.

CCA also has a probabilistic interpretation as a maximum likelihood solution of a latent variable model for two normal random vectors, each drawn based on a third latent Gaussian vector (Bach and Jordan, 2005).

The way we describe CCA for deriving word embeddings is related to Latent Semantic Indexing (LSI), which performs singular value decomposition on the matrix M directly, without doing any kind of variance normalization. Dhillon et al. (2015) describe some differences between LSI and CCA. The extra normalization step decreases the importance of frequent words when doing SVD.

4 Incorporating Prior Knowledge into Canonical Correlation Analysis

In this section, we detail the technique we use to incorporate prior knowledge into the derivation of canonical correlation analysis. The main motivation behind our approach is to improve the optimization of correlation between the two views by weighing them using the external source of prior knowledge. The prior knowledge is based on lexical resources such as WordNet, FrameNet and the Paraphrase Database. Our approach follows a similar idea to the one proposed by Koren and Carmel (2003) for improving the visualization of principal vectors with principal component analysis (PCA). It is also related to Laplacian manifold regularization (Belkin et al., 2006).

An important notion in our derivation is that of a *Laplacian matrix*. The Laplacian of an undirected weighted graph is an $n \times n$ matrix where n is the number of nodes in the graph. It equals $D - A$ where A is the adjacency matrix of the graph (so that A_{ij} is the weight for the edge (i, j) in the graph, if it exists, and 0 otherwise) and D is a diagonal matrix such that $D_{ii} = \sum_j A_{ij}$. The Laplacian is always a symmetric square matrix such that the sum over rows (or columns) is 0. It is also positive semi-definite.

We propose a generalization of CCA, in which we introduce a Laplacian matrix into the derivation of CCA itself, as shown in Figure 2. We encode prior knowledge about the distances between the projections of two views into the Laplacian. The Laplacian allows us to improve the optimization of the correlation between the two views by weighing them using the external source of prior knowledge.

4.1 Generalization of CCA

We present three lemmas (proofs are given in Appendix A), followed by our main proposition. These three lemmas are useful to prove our final proposition.

The main proposition shows that CCA maximizes the distance between the two view projections for any pair of examples i and j, $i \neq j$, while minimizing the two view projection distance for the two views of an example i. The two views we discuss here in practice are the view of the word through a one-hot representation, and the view which represents the context words for a specific word token. The distance between two view projections is defined in Eq. 2.

Lemma 1. *Let X and Y be two matrices of size $n \times d$ and $n \times d'$, respectively, for example, as defined in §3. Assume that $\sum_{i=1}^{n} X_{ij} = 0$ for $j \in [d]$ and $\sum_{i=1}^{n} Y_{ij} = 0$ for $j \in [d']$. Let L be an $n \times n$ Laplacian matrix such that*

$$L_{ij} = \begin{cases} n-1 & \text{if } i = j \\ -1 & \text{if } i \neq j. \end{cases} \tag{1}$$

Then $X^\top L Y$ equals $X^\top Y$ up to a multiplication by a positive constant.

Lemma 2. *Let $A \in \mathbb{R}^{d \times d'}$. Then the rank m thin-SVD of A can be found by solving the following optimization problem:*

$$\begin{aligned} \max_{\substack{u_1,\ldots,u_m,\\ v_1,\ldots,v_m}} \quad & \sum_{i=1}^{m} u_i^\top A v_i \\ \text{such that } & ||u_i|| = ||v_i|| = 1 \qquad i \in [m] \\ & \langle u_i, u_j \rangle = \langle v_i, v_j \rangle = 0 \qquad i \neq j \end{aligned}$$

where $u_i \in \mathbb{R}^{d \times 1}$ denote the left singular vectors, and $v_i \in \mathbb{R}^{d' \times 1}$ denote the right singular vectors.

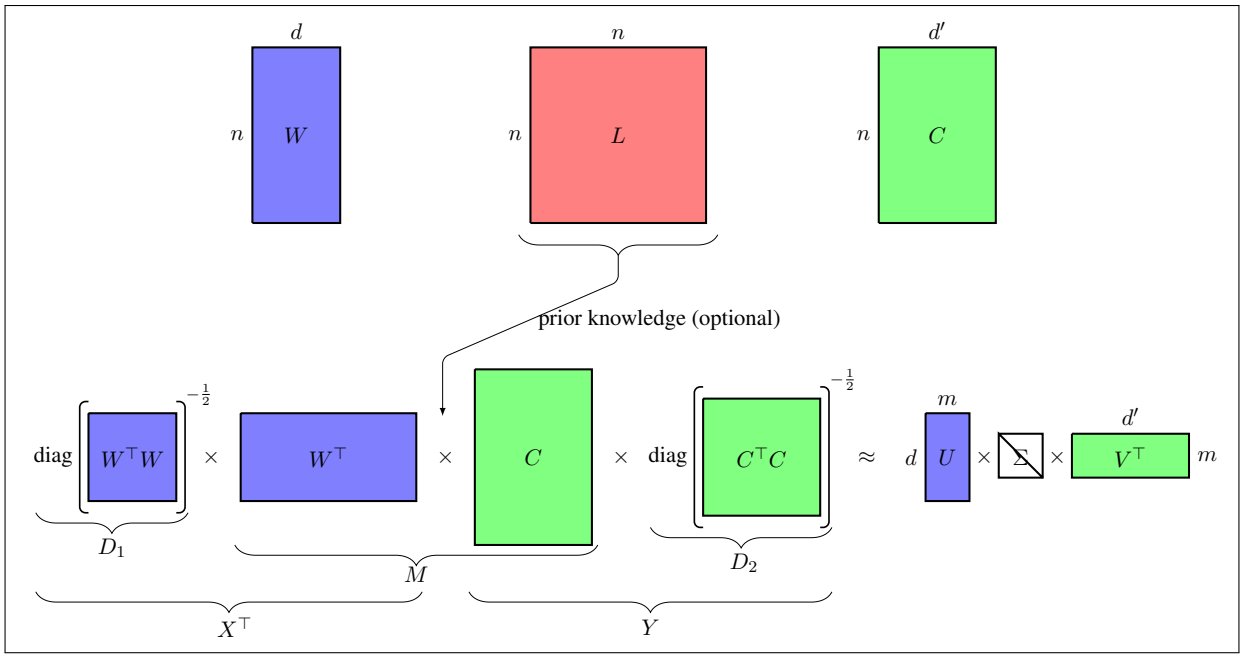

Figure 2: Introducing prior knowledge in CCA. $W \in \mathbb{R}^{n \times d}$ and $C \in \mathbb{R}^{n \times d'}$ denote the word and context views respectively. $L \in \mathbb{R}^{n \times n}$ is a Laplacian matrix encoded with the prior knowledge about the distances between the projections of W and C.

The last utility lemma we describe shows that interjecting the Laplacian between the two views can be expressed as a weighted sum of the distances between the projections of the two views (these distances are given in Eq. 2), where the weights come from the Laplacian.

Lemma 3. *Let $u_1, \ldots, u_m$ and $v_1, \ldots, v_m$ be two sets of vectors of length d and d' respectively. Let $L \in \mathbb{R}^{n \times n}$ be a Laplacian and $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times d'}$. Then:*

$$\sum_{k=1}^{m} (Xu_k)^\top L\,(Yv_k) = \sum_{i,j} -L_{ij}\left(d_{ij}^m\right)^2,$$

where

$$d_{ij}^m = \sqrt{\frac{1}{2}\left(\sum_{k=1}^{m} \left([Xu_k]_i - [Yv_k]_j\right)^2\right)}. \quad (2)$$

The following proposition is our main result for this section.

Proposition 4. *The matrices $U \in \mathbb{R}^{d \times m}$ and $V \in \mathbb{R}^{d' \times m}$ that CCA computes are the m-dimensional projections that maximize*

$$\sum_{i,j} \left(d_{ij}^m\right)^2 - n \sum_{i=1}^{n} \left(d_{ii}^m\right)^2, \quad (3)$$

where d_{ij}^m is defined as in Eq. 2 for $u_1, \ldots, u_m$ being the columns of U and $v_1, \ldots, v_m$ being the columns of V.

Proof. According to Lemma 3, the objective in Eq. 3 equals $\sum_{k=1}^{m} (Xu_k)^\top L (Yv_k)$ where L is defined as in Eq. 1. Therefore, maximizing Eq. 3 corresponds to maximization of $\sum_{k=1}^{m} (Xu_k)^\top L (Yv_k)$ under the constraints that the U and V matrices have orthonormal vectors. Using Lemma 2, it can be shown that the solution to this maximization is done by doing singular value decomposition on $X^\top LY$. According to Lemma 1, this corresponds to finding U and V by doing singular value decomposition on $X^\top Y$, because a multiplicative constant does not change the value of the right/left singular vectors. □

The above proposition shows that CCA tries to find projections of both views such that the distances between the two views for pairs of examples with indices $i \neq j$ are maximized (first term in Eq. 3), while

minimizing the distance between the projections of the two views for a specific example (second term in Eq. 3). Therefore, CCA tries to project a context and a word in that context to points that are close to each other in a shared space, while maximizing the distance between a context and a word which do not often co-occur together.

As long as L is a Laplacian, Proposition 4 is still true, only with the maximization of the objective

$$\sum_{i,j} -L_{ij} \left(d_{ij}^m\right)^2, \quad (4)$$

where $L_{ij} \leq 0$ for $i \neq j$ and $L_{ii} \geq 0$. This result lends itself to a generalization of CCA, in which we use predefined weights for the Laplacian that encode some prior knowledge about the distances that the projections of two views should satisfy.

If the weight $-L_{ij}$ is large for a specific (i, j), then we will try harder to maximize the distance between one view of example i and the other view of example j (i.e. we will try to project the word $w^{(i)}$ and the context of example j into distant points in the space).

This means that in the current formulation, $-L_{ij}$ plays the role of a *dissimiliarity* indicator between pairs of words. The more dissimilar words are, the larger the weight, and then the more distant the projections are for the contexts and the words.

4.2 From CCA with Dissimilarities to CCA with Similarities

It is often more convenient to work with *similarity* measures between pairs of words. To do that, we can retain the same formulation as before with the Laplacian, where $-L_{ij}$ now denotes a measure of similarity. Now, instead of maximizing the objective in Eq. 4, we are required to minimize it.

It can be shown that such mirror formulation can be done with an algorithm similar to CCA, leading to a proposition in the style of Proposition 4. To solve this minimization formulation, we just need to choose the singular vectors associated with the *smallest* m singular values (instead of the largest).

Once we change the CCA algorithm with the Laplacian to choose these projections, we can define L, for example, based on a similarity graph. The graph is an undirected graph that has $|H|$ nodes, for each word in the vocabulary, and there is an edge between a pair of words whenever the two words are similar to each other based on some external source of information, such as WordNet (for example, if they are synonyms).

We then define the Laplacian L such that $L_{ij} = -1$ if i and j are adjacent in the graph (and $i \neq j$), L_{ii} is the degree of the node i and $L_{ij} = 0$ in all other cases. By using this variant of CCA, we strive to maximize the distance of the two views between words which are adjacent in the graph (or continuing the example above, maximize the distance between words which are not synonyms). In addition, the fewer adjacent nodes a word has (or the more synonyms it has), the less important it is to minimize the distance between the two views of that given word.

Inputs: Set of examples $\{(w_1^{(i)}, \ldots, w_k^{(i)}, w^{(i)}, w_{k+1}^{(i)}, \ldots, w_{2k}^{(i)}) \mid i \in [n]\}$, an integer m, an $\alpha \in (0, 1]$, an undirected graph G over H, an integer N.

Data structures:

A matrix M of size $|H| \times (2k|H|)$ (cross-covariance matrix), a matrix U corresponding to the word embeddings

Algorithm:

(Cross-covariance estimation) $\forall i, j \in [n]$ such that $|i - j| \leq N$

- If $i = j$, increase M_{rs} by 1 for r denoting the index of word $w^{(i)}$ and for all s denoting the context indices of words $w_1^{(i)}, \ldots, w_k^{(i)}$ and $w_{k+1}^{(i)}, \ldots, w_{2k}^{(i)}$.
- If $i \neq j$ and word $w^{(i)}$ is connected to word $w^{(j)}$ in G, increase M_{rs} by α for r denoting the index of word $w^{(i)}$ and for all s denoting the context indices of words $w_1^{(j)}, \ldots, w_k^{(j)}$ and $w_{k+1}^{(j)}, \ldots, w_{2k}^{(j)}$.
- Calculate D_1 and D_2 as specified in §3.

(Singular value decomposition step)

- Perform singular value decomposition on $D_1^{-1/2} M D_2^{-1/2}$ to get a matrix $U \in \mathbb{R}^{|H| \times m}$.

(Word embedding projection)

- For each word h_i for $i \in [|H|]$ return the word embedding that corresponds with the ith row of U.

Figure 3: The CCA-like algorithm that returns word embeddings with prior knowledge encoded based on a similarity graph.

4.3 Final Algorithm

In order to use an arbitrary Laplacian matrix with CCA, we require that the data is centered, i.e. that the average over all examples of each of the coordinates of the word and context vectors is 0. However, such a prerequisite would make the matrices C and W dense (with many non-zero values), and hard to maintain in memory, and would also make singular value decomposition inefficient.

As such, we do not center the data to keep it sparse, and as such, use a matrix L which is not strictly a Laplacian, but that behaves better in practice.[1] Given the graph mentioned in §4 which is extracted from an external source of information, we use L such that $L_{ij} = \alpha$ for an $\alpha \in (0, 1)$ which is treated as a smoothing factor for the graph (see below the choices of α) if i and j are not adjacent in the graph, $L_{ij} = 0$ if $i \neq j$ are adjacent, and finally $L_{ii} = 1$ for all $i \in [n]$. Therefore, this matrix is symmetric, and the only constraint it does not satisfy is that of rows and columns summing to 0.

Scanning the documents and calculating the statistic matrix with the Laplacian is computationally infeasible with a large number of tokens given as input. It is quadratic in that number. As such, we make another modification to the algorithm, and calculate a "local" Laplacian. The modification requires an integer N as input (we use $N = 12$), and then it makes updates to pairs of word tokens only if they are within an N-sized window of each. The final algorithm we use is described in Figure 3. The algorithm works by directly computing the co-occurrence matrix M (instead of maintaining W and C). It does so by increasing by 1 any cells corresponding to word-context co-occurrence in the documents and by α any cells corresponding to word and contexts that are connected in the graph.

5 Experiments

In this section we describe our experiments.

5.1 Experimental Setup

Training Data We used three datasets, WIKI1, WIKI2 and WIKI5, all based on the first 1, 2 and 5 billion words from Wikipedia respectively.[2] Each dataset is broken into chunks of length 13 (window sizes of 6), corresponding to a document. The above Laplacian L is calculated within each document separately. This means that $-L_{ij}$ is 1 only if i and j denote two words that appear in the same document. This is done to make the calculations computationally feasible. We calculate word embeddings for the top most frequent 200K words.

Prior Knowledge Resources We consider three sources of prior knowledge: WordNet (Miller, 1995), the Paraphrase Database of Ganitkevitch et al. (2013), abbreviated as PPDB,[3] and FrameNet (Baker et al., 1998). Since FrameNet and WordNet index words in their base form, we use WordNet's stemmer to identify the base form for the text in our corpora whenever we calculate the Laplacian graph. For WordNet, we have an edge in the graph if one word is a synonym, hypernym or hyponym of the other. For PPDB, we have an edge if one word is a paraphrase of the other, according to the database. For FrameNet, we connect two words in the graph if they appear in the same frame.

System Implementation We modified the implementation of the `SWELL` Java package[4] of Dhillon et al. (2015). Specifically, we needed to modify the loop that iterates over words in each document to a nested loop that iterates over pairs of words, in order to compute a sum of the form $\sum_{ij} X_{ri} L_{ij} Y_{js}$.[5] Dhillon et al. (2015) use window size $k = 2$, which we retain in our experiments.[6]

5.2 Baselines

Off-the-shelf Word Embeddings We compare our word embeddings with existing state-of-the-

[1] We note that other decompositions, such as PCA, also require centering of the data, but in case of sparse data matrix, this step is not performed.

[2] We downloaded the data from `https://dumps.wikimedia.org/`, and preprocessed it using the tool available at `http://mattmahoney.net/dc/textdata.html`.

[3] We use the XL subset of the PPDB.

[4] `https://github.com/paramveerdhillon/swell`.

[5] Our implementation and the word embeddings that we calculated are available at `http://cohort.inf.ed.ac.uk/cohort/eigen/`.

[6] We also use the square-root transformation as mentioned in Dhillon et al. (2015) which controls the variance in the counts accumulated from the corpus. See a justification for this transform in Stratos et al. (2015).

		Word similarity average				Geographic analogies				NP bracketing			
		NPK	WN	PD	FN	NPK	WN	PD	FN	NPK	WN	PD	FN
Retrofitting	Glove	59.7	63.1	64.6	57.5	**94.8**	75.3	80.4	**94.8**	78.1	79.5	79.4	78.7
	Skip-Gram	64.1	65.5	**68.6**	62.3	87.3	72.3	70.5	87.7	79.9	80.4	81.5	80.5
	Global Context	44.4	50.0	50.4	47.3	7.3	4.5	18.2	7.3	79.4	79.1	80.5	80.2
	Multilingual	62.3	66.9	68.2	62.8	70.7	46.2	53.7	72.7	81.9	81.8	**82.7**	82.0
	Eigen (CCA)	59.5	62.2	63.6	61.4	89.9	79.2	73.5	89.9	81.3	81.7	81.2	80.7
CCAPrior	$\alpha = 0.1$	-	59.1	59.6	59.5	-	88.9	88.7	89.9	-	81.0	**82.4**	81.0
	$\alpha = 0.2$	-	59.9	**60.6**	60.0	-	89.1	91.3	90.1	-	81.0	81.3	80.7
	$\alpha = 0.5$	-	59.9	59.7	59.6	-	86.9	89.3	89.3	-	81.8	81.4	80.9
	$\alpha = 0.7$	-	60.7	59.3	59.5	-	86.9	89.3	92.9	-	80.3	81.2	80.8
	$\alpha = 0.9$	-	60.6	59.6	58.9	-	89.1	**93.2**	92.5	-	81.3	80.7	81.0
CCAPrior+RF	$\alpha = 0.1$	-	61.9	63.6	61.5	-	76.0	71.9	89.9	-	81.4	81.7	81.2
	$\alpha = 0.2$	-	62.6	**64.9**	61.6	-	78.0	69.3	90.1	-	81.7	81.1	80.6
	$\alpha = 0.5$	-	62.7	63.7	61.4	-	74.9	67.3	92.9	-	**81.9**	81.4	80.0
	$\alpha = 0.7$	-	63.3	63.0	61.0	-	77.4	65.6	90.3	-	81.0	80.8	80.4
	$\alpha = 0.9$	-	62.0	63.3	60.4	-	77.3	66.2	**92.5**	-	81.0	80.7	80.4

Table 1: Results for the word similarity datasets, geographic analogies and NP bracketing. The first upper blocks (A–C) present the results with retrofitting. NPK stands for no prior knowledge (no retrofitting is used), WN for WordNet, PD for PPDB and FN for FrameNet. Glove, Skip-Gram, Global Context, Multilingual and Eigen are the word embeddings of Pennington et al. (2014), Mikolov et al. (2013b), Huang et al. (2012), Faruqui and Dyer (2014) and Dhillon et al. (2015) respectively. The second middle blocks (D–F) show the results of our eigenword embeddings encoded with prior knowledge using our method. Each row in the block corresponds to a specific use of an α value (smoothing factor), as described in Figure 3. In the lower blocks (G–I) we take the word embeddings from the second block, and retrofit them using the method of Faruqui et al. (2015). Best results in each block are in bold.

art word embeddings, such as Glove (Pennington et al., 2014), Skip-Gram (Mikolov et al., 2013b), Global Context (Huang et al., 2012) and Multilingual (Faruqui and Dyer, 2014). We also compare our word embeddings with the Eigen word embeddings of Dhillon et al. (2015) without any prior knowledge.

Retrofitting for Prior Knowledge We compare our approach of incorporating prior knowledge into the derivation of CCA against the previous works where prior knowledge is introduced in the off-the-shelf embeddings as a post-processing step (Faruqui et al., 2015; Rothe and Schütze, 2015). In this paper, we focus on the retrofitting approach of Faruqui et al. (2015).

Retrofitting works by optimizing an objective function which has two terms: one that tries to keep the distance between the word vectors close to the original distances, and the other which enforces the vectors of words which are adjacent in the prior knowledge graph to be close to each other in the new embedding space. We use the retrofitting package[7] to compare our results in different settings against the results of retrofitting of Faruqui et al. (2015).

5.3 Evaluation Benchmarks

We evaluated the quality of our eigenword embeddings on three different tasks: word similarity, geographic analogies and NP bracketing.

Word Similarity For the word similarity task we experimented with 11 different widely used benchmarks. The WS-353-ALL dataset (Finkelstein et al., 2002) consists of 353 pairs of English words with their human similarity ratings. Later, Agirre et al. (2009) re-annotated WS-353-ALL for similarity (WS-353-SIM) and relatedness (WS-353-REL) with specific distinctions between them. The SimLex-999 dataset (Hill et al., 2015) was built to measure how well models capture similarity, rather than relatedness or association. The MEN-TR-3000 dataset (Bruni et al., 2014) consists of 3000 word pairs

[7] https://github.com/mfaruqui/retrofitting.

sampled from words that occur at least 700 times in a large web corpus. The datasets, MTurk-287 (Radinsky et al., 2011) and MTurk-771 (Halawi et al., 2012), were scored by Amazon Mechanical Turk workers for relatedness of English word pairs. The YP-130 (Yang and Powers, 2005) and Verb-143 (Baker et al., 2014) datasets were developed for verb similarity predictions. The last two datasets, MC-30 (Miller and Charles, 1991) and RG-65 (Rubenstein and Goodenough, 1965) consist of 30 and 65 noun pairs respectively.

For each dataset, we calculate the cosine similarity between the vectors of word pairs and measure Spearman's rank correlation coefficient between the scores produced by the embeddings and human ratings. We report the average of the correlations on all 11 datasets. Each word similarity task in the above list represents a different aspect of word similarity, and as such, averaging the results points to the quality of the word embeddings on several tasks. We later analyze specific datasets.

Geographic Analogies Mikolov et al. (2013c) created a test set of analogous word pairs such as a:b c:d raising the analogy question of the form "a is to b as c is to __" where d is unknown. We report results on a subset of this dataset which focuses on finding capitals of common countries, e.g., *Greece* is to *Athens* as *Iraq* is to __. This dataset consists of 506 word pairs. For given word pairs, a:b c:d where d is unknown, we use the vector offset method (Mikolov et al., 2013b), i.e., we compute a vector $v = v_b - v_a + v_c$ where v_a, v_b and v_c are vector representations of the words a, b and c respectively; we then return the word d with the greatest cosine similarity to v.

NP Bracketing Here the goal is to identify the correct bracketing of a three-word noun (Lazaridou et al., 2013). For example, the bracketing of *annual (price growth)* is "right," while the bracketing of *(entry level) machine* is "left." Similarly to Faruqui and Dyer (2015), we concatenate the word vectors of the three words, and use this vector for binary classification into left or right.

Since most of the datasets that we evaluate on in this paper are not standardly separated into development and test sets, we report all results we calculated (with respect to hyperparameter differences) and do not select just a subset of the results.

5.4 Evaluation

Preliminary Experiments In our first set of experiments, we vary the dimension of the word embedding vectors. We try $m \in \{50, 100, 200, 300\}$. Our experiments showed that the results consistently improve when the dimension increases for all the different datasets. For example, for $m = 50$ and WIKI1, we get an average of 46.4 on the word similarity tasks, 50.1 for $m = 100$, 53.4 for $m = 200$ and 54.2 for $m = 300$. The more data are available, the more likely larger dimension will improve the quality of the word embeddings. Indeed, for WIKI5, we get an average of 49.4, 54.9, 57.0 and 59.5 for each of the dimensions. The improvements with respect to the dimension are consistent across all of our results, so we fix m at 300.

We also noticed a consistent improvement in accuracy when using more data from Wikipedia. For example, for $m = 300$, using WIKI1 gives an average of 54.1, while using WIKI2 gives an average of 54.9 and finally, using WIKI5 gives an average of 59.5. We fix the dataset we use to be WIKI5.

Results Table 1 describes the results from our first set of experiments. (Note that the table is divided into 9 distinct blocks, labeled A through I.) In general, adding prior knowledge to eigenword embeddings does improve the quality of word vectors for the word similarity, geographic analogies and NP bracketing tasks on several occasions (blocks D–F compared to last row in blocks A–C). For example, our eigenword vectors encoded with prior knowledge (CCAPrior) consistently perform better than the eigenword vectors that do not have any prior knowledge for the word similarity task (59.5, Eigen in the first row under NPK column, versus block D). The only exceptions are for $\alpha = 0.1$ with WordNet (59.1), for $\alpha = 0.7$ with PPDB (59.3) and for $\alpha = 0.9$ with FrameNet (58.9), where α denotes the smoothing factor.

In several cases, running the retrofitting algorithm of Faruqui et al. (2015) on top of our word embeddings helps further, as if "adding prior knowledge twice is better than once." Results for these word embeddings (CCAPrior+RF) are shown in Table 1. Adding retrofitting to our encoding of prior knowl-

edge often performs better for word similarity and NP bracketing tasks (block D versus G and block F versus I). Interestingly, CCAPrior+RF embeddings also often perform better than eigenword vectors (Eigen) of Dhillon et al. (2015) when retrofitted using the method of Faruqui et al. (2015). For example, in the word similarity task, eigenwords retrofitted with WordNet get an accuracy of 62.2 whereas encoding prior knowledge using both CCA and retrofitting gets a maximum accuracy of 63.3. We see the same pattern for PPDB, with 63.6 for "Eigen" and 64.9 for "CCAPrior+RF". We hypothesize that the reason for these changes is that the two methods for encoding prior knowledge maximize different objective functions.

The performance with FrameNet is weaker, in some cases leading to worse performance (e.g., with Glove and SG vectors). We believe that FrameNet does not perform as well as the other lexicons because it groups words based on very abstract concepts; often words with seemingly distantly related meanings (e.g., push and growth) can evoke the same frame. This also supports the findings of Faruqui et al. (2015), who noticed that the use of FrameNet as a prior knowledge resource for improving the quality of word embeddings is not as helpful as other resources such as WordNet and PPDB.

We note that CCA works especially well for the geographic analogies dataset. The quality of eigenword embeddings (and the other embeddings) degrades when we encode prior knowledge using the method of Faruqui et al. (2015). Our method improves the quality of eigenword embeddings.

Global Picture of the Results When comparing retrofitting to CCA with prior knowledge, there is a noticable difference. Retrofitting performs well or badly, depending on the dataset, while the results with CCA are more stable. We attribute this to the difference between how our algorithm and retrofitting work. Retrofitting makes a *direct* use of the source of prior knowledge, by adding a regularization term that enforces words which are similar according to the prior knowledge to be closer in the embedding space. Our algorithm, on the other hand, makes a more indirect use of the source of prior knowledge, by changing the co-occurence matrix on which we do singular value decomposition.

Specifically, we believe that our algorithm is more stable to cases in which words for the task at hand are unknown words with respect to the source of prior knowledge. This is demonstrated with the geographical analogies task: in that case, retrofitting lowers the results in most cases. The city and country names do not appear in the sources of prior knowledge we used.

Further Analysis We further inspected the results on the word similarity tasks for the RG-65 and WS-353-ALL datasets. Our goal was to find cases in which either CCA embeddings by themselves outperform other types of embeddings or that encoding prior knowledge into CCA the way we describe significantly improves the results.

For the WS-353-ALL dataset, the eigenword embeddings get a correlation of 69.6. The next best performing word embeddings are the multilingual word embeddings (68.0) and skip-gram (58.3). Interestingly enough, the multilingual word embeddings also use CCA to project words into a low-dimensional space using a linear transformation, suggesting that linear projections are a good fit for the WS-353-ALL dataset. The dataset itself includes pairs of common words with a corresponding similarity score. The words that appear in the dataset are actually expected to occur in similar contexts, a property that CCA directly encodes when deriving word embeddings.

The best performance on the RG-65 dataset is with the Glove word embeddings (76.6). CCA embeddings give an accuracy of 69.7 on that dataset. However, with this dataset, we observe significant improvement when encoding prior knowledge using our method. For example, using WordNet with this dataset improves the results by 4.2 points (73.9). Using the method of Faruqui et al. (2015) (with WordNet) on top of our CCA word embeddings improves the results even further by 8.7 points (78.4).

The Role of Prior Knowledge We also designed an experiment to test whether using distributional information is necessary for having well-performing word embeddings, or whether it is sufficient to rely on the prior knowledge resource. In order to test this, we created a sparse matrix that corresponds to the graph based on the external resource graph. We then follow up with singular value decomposition on

Resource	WordSim	NP Bracketing
WordNet	35.9	73.6
PPDB	37.5	77.9
FrameNet	19.9	74.5

Table 2: Results on word similarity dataset (average over 11 datasets) and NP bracketing. The word embeddings are derived by using SVD on the similarity graph extracted from the prior knowledge source (WordNet, PPDB and FrameNet).

that graph, and get embeddings of size 300. Table 2 gives the results when using these embeddings. We see that the results are consistently lower than the results that appear in Table 1, implying that the use of prior knowledge comes hand in hand with the use of distributional information. When using the retrofitting method by Faruqui et al. on top of these word embeddings, the results barely improved.

6 Related Work

Our ideas in this paper for encoding prior knowledge in eigenword embeddings relate to three main threads in existing literature.

One of the threads focuses on modifying the objective of word vector training algorithms. Yu and Dredze (2014), Xu et al. (2014), Fried and Duh (2015) and Bian et al. (2014) augment the training objective in neural language models of Mikolov et al. (2013a) to encourage semantically related word vectors to come closer to each other. Wang et al. (2014) propose a method for jointly embedding entities (from FreeBase, a large community-curated knowledge base) and words (from Wikipedia) into the same continuous vector space. Chen and de Melo (2015) propose a similar joint model to improve the word embeddings, but rather than using structured knowledge sources their model focuses on discovering stronger semantic connections in specific contexts in a text corpus.

Another research thread relies on post-processing steps to encode prior knowledge from semantic lexicons in off-the-shelf word embeddings. The main intuition behind this trend is to update word vectors by running belief propagation on a graph extracted from the relation information in semantic lexicons. The retrofitting approach of Faruqui et al. (2015) uses such techniques to obtain higher quality semantic vectors using WordNet, FrameNet, and the Paraphrase Database. They report on how retrofitting helps improve the performance of various off-the-shelf word vectors such as Glove, Skip-Gram, Global Context, and Multilingual, on various word similarity tasks. Rothe and Schütze (2015) also describe how standard word vectors can be extended to various data types in semantic lexicons, e.g., synsets and lexemes in WordNet.

Most of the standard word vector training algorithms use co-occurrence within window-based contexts to measure relatedness among words. Several studies question the limitations of defining relatedness in this way and investigate if the word co-occurrence matrix can be constructed to encode prior knowledge directly to improve the quality of word vectors. Wang et al. (2015) investigate the notion of relatedness in embedding models by incorporating syntactic and lexicographic knowledge. In spectral learning, Yih et al. (2012) augment the word co-occurrence matrix on which LSA operates with relational information such that synonyms will tend to have positive cosine similarity, and antonyms will tend to have negative similarities. Their vector space representation successfully projects synonyms and antonyms on opposite sides in the projected space. Chang et al. (2013) further generalize this approach to encode multiple relations (and not just opposing relations, such as synonyms and antonyms) using multi-relational LSA.

In spectral learning, most of the studies on incorporating prior knowledge in word vectors focus on LSA based word embeddings (Yih et al., 2012; Chang et al., 2013; Turney and Littman, 2005; Turney, 2006; Turney and Pantel, 2010).

From the technical perspective, our work is also related to that of Jagarlamudi et al. (2011), who showed how to generalize CCA so that it uses locality preserving projections (He and Niyogi, 2004). They also assume the existence of a weight matrix in a multi-view setting that describes the distances between pairs of points in the two views.

More generally, CCA is an important component for spectral learning algorithms in the unsupervised setting and with latent variables (Cohen et al., 2014; Narayan and Cohen, 2016; Stratos et al., 2016). Our method for incorporating prior knowledge into CCA could potentially be transferred to these algorithms.

7 Conclusion

We described a method for incorporating prior knowledge into CCA. Our method requires a relatively simple change to the original canonical correlation analysis, where extra counts are added to the matrix on which singular value decomposition is performed. We used our method to derive word embeddings in the style of eigenwords, and tested them on a set of datasets. Our results demonstrate several advantages of encoding prior knowledge into eigenword embeddings.

Acknowledgements

The authors would like to thank Paramveer Dhillon for his help with running the `SWELL` package. The authors would also like to thank Manaal Faruqui and Sujay Kumar Jauhar for their help and technical assistance with the retrofitting package and the word embedding evaluation suite. Thanks also to Ankur Parikh for early discusions on this project. This work was completed while the first author was an intern at the University of Edinburgh, as part of the Equate Scotland program. This research was supported by an EPSRC grant (EP/L02411X/1) and an EU H2020 grant (688139/H2020-ICT-2015; SUMMA).

Appendix A: Proofs

Proof of Lemma 1. The proof is similar to the one that appears in Koren and Carmel (2003) for Lemma 3.1. The only difference is the use of two views. Note that $[X^\top LY]_{ij} = \sum_{k,k'} X_{ki} L_{kk'} Y_{k'j}$. As such,

$$\begin{aligned}
[X^\top LY]_{ij} &= \sum_{k,k'} (n\delta_{kk'} - 1) X_{ki} Y_{k'j} \\
&= \sum_{k=1}^{n} n X_{ki} Y_{kj} - \underbrace{\left(\sum_{k=1}^{n} X_{ki}\right)}_{0} \times \underbrace{\left(\sum_{k'=1}^{n} Y_{k'j}\right)}_{0} \\
&= n[X^\top Y]_{ij},
\end{aligned}$$

where $\delta_{kk'} = 1$ iff $k = k'$ and 0 otherwise, and the second equality relies on the assumption of the data being centered. □

Proof of Lemma 2. Without loss of generality, assume $d \leq d'$. Let $u'_1, \ldots, u'_d$ be the left singular vectors of A and $v'_1, \ldots, v'_{d'}$ be the right ones, and $\sigma_1, \ldots, \sigma_d$ be the singular values. Therefore $A = \sum_{j=1}^{d} \sigma_j u'_j (v'_j)^\top$. In addition, the objective equals (after substituting A):

$$\sum_{i=1}^{m}\sum_{j=1}^{d} \sigma_j \langle u_i, u'_j\rangle\langle v_i, v'_j\rangle = \sum_{j=1}^{d} \sigma_j \left(\sum_{i=1}^{m} \langle u_i, u'_j\rangle\langle v_i, v'_j\rangle\right) \tag{5}$$

Note that by the Cauchy-Schwartz inequality:

$$\begin{aligned}
\sum_{j=1}^{d}\sum_{i=1}^{m} \langle u_i, u'_j\rangle\langle v_i, v'_j\rangle &= \sum_{i=1}^{m}\sum_{j=1}^{d} \langle u_i, u'_j\rangle\langle v_i, v'_j\rangle \\
&\leq \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{d} |\langle u_i, u'_j\rangle|^2} \sqrt{\sum_{j=1}^{d} |\langle v_i, v'_j\rangle|^2} \leq m
\end{aligned}$$

In addition, note that if we choose $u_i = u'_i$ and $v_i = v'_i$, then the inequality above becomes an equality, and in addition, the objective in Eq. 5 will equal the sum of the m largest singular vectors $\sum_{j=1}^{m} \sigma_j$. As such, this assignment to u_i and v_i maximizes the objective. □

Proof of Lemma 3. First, by definition of matrix multiplication,

$$\sum_{k=1}^{m} (Xu_k)^\top L\,(Yv_k) = \sum_{i,j} L_{ij} \left(\sum_{k=1}^{m} [Xu_k]_i [Yv_k]_j\right). \tag{6}$$

Also,

$$\left(d_{ij}^m\right)^2 = \frac{1}{2}\left(\sum_{k=1}^{m} [Xu_k]_i^2 - 2[Xu_k]_i[Yv_k]_j + [Yv_k]_j^2\right).$$

Therefore,

$$\begin{aligned}
&2\sum_{i,j} -L_{ij} \left(d_{ij}^m\right)^2 \\
&\quad= \sum_{i,j} -L_{ij} \left(\sum_{k=1}^{m} -2[Xu_k]_i[Yv_k]_j\right) \\
&\qquad + \underbrace{\sum_{i,j} -L_{ij} \left(\sum_{k=1}^{m} [Xu_k]_i^2 + [Yv_k]_j^2\right)}_{0} \\
&\quad= 2\sum_{i,j} L_{ij} \left(\sum_{k=1}^{m} [Xu_k]_i [Yv_k]_j,\right)
\end{aligned} \tag{7}$$

where the first two terms disappear because of the definition of the Laplacian. The comparison of Eq. 6 to Eq. 7 gives us the necessary result. □

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*.

Francis Bach and Michael Jordan. 2005. A probabilistic interpretation of canonical correlation analysis. Tech Report 688, Department of Statistics, University of California, Berkeley.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of ACL*.

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of EMNLP*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Machine Learning and Knowledge Discovery in Databases*, volume 8724 of *Lecture Notes in Computer Science*, pages 132–148.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of EMNLP*.

Jiaqiang Chen and Gerard de Melo. 2015. Semantic information extraction for improved word embeddings. In *Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Shay B. Cohen, K. Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *Journal of Machine Learning Research*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

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(6):391–407.

Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research*, 16:3035–3078.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *Proceedings of ACL*.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Lev Finkelstein, Gabrilovich Evgenly, Matias Yossi, Rivlin Ehud, Solan Zach, Wolfman Gadi, and Ruppin Eytan. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Daniel Fried and Kevin Duh. 2015. Incorporating both distributional and relational semantics in word representations. In *Proceedings of ICLR*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL*.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of ACM SIGKDD*.

Zellig S. Harris. 1957. Co-occurrence and transformation in linguistic structure. *Language*, 33(3):283–340.

Xiaofei He and Partha Niyogi. 2004. Locality preserving projections. In *Proceedings of NIPS*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*.

Jagadeesh Jagarlamudi and Hal Daumé. 2012. Regularized interlingual projections: Evaluation on multilingual transliteration. In *Proceedings of EMNLP-CoNLL*.

Jagadeesh Jagarlamudi, Raghavendra Udupa, and Hal Daumé. 2011. Generalization of CCA via spectral embedding. In *Proceedings of the Snowbird Learning Workshop of AISTATS*.

Yehuda Koren and Liran Carmel. 2003. Visualization of labeled data using linear transformations. In *Proceedings of IEEE Conference on Information Visualization*.

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

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proceedings of EMNLP*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*.

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

George A Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of ICML*.

Shashi Narayan and Shay B. Cohen. 2016. Optimizing spectral learning for parsing. In *Proceedings of ACL*.

Ankur P. Parikh, Shay B. Cohen, and Eric Xing. 2014. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of ACM WWW*.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL-IJCNLP*.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *Proceedings of ACL*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of ACL*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*, 4:245–257.

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

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

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

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of EMNLP*.

Tong Wang, Abdelrahman Mohamed, and Graeme Hirst. 2015. Learning lexical embeddings with syntactic and lexicographic knowledge. In *Proceedings of ACL-IJCNLP*.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the ACM CIKM*.

Dongqiang Yang and David MW Powers. 2005. Measuring semantic similarity in the taxonomy of WordNet. In *Proceedings of the Australasian Conference on Computer Science*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*.

Wen-tau Yih, Geoffrey Zweig, and John Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of EMNLP-CoNLL*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*.