Journal of Machine Learning Research 16 (2015) 3035-3078 Submitted 8/14; Revised 3/15; Published 12/15

Eigenwords: Spectral Word Embeddings

Paramveer S. Dhillon* DHILLON@MIT.EDU
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142, USA

Dean P. Foster FOSTER@WHARTON.UPENN.EDU
Department of Statistics
The Wharton School, University of Pennsylvania
Philadelphia, PA 19104, USA

Lyle H. Ungar UNGAR@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: Ivan Titov

Abstract

Spectral learning algorithms have recently become popular in data-rich domains, driven in part by recent advances in large scale randomized SVD, and in spectral estimation of Hidden Markov Models. Extensions of these methods lead to statistical estimation algorithms which are not only fast, scalable, and useful on real data sets, but are also provably correct. Following this line of research, we propose four fast and scalable spectral algorithms for learning word embeddings – low dimensional real vectors (called *Eigenwords*) that capture the "meaning" of words from their context. All the proposed algorithms harness the multi-view nature of text data i.e. the left and right context of each word, are fast to train and have strong theoretical properties. Some of the variants also have lower sample complexity and hence higher statistical power for rare words. We provide theory which establishes relationships between these algorithms and optimality criteria for the estimates they provide. We also perform thorough qualitative and quantitative evaluation of *Eigenwords* showing that simple linear approaches give performance comparable to or superior than the state-of-the-art non-linear deep learning based methods.

Keywords: spectral learning, CCA, word embeddings, NLP

1. Introduction

In recent years there has been immense interest in learning embeddings for words from large amounts of raw text[1]. Word embeddings map each word in text to a 'k' dimensional (~ 50) real valued vector. They are typically learned in a totally unsupervised manner by exploiting the co-occurrence structure of words in unlabeled text. Ideally these embeddings should capture a rich variety of information about that word, including topic, part of speech,

*. This work was done when PSD was a graduate student at the University of Pennsylvania.
1. This paper is based in part on work in (Dhillon et al., 2011),(Dhillon et al., 2012b).

word features such as animacy, sentiment, gender, whether the numbers are years or small numbers, and the direction of sentiment (happy vs. sad).

The importance of word embeddings has been amplified by the fact that over the past decade there has been increased interest in using unlabeled data to supplement the labeled data in semi-supervised learning. Semi-supervised learning reduces data sparsity and gives improved generalization accuracies in high dimensional domains like NLP. Approaches like (Ando and Zhang, 2005; Suzuki and Isozaki, 2008) have been empirically very successful, achieving excellent accuracies on a variety of NLP tasks. However, it is often difficult to adapt these approaches to use in conjunction with an existing supervised NLP system as they enforce a particular choice of model.

An increasingly popular alternative is to learn representational embeddings for words from a large collection of unlabeled data, either using a generative model or an artificial neural network, and to use these embeddings to augment the feature set of a supervised learner, thereby improving the performance of a state-of-the-art NLP system such as a sentiment analyzer, parser or part of speech tagger.

Word embeddings have proven useful and have given state-of-the-art performance on many natural language processing tasks e.g. syntactic parsing (Täckström et al., 2012; Parikh et al., 2014), POS Tagging (Dhillon et al., 2012b; Huang et al., 2013), dependency parsing (Bansal et al., 2014; Koo et al., 2008; Dhillon et al., 2012a), sentiment analysis (Dhillon et al., 2012b), chunking (Turian et al., 2010; Dhillon et al., 2011), Named Entity Recognition (NER) (Turian et al., 2010; Dhillon et al., 2011), word analogies (Mikolov et al., 2013a,b) and word similarity (Huang et al., 2012) to name a few.

These NLP systems use labeled data to learn a model, but there is often only a limited amount of labeled text available for these tasks. (This is less of a problem for English, but other languages often have very little labeled data.) Thus, word embeddings, which can be learned from large amounts of unlabeled data, provide a highly discriminative set of features which enable the supervised learner to perform better.

As mentioned earlier, embedding methods produce features in low dimensional spaces, unlike the traditional approach of working in the original high dimensional vocabulary space with only one dimension "on" at a given time.

Broadly speaking, embedding methods fall into two categories:

1. *Clustering based word embeddings*: Clustering methods, often hierarchical, are used to group distributionally similar words based on their contexts. The two dominant approaches are Brown Clustering (Brown et al., 1992) and (Pereira et al., 1993). As recently shown, HMMs can also be used to induce a multinomial distribution over possible clusters (Huang and Yates, 2009).

2. *Dense embeddings*: These embeddings are dense, low dimensional and real-valued. Each dimension of these embeddings captures latent information about a combination of syntactic and semantic word properties. They can either be induced using neural networks like C&W embeddings (Collobert and Weston, 2008), *Hierarchical log-linear* (HLBL) embeddings (Mnih and Hinton, 2007), `word2vec` embeddings (Mikolov et al., 2013a,b) or by eigen-decomposition of the word co-occurrence matrix, e.g. *Latent Semantic Analysis/Latent Semantic Indexing* (LSA/LSI) (Dumais et al., 1988).

The most classic and successful algorithm for learning word embeddings is Latent Semantic Analysis (LSA) (Landauer et al., 2008), which works by performing SVD on the word by document matrix.

Unfortunately, the state-of-the-art embedding methods suffer from a number of shortcomings: 1). They are slow to train (especially, the Deep Learning based approaches (Collobert and Weston, 2008; Mnih and Hinton, 2007). Recently, (Mikolov et al., 2013a,b) have proposed neural network based embeddings which avoid using the hidden layers which are typical in Deep Learning. This, coupled with good engineering allows their embeddings to be trained in minutes. 2). Are sensitive to the scaling of the embeddings (especially ℓ_2 based approaches like LSA/PCA). 3). Learn a single embedding for a given word type; i.e. all the occurrences of the word *"bank"* will have the same embedding, irrespective of whether the context of the word suggests it means *"a financial institution"* or *"a river bank."* Recently, (Huang et al., 2012) have proposed context specific word embeddings, but their Deep Learning based approach is slow and can not scale to large vocabularies.

In this paper we provide spectral algorithms (based on eigen-decomposition) for learning word embeddings, as they have been shown to be fast and scalable for learning from large amounts of unlabeled data (Turney and Pantel, 2010), have a strong theoretical grounding, and are guaranteed to converge to globally optimal solutions (Hsu et al., 2009). Particularly, we are interested in Canonical Correlation Analysis (CCA) (Hotelling, 1935) based methods since:

1. Unlike PCA or LSA based methods, they are scale invariant and
2. Unlike LSA, they can capture multi-view information. In text applications the left and right contexts of the words provide a natural split into two views which is totally ignored by LSA as it throws the entire context into a bag of words while constructing the term-document matrix.

We propose a variety of dense embeddings; they learn real-valued word embeddings by performing Canonical Correlation Analysis (CCA) (Hotelling, 1935) between the past and future views of the data. All our embeddings have a number of common characteristics and address the shortcomings of the current state-of-the-art embeddings. In particular, they are:

1. Fast, scalable and scale invariant.
2. Provide better sample complexity[2] for rare words.
3. Can induce context-specific embeddings i.e. different embeddings for *"bank"* based on whether it means *"a financial institution"* or *"a river bank."*
4. Have strong theoretical foundations.

Most importantly, in this paper we show that simple linear methods based on eigen-decomposition of the context matrices at the simplest level give accuracies comparable to or better than state-of-the-art highly non-linear deep learning based approaches like (Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et al., 2013a,b).

2. In the sense that relative statistical efficiency is better.

The remainder of the paper is organized as follows. In the next section we give a brief overview of CCA, which forms the core of our method. The following section describes our four proposed algorithms. After a brief description of context-specific embeddings and of the efficient SVD method we use, we present a set of systematic studies. These studies evaluate our CCA variants and alternatives including those derived from deep neural networks, including C&W, HLB, SENNA, and word2vec on problems in POS tagging, word similarity, generalized sentiment classification, NER, cross-lingual WSD and semantic & syntactic analogies.

2. Brief Review: Canonical Correlation Analysis (CCA)

CCA (Hotelling, 1935) is the analog to Principal Component Analysis (PCA) for pairs of matrices. PCA computes the directions of maximum covariance between elements in a single matrix, whereas CCA computes the directions of maximal correlation between a pair of matrices. Like PCA, CCA can be cast as an eigenvalue problem on a covariance matrix, but can also be interpreted as deriving from a generative mixture model (Bach and Jordan, 2005). See (Hardoon et al., 2004) for a review of CCA with applications to machine learning.

More specifically, given n i.i.d samples from two sets of multivariate data $\mathcal{D}_z = \{z_1, \ldots, z_n\} \in \mathbb{R}^{m_1}$ and $\mathcal{D}_w = \{w_1, \ldots, w_n\} \in \mathbb{R}^{m_2}$ where pairs $(z_1,\ w_1)$ have correspondence and so on, CCA tries to find a pair of linear transformations $\boldsymbol{\phi}_z \in \mathbb{R}^{m_1 \times k}$ and $\boldsymbol{\phi}_w \in \mathbb{R}^{m_2 \times k}$, (where $k \leq m_1 \leq m_2$) such that the correlation between the projection of z onto $\boldsymbol{\phi}_z$ and w onto $\boldsymbol{\phi}_w$ is maximized. This can be expressed as the following optimization problem:

$$\max_{\phi_z, \phi_w} \frac{\boldsymbol{\phi}_z^\top \mathbf{C}_{zw} \boldsymbol{\phi}_w}{\sqrt{\boldsymbol{\phi}_z^\top \mathbf{C}_{zz} \boldsymbol{\phi}_z}\sqrt{\boldsymbol{\phi}_w^\top \mathbf{C}_{ww} \boldsymbol{\phi}_w}},$$

where $\mathbf{C}_{zw}$ $(=\sum_{i=1}^{n}(z_i - \mu_z)^\top (w_i - \mu_w))$, $\mathbf{C}_{ww}$ $(=\sum_{i=1}^{n}(w_i - \mu_w)^\top (w_i - \mu_w))$, and $\mathbf{C}_{zz}$ $(=\sum_{i=1}^{n}(z_i - \mu_z)^\top (z_i - \mu_z))$, are the sample covariance matrices and $\mu_{(\cdot)}$ are the sample means.

The above optimization problem can be solved via simple eigendecomposition (e.g. using *eig*() function in MATLAB or R). The left and right canonical correlates (ϕ_z, ϕ_w) are the 'k' principal eigenvectors corresponding to the $\lambda_1 \geq, \ldots, \geq \lambda_k$ eigenvalues of the following equations:

$$\mathbf{C}_{zz}^{-1}\mathbf{C}_{zw}\mathbf{C}_{ww}^{-1}\mathbf{C}_{wz}\boldsymbol{\phi}_z = \lambda\boldsymbol{\phi}_z,$$
$$\mathbf{C}_{ww}^{-1}\mathbf{C}_{wz}\mathbf{C}_{zz}^{-1}\mathbf{C}_{zw}\boldsymbol{\phi}_w = \lambda\boldsymbol{\phi}_w.$$

There is an equivalent formulation of CCA which allows us to compute the solution via SVD of ${\mathbf{C}_{zz}}^{-1/2}\mathbf{C}_{zw}{\mathbf{C}_{ww}}^{-1/2}$. (See the appendix for proof.)

$${\mathbf{C}_{zz}}^{-1/2}\mathbf{C}_{zw}{\mathbf{C}_{ww}}^{-1/2} = \boldsymbol{\phi}_z \Lambda \boldsymbol{\phi}_w^\top, \tag{1}$$

where $(\boldsymbol{\phi}_z, \boldsymbol{\phi}_w)$ are the left and right singular vectors and Λ is the diagonal matrix of singular values. Finally, the CCA projections are gotten by "de-whitening"[3] as ${\boldsymbol{\phi}_z}^{proj} = \mathbf{C}_{zz}^{-1/2}\boldsymbol{\phi}_z$ and ${\boldsymbol{\phi}_w}^{proj} = \mathbf{C}_{ww}^{-1/2}\boldsymbol{\phi}_w$.

3. One way to think about CCA is as "whitening" the covariance matrix. Whitening is a decorrelation transformation that transforms a set of random variables with an arbitrary covariance matrix into a set

For most of the embeddings proposed in this paper, the SVD formulation (Equation 1) is preferred since it requires fewer multiplications of large sparse matrices which is an expensive operation. Hence, we define the operation $({\phi_z}^{proj}, {\phi_w}^{proj}) \equiv \text{CCA}(\mathbf{Z}, \mathbf{W})$, where $\mathbf{Z}$ $(\in \mathbb{R}^{n \times m_1})$ and $\mathbf{W}$ $(\in \mathbb{R}^{n \times m_2})$ are the matrices constructed from the data $\mathcal{D}_z$ and $\mathcal{D}_w$ respectively.

2.1 Suitability of CCA for Learning Word Embeddings

Recently, (Foster et al., 2008) showed that CCA can exploit multi-view nature of the data and provide sufficient conditions for CCA to achieve dimensionality reduction without losing predictive power. They assume that the data was generated by the model shown in Figure 1. The two assumptions that they make are that 1) Each of the two views are independent conditional on a k-dimensional hidden state $\hbar$ and that 2) The two views provide a redundant estimate of the hidden state $\hbar$.

These two assumptions are generalization of the assumptions made by co-training (Blum and Mitchell, 1998) (Figure 2), as co-training conditions on the observed labels y and not on a more flexible representation i.e. a hidden state $\hbar$.

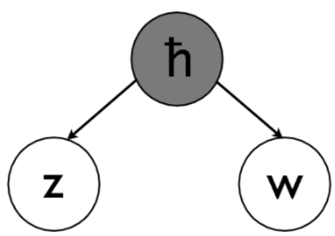

Figure 1: Multi-View Assumption. Grey color indicates that the state is hidden.

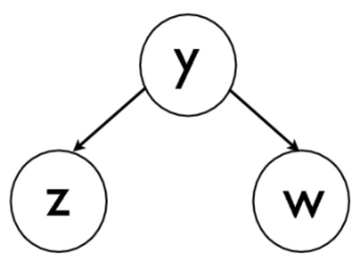

Figure 2: Co-training Assumption.

In text and Natural Language Processing (NLP) applications, its typical to assume a Hidden Markov Model (HMM) as the data generating model (Jurafsky and Martin, 2000). Its easy to see that a Hidden Markov Model (HMM) satisfies the multi-view assumption. Hence, the left and right context of a given word provides two natural views and one could use CCA to estimate the hidden state $\hbar$.

of new random variables whose covariance is the identity matrix i.e. they are uncorrelated. De-whitening, on the other hand, transforms the set of random variables to have a covariance matrix that is not an identity matrix.

Furthermore, as mentioned earlier, CCA is scale invariant and provides a natural scaling (inverse or square root of the inverse of the auto-covariance matrix, depending on whether we use Eigen-decomposition or SVD formation) for the observations. If we further use the SVD formulation, then it also allows us to harness the recent advances in large scale randomized SVD (Halko et al., 2011), which allows the embeddings learning algorithms to be fast and scalable.

The invariance of CCA to linear data transformations allows proofs that keeping the dominant singular vectors (those with largest singular values) will faithfully capture any state information (Kakade and Foster, 2007). Also, CCA extends more naturally than LSA to sequences of words[4]. Remember that LSA uses "bags of words," which are good for capturing topic information, but fail for problems like part of speech (POS) tagging which need sequence information.

Finally, as we show in the next section the CCA formulation can be naturally extended to a two step procedure that, while equivalent in the limit of infinite data, gives higher accuracies for finite corpora and provides better sample complexity.

So, in summary we estimate a hidden state associated with words by computing the dominant canonical correlations between target words and the words in their immediate context. The main computation, finding the singular value decomposition of a scaled version of the co-occurrence matrix of counts of words with their contexts, can be done highly efficiently. Use of CCA also allows us to prove theorems about the optimality of our reconstruction of the state.

In the next section we show how to efficiently compute a vector that characterizes each word type by using the left singular values of the above CCA to map from the word space (size v) to the state space (size k). We call this mapping the *eigenword dictionary* for words, as it associates with every word a vector that captures that word's syntactic and semantic attributes. As will be made clear below, the *eigenword dictionary* is arbitrary up to a rotation, but captures the information needed for any linear model to predict properties of the words such as part of speech or word sense.

3. Problem Formulation

Our goal is to estimate a vector for each word *type* that captures the distributional properties of that word in the form of a low dimensional representation of the correlation between that word and the words in its immediate context.

More formally, assume a document (in practice a concatenation of a large number of documents) consisting of n tokens $\{w_1, w_2, ..., w_n\}$, each drawn from a vocabulary of v words. Define the left and right contexts of each token w_i as the h words to the left or right of that token. The context sits in a very high dimensional space, since for a vocabulary of size v, each of the $2h$ words in the combined context requires an indicator function of dimension v. The tokens themselves sit in a v dimensional space of words which we want to project down to a k dimensional state space. We call the mapping from word types to their latent vectors the *eigenword dictionary*.

4. It is important to note that it is possible to come up with PCA variants which take sequence information into account.

For a set of documents containing n tokens, define $\mathbf{L}, \mathbf{R} \in \mathbb{R}^{n \times vh}$ as the matrices specifying the left and right contexts of the tokens, and $\mathbf{W} \in \mathbb{R}^{n \times v}$ as the matrix of the tokens themselves. In $\mathbf{W}$, we represent the presence of the j^{th} word type in the i^{th} position in a document by setting matrix element $w_{ij} = 1$. $\mathbf{L}$ and $\mathbf{R}$ are similar, but have columns for each word in each position in the context. (For example, in the sentence "I ate green apples yesterday.", for a context of size $h = 2$, the left context of "green" would be "I ate" and the right context would be "apples yesterday" and the third row of $\mathbf{W}$ would have a "1" in the column corresponding to the word "green.")

Define the complete context matrix $\mathbf{C}$ as the concatenation $[\mathbf{L}\ \mathbf{R}]$. Thus, for a trigram representation with vocabulary size v words, history size $h = 1$, $\mathbf{C}$ has $2v$ columns – one for each possible word to the left of the target word and one for each possible word to the right of the target word.

$\mathbf{W}^\top\mathbf{C}$ then contains the counts of how often each word w occurs in each context c, the matrix $\mathbf{C}^\top\mathbf{C}$ gives the covariance of the contexts, and $\mathbf{W}^\top\mathbf{W}$, the word covariance matrix, is a diagonal matrix with the counts of each word on the diagonal[5].

All the matrices i.e. $\mathbf{L}$, $\mathbf{R}$, $\mathbf{W}$ and $\mathbf{C}$, are instantiations of the underlying multivariate random variables l, r, w and c of dimensions vh, vh, v and $2vh$ respectively. We define these multivariate random variables as we will operate on them to prove the theoretical properties of some of our algorithms.

We want to find a vector representation of each of the v word types such that words that are distributionally similar (ones that have similar contexts) have similar state vectors. We will do this using Canonical Correlation Analysis (CCA) (Hotelling, 1935; Hardoon and Shawe-Taylor, 2008), by taking the CCA between the combined left and right contexts $\mathbf{C} = [\mathbf{L}\ \mathbf{R}]$ and their associated tokens, $\mathbf{W}$.

3.1 One Step CCA (OSCCA)

Using the above, we can define a "One step CCA" (OSCCA), procedure to estimate the *eigenword dictionary* as follows:

$$(\boldsymbol{\phi}_w, \boldsymbol{\phi}_c) = CCA(\mathbf{W}, \mathbf{C}), \tag{2}$$

where the $v \times k$ matrix $\boldsymbol{\phi}_w$ contains the *eigenword dictionary* that characterizes each of the v words in the vocabulary using a k dimensional vector. More generally, the "state" vectors $\mathbf{S}$ for the n tokens can be estimated either from the context as $\mathbf{C}\phi_c$ or (trivially) from the tokens themselves as $\mathbf{W}\boldsymbol{\phi}_w$. Its important to note that both these estimation procedures give a redundant estimate of the same hidden "state."

The left canonical correlates found by OSCCA give an optimal approximation to the state of each word, where "optimal" means that it gives the linear model of a given size, k that is best able to estimate labels that depend linearly on state, subject to only using the word and not its context. The right canonical correlates similarly give optimal state estimates given the context.

5. Due to the Zipfian nature of the word distribution, we will pretend that the means are all in fact zero and refer to these matrices as covariance matrices, when in fact they are second moment matrices.

OSCCA, as defined in Equation 2 thus gives an efficient way to calculate the eigenword dictionary $\boldsymbol{\phi}_w$ for a set of v words given the context and associated word matrices from a corpus.

3.1.1 Theoretical Properties

We now discuss how well the hidden state can be estimated from the target word. (A similar result can be derived for estimating hidden state from the context.) The state estimated is arbitrary up to any linear transformation, so all our comments address our ability to use the state to estimate some label which depends linearly on the state.

Keeping the dominant singular vectors in $\boldsymbol{\phi}_w$ and $\boldsymbol{\phi}_c$ provides two different bases for the estimated state. Each is optimal in its own way, as explained below.

The following Theorem 1 shows that the left canonical correlates give an optimal approximation to the state of each word (in the sense of being able to estimate an emission or label y for each state), subject to only using the word and not its context.

Theorem 1 *Let* $\{w_i, c_i, y_i\}$ *(*$\in \mathbb{R}^v \times \mathbb{R}^{hv} \times \mathbb{R}$*) for* $i = 1 \ldots n$ *be* n *observations of random variables drawn i.i.d. from some distribution (pdf or pmf)* $\mathbb{D}(w, c, y)$*. We call the pair* $(y_1 \ldots y_n, \beta)$ *a* linear context problem *if*

1. y_i *is a linear function of the context (i.e.* $y_i = \alpha^\top c_i$*).*
2. $\beta^\top w_i$ *is the best linear estimator of* y_i *given* w_i*, namely* β *minimizes* $\sum_{i=1}^n (y_i - \beta^\top w_i)^2$ *and*
3. *Var*$(y_i) \leq 1$.

Let ($\boldsymbol{\phi}_w$*,* $\boldsymbol{\phi}_c$*)* $\equiv$ *CCA(*$\mathbf{W}$*,* $\mathbf{C}$*) where* $\mathbf{W}$ *and* $\mathbf{C}$ *are the matrices constructed from* $\{w\}_{i=1}^n$ *and* $\{c\}_{i=1}^n$ *respectively. Also, let* ${\boldsymbol{\phi}_w}^i$ *be the* i^{th} *left singular vector. Then, for all* $\epsilon > 0$ *there exists a* k *such that for any linear context problem* $(y_1 \ldots y_n, \beta)$*, there exists a* $\gamma \in \mathbb{R}^k$ *such that* $\widehat{y}_i = \sum_{j=1}^k \gamma_j \boldsymbol{\phi}_w^{ji}$ *is a good approximation to* y_i *in the sense that* $\sum_{i=1}^n (\widehat{y}_i - \beta^\top w_i)^2 \leq \epsilon$*.*

Please see Appendix A for the proof.

To understand the above theorem, note that we would have liked to have a linear regression predicting some label y from the original data w. However, the original data is very high ('v') dimensional. Instead, we can first use CCA to map high dimensional vectors w to lower dimensional vectors $\boldsymbol{\phi}_w$, from which y can be predicted. For example with a few labeled examples of the form (w, y), we can recover the γ_i parameters using linear regression. The $\boldsymbol{\phi}_w$ subspace is guaranteed to hold a good approximation. A special case of interest occurs when estimating a label z $(= \alpha^\top c)$ plus zero mean noise. In this case, one can pick $y = \mathbb{E}(z)$ and proceed as above. This effectively extends the theorem to the case where the mapping from c to y is random, not deterministic.

Note that if we had used covariance rather than correlation as done by LSA/PCA then in the worst case, the key singular vectors for predicting state could be those with arbitrarily small singular values. This corresponds to the fact that for principle component regression (PCR), there is no guarantee that the largest principle components will prove predictive of an associated label.

One can think of Theorem 1 as implicitly estimating a k-dimensional hidden state from the observed w. This hidden state can be used to estimate y. Note that for Theorem 1, the state estimate is “trivial” in the sense that because it comes from the words, not the context, every occurrence of each word must give the same state estimate. This is attractive in that it associates a latent vector with every word type, but limiting in that it does not allow for any word ambiguity. The right canonical vectors allow one to estimate state from the context of a word, giving different state estimates for the same word in different contexts, as is needed for word sense disambiguation. We relegate that discussion to later in the paper, when we discuss induction of context-specific word embeddings. For now, we focus on the simpler use of left canonical covariates to map each word type to a k dimensional vector.

4. Efficient Eigenwords with Better Sample Complexity

OSCCA is optimal only in the limit of infinite data. In practice, data is, of course, always limited. In languages, lack of data comes about in two ways. Some languages are resource poor; one just does not have that many tokens of them (especially languages that lack a significant written literature). Even for most modern languages, many of the individual words in them are quite rare. Due to the Zipfian distribution of words, many words do not show up very often. A typical year’s worth of Wall Street Journal text only has “lasagna” or “backpack” a handful of times and “ziti” at most once or twice. To overcome these issues we propose a two-step procedure which gives rise to two algorithms, Two Step CCA (TSCCA) and Low-Rank Multi-View Learning (LR-MVL) that have better sample complexity for rare words.

4.1 Two Step CCA (TSCCA) for Estimating Eigenword Dictionary

We now introduce our two step procedure TSCCA of computing an *eigenword dictionary* and show theoretically that it gives better estimates than the OSCCA method described in the last section.

In the two-step method, instead of taking the CCA between the combined context $[\mathbf{L}\ \mathbf{R}]$ and the words $\mathbf{W}$, we first take the CCA between the left and right contexts and use the result of that CCA to estimate the state $\mathbf{S}$ (an empirical estimate of the true hidden state $\hbar$) of all the tokens in the corpus from their contexts. Note that we get partially redundant state estimates from the left context and from the right context; these are concatenated to make combined state estimate. This will contain some redundant information, but will not lose any of the differences in information from the left and right sides. We then take the CCA between $\mathbf{S}$ and the words $\mathbf{W}$ to get our final *eigenword dictionary*. This is summarized in Algorithm 1. The first step, the CCA between $\mathbf{L}$ and $\mathbf{R}$, must produce at least as many canonical components as the second step, which produces the final output.

The two step method requires fewer tokens of data to get the same accuracy in estimating the *eigenword dictionary* because its final step estimates fewer parameters $O(vk)$ than the OSCCA does $O(v^2)$.

Before stating the theorem, we first explain this intuitively. Predicting each word as a function of all other word combinations that can occur in the context is far sparser than predicting low dimensional state from context, and then predicting word from state. Thus, for relatively infrequent words, OSCCA should have significantly lower accuracy than the

Algorithm 1 Two step CCA

1: **Input:** $\mathbf{L}, \mathbf{W}, \mathbf{R}$
2: $(\boldsymbol{\phi}_l, \boldsymbol{\phi}_r) = CCA(\mathbf{L}, \mathbf{R})$
3: $\mathbf{S} = [\mathbf{L}\boldsymbol{\phi}_l \quad \mathbf{R}\boldsymbol{\phi}_r]$
4: $(\boldsymbol{\phi}_s, \boldsymbol{\phi}_w) = CCA(\mathbf{S}, \mathbf{W})$
5: **Output:** $\boldsymbol{\phi}_w$, the eigenword dictionary

two step version. Phrased differently, mapping from context to state and then from state to word (TSCCA) gives a more parsimonious model than mapping directly from context to word (OSCCA).

The relative ability of OSCCA to estimate hidden state compared to that of TSCCA can be summarized as follows:

Theorem 2 *Given a matrix of words,* $\mathbf{W}$ *and their associated left and right contexts,* $\mathbf{L}$ *and* $\mathbf{R}$ *with vocabulary size* v*, context size* h*, and corpus of* n *tokens. Consider a linear estimator built on the state estimates estimated by either TSCCA or OSCCA, then the ratio of their squared prediction errors (i.e. relative statistical efficiency) is* $\frac{h+k}{hv}$.

Please see Appendix A for the proof.

Since the corpora we care about (i.e. text and language corpora) usually have $vh \gg h+k$, the TSCCA procedure will in expectation correctly estimate hidden state with a much smaller number of components k than the one step procedure. Or, equivalently, for an estimated hidden state of given size k, TSCCA will correctly estimate more of the hidden state components.

As mentioned earlier, words have a Zipfian distribution so most words are rare. For such rare words, if one does a CCA between them and their contexts, one will have very few observations, and hence will get a low quality estimate of their eigenword vector. If, on the other hand, one first estimates a state vector for the rare words, and then does a CCA between this state vector and the context, the rare words can be thought of as borrowing strength from more common distributionally similar words. For example, "umbrage" (56,020) vs. "annoyance" (777,061) or "unmeritorious" (9,947) vs. "undeserving" (85,325). The numbers in parentheses are the number of occurrences of these words in the Google n-gram collection used in some of our experiments.

4.2 Low Rank Multi-View Learning (LR-MVL)

The context around a word, consisting of the h words to the right and left of it, sits in a high dimensional space, since for a vocabulary of size v, each of the h words in the context requires an indicator function of dimension v. So, we propose an algorithm Low Rank Multi-View Learning (LR-MVL), where we work in the k dimensional space to begin with.

The key move in LR-MVL is to project the hv-dimensional $\mathbf{L}$ and $\mathbf{R}$ matrices down to a k dimensional state space before performing the first CCA. This is where it differs from TSCCA. Thus, all eigenvector computations are done in a space that is v/k times smaller than the original space. Since a typical vocabulary contains at least $100,000$ words, and we use state spaces of order $k \approx 100$ dimensions, this gives a 1,000-fold reduction in the size of calculations that are needed.

LR-MVL iteratively updates the real-valued state of a token $\mathbf{Z_t}$, till convergence. Since, the state is always real-valued, this also allows us to replace the projected left and right contexts with exponential smooths (weighted average of the previous (or next) token's state i.e. $\mathbf{Z_{t-1}}$ (or $\mathbf{Z_{t+1}}$) and previous (or next) token's smoothed state i.e. $\mathbf{S_{t-1}}$ (or $\mathbf{S_{t+1}}$).), of them at a few different time scales. One could use a mixture of both very short and very long contexts which capture short and long range dependencies as required by NLP problems as NER, Chunking, WSD etc. Since exponential smooths are linear, we preserve the linearity of our method.

We now describe the LR-MVL algorithms.

4.2.1 The LR-MVL Algorithms

Based on our theory (described in next subsection), various algorithms are possible for LR-MVL. We provide two algorithms, Algorithms 2, 3 (without and with exponential smooths).

Algorithm 2 LR-MVL Algorithm - Learning from Large amounts of Unlabeled Data (no exponential smooths).

1: **Input:** Token sequence $\mathbf{W}_{n\times v}$, state space size k.
2: Initialize the eigenfeature dictionary $\boldsymbol{\phi}_w$ to random values $\mathcal{N}(0,1)$.
3: **repeat**
4: Project the left and right context matrices $\mathbf{L}_{n\times vh}$ and $\mathbf{R}_{n\times vh}$ down to 'k' dimensions and compute CCA between them. $[\boldsymbol{\phi}_l, \boldsymbol{\phi}_r]$=CCA($\mathbf{L}\boldsymbol{\phi}^{\mathbf{h}}_{\mathbf{w}}$, $\mathbf{R}\boldsymbol{\phi}^{\mathbf{h}}_{\mathbf{w}}$). //$\boldsymbol{\phi}^h_w$ is the stacked version of $\boldsymbol{\phi}_w$ matrix as many times as the context length 'h.'
5: Normalize $\boldsymbol{\phi}^{(k)}_l$ and $\boldsymbol{\phi}^{(k)}_r$. //Divide each row by the maximum absolute value in that row (Scales between -1 and +1).
6: Compute a second CCA between the estimated state and the word itself $[\boldsymbol{\phi}_w, \boldsymbol{\phi}_c]$=CCA($\mathbf{W}$, $[\mathbf{L}\boldsymbol{\phi}^{\mathbf{h}}_{\mathbf{w}}\boldsymbol{\phi}^{(k)}_l$, $\mathbf{R}\boldsymbol{\phi}^{\mathbf{h}}_{\mathbf{w}}\boldsymbol{\phi}^{(k)}_r]$).
7: Compute the change in $\boldsymbol{\phi}_w$ from the previous iteration
8: **until** $|\Delta\boldsymbol{\phi}^h_w| < \epsilon$
9: **Output:** $\boldsymbol{\phi}_l$, $\boldsymbol{\phi}_r$, $\boldsymbol{\phi}_w$.

A few iterations (~ 10) of the above algorithms are sufficient to converge to the solution[6].

4.2.2 Theoretical Properties of LR-MVL

We now present the theory behind the LR-MVL algorithms; particularly we show that the reduced rank matrix $\boldsymbol{\phi}_w$ allows a significant data reduction while preserving the information in our data and the estimated state does the best possible job of capturing any label information that can be inferred by a linear model.

The key difference from TSCCA is that we can initialize the state of each word randomly and work in a low (k) dimensional space from the beginning, iteratively refine the state till convergence and still we can recover the eigenword dictionary $\boldsymbol{\phi}_w$.

6. Though the optimization problem and our iterative procedure are non-convex, empirically we did not face any issues with convergence.

Algorithm 3 LR-MVL Algorithm - Learning from Large amounts of Unlabeled Data (with exponential smooths).

1: **Input:** Token sequence $\mathbf{W}_{n\times v}$, state space size k, smoothing rates α^j
2: Initialize the eigenfeature dictionary $\boldsymbol{\phi}_w$ to random values $\mathcal{N}(0,1)$.
3: **repeat**
4: Set the state Z_t $(1 < t \le n)$ of each token w_t to the eigenword vector of the corresponding word.
$Z_t = (\boldsymbol{\phi}_w : w = w_t)$
5: Smooth the state estimates before and after each token to get a pair of views for each smoothing rate α^j.
$S_t^{(l,j)} = (1-\alpha^j)S_{t-1}^{(l,j)} + \alpha^j Z_{t-1}$ // left view $\mathbf{L}$
$S_t^{(r,j)} = (1-\alpha^j)S_{t+1}^{(r,j)} + \alpha^j Z_{t+1}$ // right view $\mathbf{R}$.
where the t^{th} rows of $\mathbf{L}$ and $\mathbf{R}$ are, respectively, concatenations of the smooths $S_t^{(l,j)}$ and $S_t^{(r,j)}$ for each of the $\alpha^{(j)}$s.
6: Find the left and right canonical correlates, which are the eigenvectors $\boldsymbol{\phi}_l$ and $\boldsymbol{\phi}_r$ of
$(\mathbf{L}^\top\mathbf{L})^{-\mathbf{1}}\mathbf{L}^\top\mathbf{R}(\mathbf{R}^\top\mathbf{R})^{-\mathbf{1}}\mathbf{R}^\top\mathbf{L}\boldsymbol{\phi}_l = \lambda\boldsymbol{\phi}_l$.
$(\mathbf{R}^\top\mathbf{R})^{-\mathbf{1}}\mathbf{R}^\top\mathbf{L}(\mathbf{L}^\top\mathbf{L})^{-\mathbf{1}}\mathbf{L}^\top\mathbf{R}\boldsymbol{\phi}_r = \lambda\boldsymbol{\phi}_r$.
7: Project the left and right views on to the space spanned by the top k left and right CCAs respectively
$\mathbf{X_l} = \boldsymbol{L}\boldsymbol{\phi}_l^{(k/2)}$ and $\mathbf{X_r} = \boldsymbol{R}\boldsymbol{\phi}_r^{(k/2)}$
where $\boldsymbol{\phi}_l^{(k)}$, $\boldsymbol{\phi}_r^{(k)}$ are matrices composed of the singular vectors of $\boldsymbol{\phi}_l$, $\boldsymbol{\phi}_r$ with the k largest magnitude singular values. Estimate the state for each word w_t as the union of the left and right estimates: $\mathbf{Z} = [\mathbf{X_l},\ \mathbf{X_r}]$
8: Compute a second CCA between the estimated state and the word itself $[\boldsymbol{\phi}_w, \boldsymbol{\phi}_z]$=CCA($\mathbf{W}$, $\mathbf{Z}$).
9: Normalize $\boldsymbol{\phi}_w$. //Divide each row by the maximum absolute value in that row (Scales between -1 and +1).
10: Compute the change in $\boldsymbol{\phi}_w$ from the previous iteration.
11: **until** $|\Delta\boldsymbol{\phi}_w| < \epsilon$
12: **Output:** $\boldsymbol{\phi}_l^k$, $\boldsymbol{\phi}_r^k$, $\boldsymbol{\phi}_w$.

As earlier, let $\mathbf{L}$ be an $n\times hv$ matrix giving the words in the left context of each of the n tokens, where the context is of length h, $\mathbf{R}$ be the corresponding $n\times hv$ matrix for the right context, and $\mathbf{W}$ be an $n\times v$ matrix of indicator functions for the words themselves. Note that $\mathbf{L}$, $\mathbf{R}$ and $\mathbf{W}$ are the observed instantiations of the corresponding multivariate random variables l, r and w.

The theory of LR-MVL hinges on four assumptions which are described in detail in the appendix. Basically, they entail that there exists a k dimensional linear hidden state for l, r and w and that they come from a HMM with rank k observation and transition matrices. It's further assumed that the pairwise expected correlations between l, r and w, also have rank k.

Lemma 3 *Define $\boldsymbol{\phi}_w$ as the left singular vectors:*

$$\boldsymbol{\phi}_w \equiv CCA(w, [l \quad r])_{left}.$$

where CCA(z, w) is defined as in Equation 1 but using population covariance matrices i.e. $\mathbf{C}_{zw} = \mathbb{E}(z^\top w)$, $\mathbf{C}_{zz} = \mathbb{E}(z^\top z)$ *and* $\mathbf{C}_{ww} = \mathbb{E}(w^\top w)$.
Under assumptions 2, 3 and 1A(in appendix) such that if $(\boldsymbol{\phi}_l, \boldsymbol{\phi}_r) \equiv CCA(l, r)$ *then*

$$\boldsymbol{\phi}_w = CCA(w, [l\phi_l \quad r\phi_r])_{left}.$$

Please see Appendix A for the proof.

Lemma 3 shows that instead of finding the CCA between the full context and the words, we can take the CCA between the Left and Right contexts, estimate a k dimensional state from them, and take the CCA of that state with the words and get the same result. Lemma 3 is similar to Theorem 2, except that it does not provide ratios of the estimated state sizes.

Let $\boldsymbol{\phi}_w^h$ denote a matrix formed by stacking h copies of $\boldsymbol{\phi}_w$ on top of each other. Right multiplying l or r by $\boldsymbol{\phi}_w^h$ projects each of the words in that context into the k-dimensional reduced rank space.

The following theorem addresses the core of the LR-MVL algorithm, showing that there is an $\boldsymbol{\phi}_w$ which gives the desired dimensionality reduction. Specifically, it shows that the previous lemma also holds in the reduced rank space.

Theorem 4 *Under assumptions 1, 1A and 2 (in appendix) there exists a unique matrix* $\boldsymbol{\phi}_w$ *such that if*

$$(\boldsymbol{\phi}_l^h, \boldsymbol{\phi}_r^h) \equiv CCA(l\boldsymbol{\phi}_w^h, r\boldsymbol{\phi}_w^h),$$

then

$$\boldsymbol{\phi}_w = CCA(w, [l\boldsymbol{\phi}_w^h\boldsymbol{\phi}_l^h \quad r\boldsymbol{\phi}_w^h\boldsymbol{\phi}_r^h])_{left},$$

where $\boldsymbol{\phi}_w^h$ *is the stacked form of* $\boldsymbol{\phi}_w$.

Please see Appendix A for the proof[7].

Because of the Zipfian distribution of words, many words are rare or even unique. So, just as in the case of TSCCA, CCA between the rare words and context will not be informative, whereas finding the CCA between the projections of left and right contexts gives a good state vector estimate even for unique words. One can then fruitfully find the CCA between the contexts and the estimated state vector for their associated words.

5. Generating Context Specific Embeddings

Once we have estimated the CCA model using any of our algorithms (i.e. OSCCA, TSCCA, LR-MVL), it can be used to generate context specific embeddings for the tokens from training, development and test sets (as described in Algorithm 4). These embeddings could be

7. It is worth noting that our matrix $\boldsymbol{\phi}_w$ corresponds to the matrix $\hat{\mathbf{U}}$ used by (Hsu et al., 2009; Siddiqi et al., 2010). They showed that $\mathbf{U}$ is sufficient to compute the probability of a sequence of words generated by an HMM, our $\phi_{\mathbf{w}}$ provides a more statistically efficient estimate of $\mathbf{U}$ than their $\hat{\mathbf{U}}$, and hence can also be used to estimate the sequence probabilities.

further supplemented with other baseline features and used in a supervised learner to predict the label of the token.

Algorithm 4 Inducing Context Specific Embeddings for Train/Dev/Test Data

---

1: **Input:** Model (ϕ_l^k, ϕ_r^k, ϕ_w) output from above algorithm and Token sequences $\mathbf{W^{train}}$, ($\mathbf{W^{dev}}$, $\mathbf{W^{test}}$)

2: Project the left and right views $\mathbf{L}$ and $\mathbf{R}$ onto the space spanned by the top k left and right CCAs respectively. If algorithm is Algorithm 3, then, smooth $\mathbf{L}$ and $\mathbf{R}$ first.
$\mathbf{X_l} = \boldsymbol{L}\boldsymbol{\phi}_l^k$ and $\mathbf{X_r} = \boldsymbol{R}\boldsymbol{\phi}_r^k$
and the words onto the eigenfeature dictionary $\mathbf{X_w} = \boldsymbol{W}^{train}\boldsymbol{\phi}_w$

3: Form the final embedding matrix $\mathbf{X_{train:embed}}$ by concatenating these three estimates of state
$\mathbf{X_{train:embed}} = [\mathbf{X_l}\ \ \mathbf{X_w}\ \ \mathbf{X_r}]$

4: **Output:** The embedding matrices $\mathbf{X_{train:embed}}$, ($\mathbf{X_{dev:embed}}$, $\mathbf{X_{test:embed}}$) with context-specific representations for the tokens.

Note that we can get context "oblivious" embeddings i.e. one embedding per word type, just by using the eigenfeature dictionary $\boldsymbol{\phi}_w$. Later in the experiments section we show that this approach of inducing context specific embeddings gives results which are similar to a simpler alternative of just using the context "oblivious" embeddings but augmenting them with the embeddings of the words in a window of 2 around the current word before using them in a classifier.

6. Efficient Estimation

As mentioned earlier, CCA can be done by taking the singular value decomposition of a matrix. For small matrices, this can be done using standard functions in e.g. MATLAB, but for very large matrices (e.g. for vocabularies of tens or hundreds of thousands of words), it is important to take advantage of the recent advances in SVD algorithms. For our experiments we use the method of (Halko et al., 2011), which uses random projections to compute SVD of large matrices.

The key idea is to find a lower dimensional basis for $\mathbf{A}$, and to then compute the singular vectors in that lower dimensional basis. The initial basis is generated randomly, and taken to be slightly larger than the eventual basis. If $\mathbf{A}$ is $v \times hv$, and we seek a state of dimension k, we start with a $hv \times (k + l)$ matrix $\mathbf{\Omega}$ of random numbers, where l is number of "extra" basis vectors between 0 and k. We then project $\mathbf{A}$ onto this matrix and take the SVD decomposition of the resulting matrix ($\mathbf{A} \approx \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^{\top}$).

Since $\mathbf{A\Omega}$ is $v \times (k + l)$, this is much cheaper than working on the original matrix $\mathbf{A}$. We keep the largest k components of $\mathbf{U}$ and of $\mathbf{V}$, which form a left and a right basis for $\mathbf{A}$ respectively.

This procedure is repeated for a few (~ 5) iterations. The algorithm is summarized in Algorithm 5. The runtime of the procedure for projecting a matrix of size $m \times p$ down to a size $m \times k$ where $p \gg k$ is O(mpk) floating point operations, which in our case becomes O(v^2hk).

(Halko et al., 2011) prove a number of nice properties of the above algorithm. In particular, they guarantee that the algorithm, even without the extra iterations in steps 3 and 6 produces an approximation whose error is bounded by a small polynomial factor times the

Algorithm 5 Randomized singular value decomposition

Input: Matrix $\mathbf{A}$ of size $v \times hv$, the desired hidden state dimension k, and the number of "extra" singular vectors, l
Generate a $hv \times (k+l)$ random matrix $\mathbf{\Omega}$
for i =1:5 **do**
  $\mathbf{M} = \mathbf{A}\mathbf{\Omega}$.
  $[\mathbf{Q}, \mathbf{R}]$=QR($\mathbf{M}$) //Find $v \times (k+l)$ orthogonal matrix $\mathbf{Q}$.
  $\mathbf{B} = \mathbf{Q}^\top \mathbf{A}$
  $\mathbf{\Omega} = \mathbf{B}$
end for
Find the SVD of $\mathbf{B}$. $[\hat{\mathbf{U}}, \hat{\mathbf{\Lambda}}, \hat{\mathbf{V}}^\top]$ =SVD($\mathbf{B}$), and keep the k components of $\hat{\mathbf{U}}$ with the largest singular values.
$\tilde{\mathbf{A}} = \mathbf{Q}\hat{\mathbf{U}}$. //Compute the rank-k projection.
Output: The rank-k approximation $\tilde{\mathbf{A}}$. (Similar procedure can be repeated to get the right singular values and the corresponding projections.)

size of the largest singular value whose singular vectors are *not* part of the approximation, σ_{k+1}. They also show that using a small number of "extra" singular vectors (l) results in a substantial tightening of the bound, and that the extra iterations, which correspond to power iteration, drive the error bound exponentially quickly to one times the largest non-included singular value, σ_{k+1} and also provide better separation between the singular values.

7. Evaluating Eigenwords

In this section we provide qualitative and quantitative evaluation of the various eigenword algorithms.

The state estimates for words capture a wide range of information about them that can be used to predict part of speech, linguistic features, and meaning. Before presenting a more quantitative evaluation of predictive accuracy, we present some qualitative results showing how word states, when projected in appropriate directions usefully characterize the words.

We compare our approach against several state-of-the-art word embeddings:

1. Turian Embeddings (C&W and HLBL) (Turian et al., 2010).

2. SENNA Embeddings (Collobert et al., 2011).

3. word2vec Embeddings (Mikolov et al., 2013a,b).

We also compare against simple PCA/LSA embeddings and other model based approaches wherever applicable.

We downloaded the Turian embeddings (C&W and HLBL), from `http://metaoptimize.com/projects/wordreprs` and use the best 'k' reported in the paper (Turian et al., 2010) i.e. k=200 and 100 respectively. SENNA embeddings were downloaded from `http://ronan.collobert.com/senna/`. word2vec code was downloaded from `https://code.google.com/p/word2vec/`. Since they made the code available we could train them on the exact same corpora, had the exact same context window and vocabulary size as the eigenword embeddings. The PCA baseline used is similar to the one that has recently been proposed by (Lamar et al., 2010) except that here we are interested in supervised accuracy and not the unsupervised accuracy as in that paper.

In the results presented below (qualitative and quantitative), we trained all the algorithms (including eigenwords) on Reuters RCV1 corpus (Rose et al., 2002) for uniformity of comparison[8]. Case was left intact and we did not do any other "cleaning" of data. Tokenization was performed using NLTK tokenizer (Bird and Loper, 2004). RCV1 corpus contains Reuters newswire from Aug '96 to Aug '97 and containing about 215 million tokens after tokenization.

Unless otherwise stated, we consider a fixed window of two words (h=2) on either side of a given word and a vocabulary of 100,000 most frequent words for all the algorithms[9], in order to ensure fairness of comparison.

Eigenword algorithms are robust to the dimensionality of hidden space (k), so we did not tune it and fixed it at 200. For other algorithms, we report results using their best hidden space dimensionality.

Our theory and CCA in general (Bach and Jordan, 2005) rely on normality assumptions[10], however the words follow Zipfian (heavy tailed) distribution. So, we took the square root of the word counts in the context matrices (i.e. $\mathbf{W}^\top\mathbf{C}$) before running OSCCA, TSCCA and LR-MVL(I). This squishes the word distributions and makes them look more normal (Gaussian). This idea is not novel and dates back in statistics to Anscombe Transform (Anscombe, 1948) and has precedents even in word representation learning literature (Turney and Pantel, 2010).

We ran LR-MVL(I) and LR-MVL(II) for 5 iterations and only used one exponential smooth of 0.5 for LR-MVL(II). Table 1 shows the details of all the embeddings used in our experiments.

8. Qualitative Evaluation of OSCCA

To illustrate the sorts of information captured in our state vectors, we present a set of figures constructed by projecting selected small sets of words onto the space spanned by the second and third largest principal components of their eigenword dictionary values, which are simply the left canonical correlates calculated from Equation 2. (The first principle component generally just separates the selected words from other words, and so is less interesting here.)

Figure 3 shows plots for three different sets of words. The left column uses the eigenword dictionary learned using OSCCA (CCA($\mathbf{W}$, $\mathbf{C}$), where $\mathbf{C}$=[$\mathbf{L}$ $\mathbf{R}$] with h=2 on either side) (the other eigenword algorithms gave similar results), while the right column uses the corresponding latent vectors derived using PCA on the same data. In all cases, the 200-

8. word2vec, PCA and Turian (C&W and HLBL) embeddings are all trained on Reuters RCV1, but SENNA embeddings (training code not available) were trained on a larger Wikipedia corpus.
9. Turian (C&W and HLBL), SENNA embeddings had much bigger vocabulary sizes of 268,000 and 130,000, though they also use a window of 2 as context.
10. CCA can be thought of as least squares regression (Please see the proof of Theorem 2 in Appendix A.) and hence has error terms distributed normally.

Embedding	Unlabeled Data Trained	Window size	Vocab. Size	Hidden State Size	Availability
C&W (Turian)	Reuters RCV1 cleaned and lower-cased (See Turian et al. 2010.)	2	268,810	200	Only Embeddings available (No Code).
HLBL (Turian)	Reuters RCV1 cleaned and lower-cased (See Turian et al. 2010.)	2	268,810	100	Only Embeddings available (No Code).
SENNA	Wikipedia (much larger than RCV1) (See Collobert et al. 2011.)	2	130,000	50	Only Embeddings available (No Code).
Word2vec (SK-Continuous Skip-gram) & (CB-Continuous Bag-of-words)	Reuters RCV1 uncleaned and case intact	2	100,000	200	Code available
Eigenwords	Reuters RCV1 uncleaned and case intact	2	100,000	200	Code and Embeddings available

Table 1: Details of various embeddings used in the experiments. Note: Eigenwords and Word2vec provide the most controlled comparison.

dimensional vectors have been projected onto two dimensions (using a second PCA) so that they can be visualized.

The PCA algorithm differs from CCA based (eigenword) algorithms in that it does not whiten the matrices via ($\mathbf{C}_{zz}^{-1/2}$ and $\mathbf{C}_{ww}^{-1/2}$) before performing SVD. In other words, the PCA algorithm just operates on $\mathbf{W}^\top \mathbf{C}$. If one considers a word and its two grams to the left and right as a document, then its equivalent to the Latent Semantic Analysis (LSA) algorithm.

The results for various (handpicked) semantic categories are shown in Figure 3 and 4.

The top row shows a small set of randomly selected nouns and verbs. Note that for CCA, nouns are on the left, while verbs are on the right. Words that are of similar or opposite meaning (e.g. “agree” and “disagree”) are distributionally similar, and hence close. The corresponding plot for PCA shows some structure, but does not give such a clean separation. This is not surprising; predicting the part of speech of words depends on the exact order of the words in their context (as we capture in CCA); a PCA-style bag-of-words can’t capture part of speech well.

The bottom row in Figure 3 shows names of numbers or the numerals representing numbers and years. Numbers that are close to each other in value tend to be close in the

Center Word	OSCCA NN	PCA NN
market	markets, trade, currency, sector, activity.	dollar, economy, government, sector, industry.
company	firm, group, giant, operator, maker.	government, group, dollar, following, firm.
Ltd	Limited, Bhd, Plc, Co, Inc.	Corp, Plc, Inc, name, system.
President	Governor, secretary, Chairman, leader, Director.	Commerce, General, fuel, corn, crude.
Nomura	Daiwa, UBS, HSBC, NatWest, BZW.	Chrysler, Sun, Delta, Bre-X, Renault.
jump	drop, fall, rise, decline, climb.	surge, stakes, slowdown, participation, investing.
rupee	peso, zloty, crown, pound, franc.	crown, CAC-40, FTSE, Nikkei, 30-year.

Table 2: Nearest Neighbors of OSCCA and PCA word embeddings.

plot, thus suggesting that state captures not just classifications, but also more continuous hidden features.

The plots in Figure 4 show a similar trend i.e., eigenword embeddings are able to provide a clear separation between different syntactic/semantic categories and capture a rich set of features characterizing the words, whereas PCA mostly just squishes them together.

Table 2 shows the five nearest neighbors for a few representative words using OSCCA and PCA. As can be seen, the OSCCA based nearest neighbors capture subtle semantic and syntactic cues e.g Japanese investment bank (Nomura) having another Japanese investment bank (Daiwa) as the nearest neighbor, whereas the PCA nearest neighbors are more noisy and capture mostly syntactic aspects of the word.

9. Quantitative Evaluation

This section describes the performance (accuracy and richness of representation) of various eigenword algorithms. We evaluate the quality of the *eigenword dictionary* by using it in a supervised learning setting to predict a wide variety of labels that can be attached to words.

We perform experiments for a variety of NLP tasks including, Word Similarity, Sentiment Classification, Named Entity Recognition (NER), chunking, Google semantic and syntactic analogy tasks and Word Sense Disambiguation (WSD) to demonstrate the richness of the state learned by eigenwords and that they perform comparably or better than other state-of-the-art approaches. For these tasks, we report results using the best eigenwords for compactness, though all the four algorithms gave similar performances.

However, before we proceed to do that, we compare OSCCA against TSCCA, LR-MVL(I) and LR-MVL(II) embeddings on a set of Part of Speech (POS) tagging problems for different languages, looking at how the predictive accuracy scales with corpus size for predictions on a fixed vocabulary. These results use small corpora and demonstrate that TSCCA, LR-MVL(I) and LR-MVL(II) perform better for rarer words.

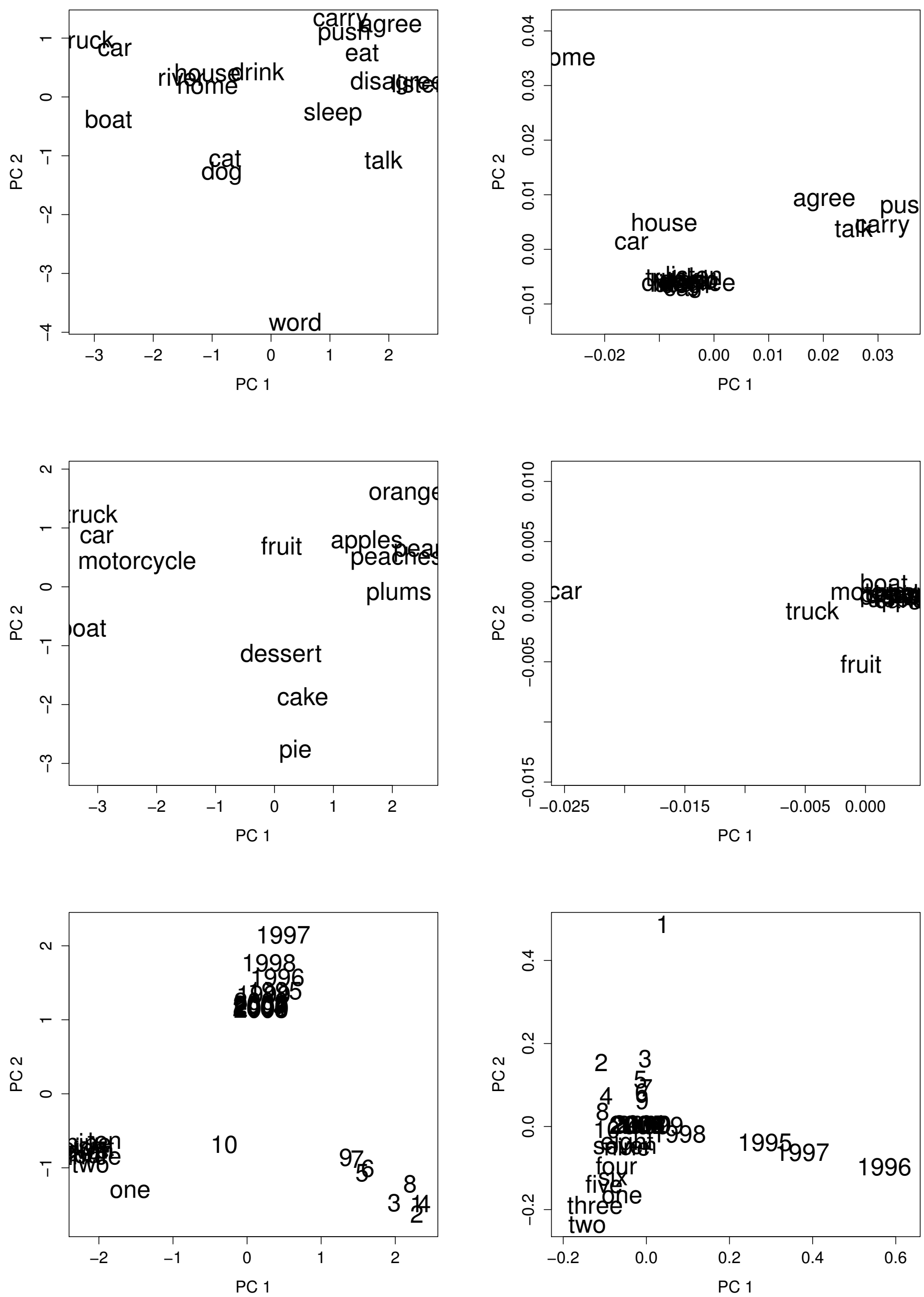

Figure 3: Projections onto two dimension of selected words in different categories using both OSCCA (left) and PCA (Right). Top to bottom: 1). (Nouns vs Verbs): house, home, dog, truck, boat, word, river, cat, car, sleep, eat, push, drink, listen, carry, talk, disagree, agree. 2). (Eateries vs vehicles): apples, pears, plums, oranges, peaches, fruit, cake, pie, dessert, truck, boat, car, motorcycle. 3). (Numerals vs letter numbers vs years): one, two, three, four, five, six, seven, eight, nine, ten, 1, 2,. . ., 10, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009.

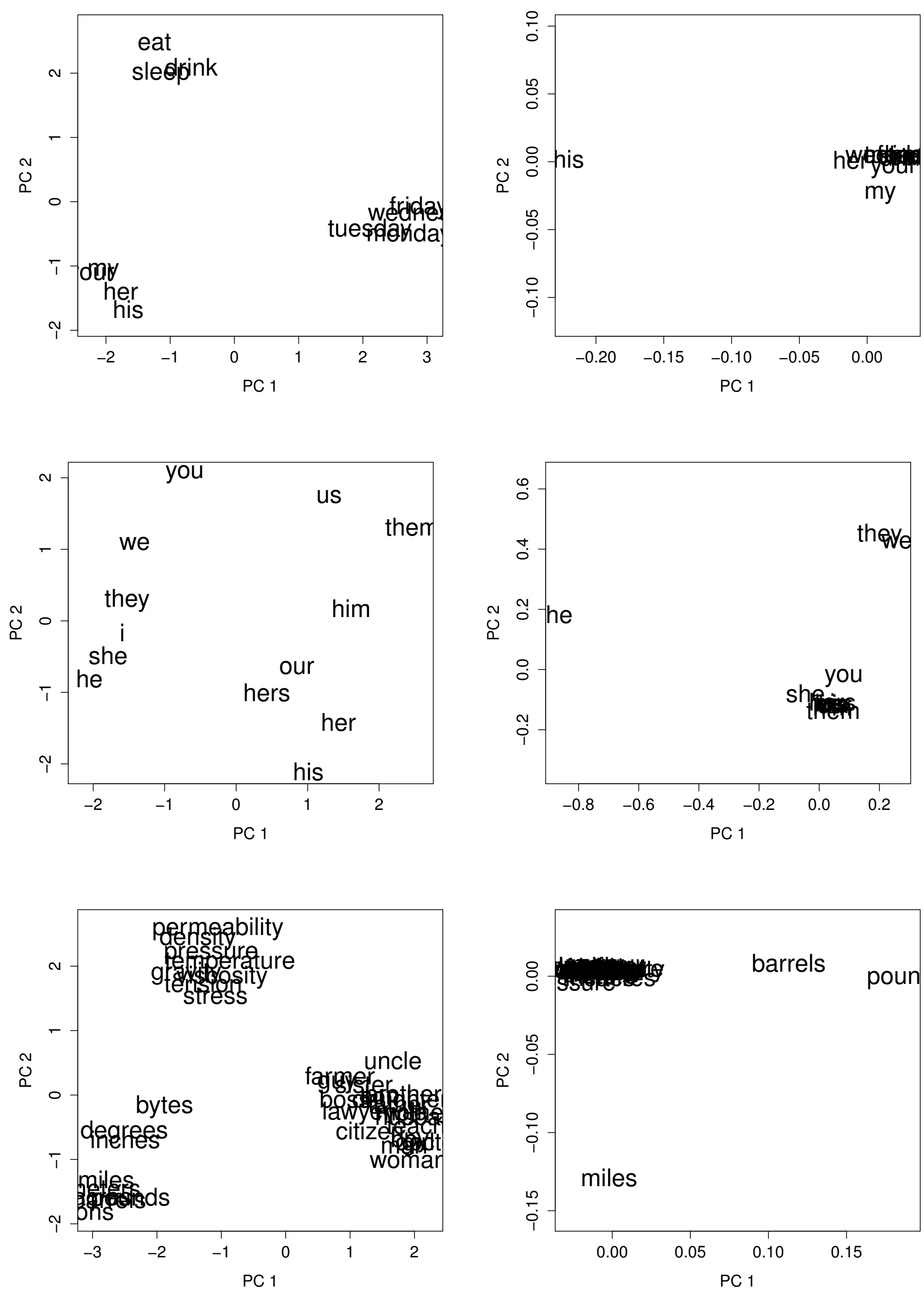

Figure 4: Projections onto two dimension of selected words in different categories using both OSCCA (left) and PCA (Right). Top to bottom: 1). (Weekdays vs verbs vs pronouns): monday, tuesday, wednesday, sunday, friday, eat, drink, sleep, his, her, my, your. 2). (Different kinds of pronouns): i, you, he, she, they, we, us, them, him, her, our, his, hers. 3). (Nouns vs Adjectives vs Units of measurement): man, woman boy, girl, lawyer, doctor, guy, farmer, teacher, citizen, mother, wife, father, son, husband, brother, daughter, sister, boss, uncle, pressure, temperature, permeability, density, stress, viscosity, gravity, tension, miles, pounds, degrees, inches, barrels, tons, acres, meters, bytes.

Language	Number of POS tags	Number of tokens
English	17	100311
Danish	25	100238
Bulgarian	12	100489
Portuguese	22	100367

Table 3: Description of the POS tagging data sets

9.1 Part of Speech (POS) Tagging

In this experiment we compare the performance of various eigenword algorithms on the task of non-disambiguating POS tagging for four languages; i.e., each word type has a single POS tag. Table 2 provides statistics on all the corpora used, namely: the Wall Street Journal portion of the Penn treebank (Marcus et al., 1993) (we consider the 17 tags of (PTB 17) (Smith and Eisner, 2005)), the Bosque subset of the Portuguese Floresta Sinta(c)tica Treebank (Afonso et al., 2002), the Bulgarian BulTreeBank (Simov et al., 2002) (with only the 12 coarse tags), and the Danish Dependency Treebank (DDT) (Kromann, 2003).

Note the corpora range widely in size; English has ~ 1 million tokens whereas Danish only has $\sim 100k$ tokens. To address this data imbalance we kept only the first $\sim 100k$ tokens of the larger corpora so as to perform a uniform evaluation across all corpora.

The goal of this experiment is see to how the eigenword dictionary estimates for the word types (for a fixed vocabulary) improve with increased training data.

Theorem 2 implies that the difference between OSCCA and TSCCA/LR-MVL(I)/LR-MVL(II) should be more pronounced at smaller sample sizes, where the errors are higher and that they should have similar predictive power in the limit of large training data. We therefore evaluate the performance of the methods on varying data sizes ranging from $5k$ to the entire $100k$ tokens.

When varying the unlabeled data from $5k$ to $100k$ we made sure that they had the exact same vocabulary to assure that the performance improvement is not coming from word types not present in the $5k$ tokens but present in the total $100k$. This gives a clear picture of the effect of varying training set size.

To evaluate the predictive accuracy of the descriptors learned using different amounts of unlabeled data, we learn a multi-class logistic regression to predict the POS tag of each type. We trained using 80% of the word types chosen randomly and then tested on the remaining 20% types. This procedure was repeated 10 times. Note that our train and test sets do not contain any of the same word types[11].

The accuracy of using OSCCA, TSCCA, LR-MVL(I), LR-MVL(II) and PCA features in a supervised learner are shown in Figure 5 for the task of POS tagging. As can be seen from the results, eigenword embeddings are significantly better (5% significance level in a paired t-test) than the PCA-based supervised learner. Among the eigenwords, TSCCA, LR-MVL(I) and LR-MVL(II) are significantly better than OSCCA for small amounts of data and, as predicted by theory, the two become comparable in accuracy as the amount of unlabeled data used to learn the CCAs becomes large.

11. As noted, we are doing non-disambiguating POS tagging so that each word type has a single POS tag, so if the same word type occurred in both the training and testing data, a learning algorithm that just memorized the training set would perform reasonably well.

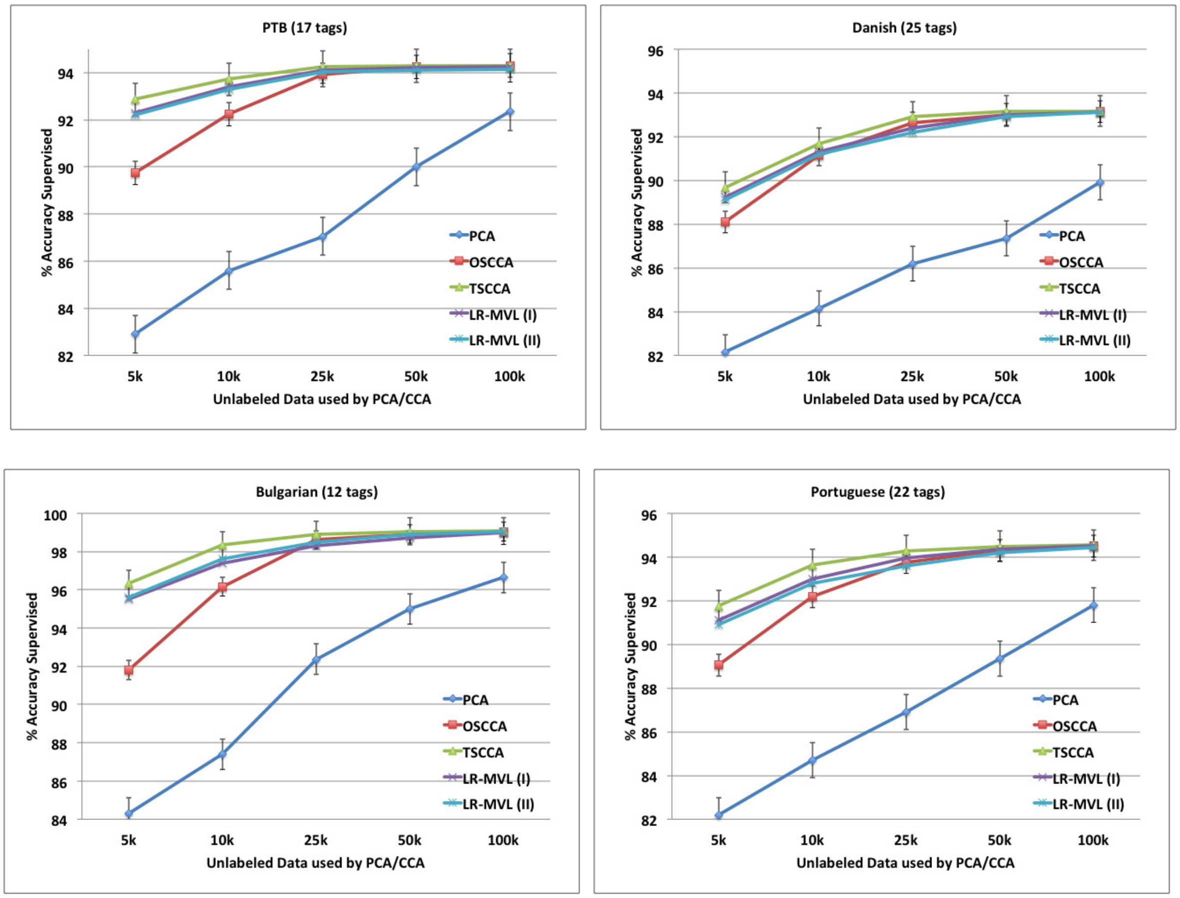

Figure 5: Plots showing accuracy as a function of number of tokens used to train the PCA/eigenwords for various languages. **Note:** The results are averaged over 10 random, 80 : 20 splits of word types.

9.2 Word Similarity Task (WordSim-353)

A standard data set for evaluating vector-space models is the WordSim-353 data set (Finkelstein et al., 2001), which consists of 353 pairs of nouns. Each pair is presented without context and associated with 13 to 16 human judgments on similarity and relatedness on a scale from 0 to 10. For example, (professor, student) received an average score of 6.81, while (professor, cucumber) received an average score of 0.31.

For this task, it is interesting to see how well the cosine similarity between the word embeddings correlates with the human judgment of similarity between the same two words. The results in Table 4 show the Spearman's correlation between the cosine similarity of the respective word embeddings and the human judgments.

As can be seen, eigenwords are statistically significantly (computed using resampled bootstrap) better than all embeddings except SENNA.

Model	$\rho\times$ 100
PCA	30.25
Turian (C&W)	28.08
Turian (HLBL)	35.24
SENNA	44.32
word2vec (SK)	42.73
word2vec (CB)	42.97
eigenwords (OSCCA)	43.00
eigenwords (TSCCA)	**44.85**
eigenwords (LR-MVL(I))	43.83
eigenwords (LR-MVL(II))	37.92

Table 4: Table showing the Spearman correlation between the word embeddings based similarity and human judgment based similarity. Note that the numbers for word2vec are different from the ones reported elsewhere, which is due to the fact that we considered a 100,000 vocabulary and a context window of 2 just like eigenwords, in order to make a fair comparison.

9.3 Sentiment Classification

It is often useful to group words into semantic classes such as colors or numbers, professionals or disciplines, happy or sad words, words of encouragement or discouragement, and, of course, words indicating positive or negative sentiment. Substantial effort has gone into creating hand-curated words that can be used to capture a variety of opinions about different products, papers, or people. To pick one example, (Teufel, 2010) contains dozens of carefully constructed lists of words that she uses to categorize what authors say about other scientific papers. Her categories include "problem nouns" (caveat, challenge, complication, contradiction,...), "comparison nouns" (accuracy, baseline, comparison, evaluation,...), "work nouns" (account, analysis, approach,...) as well as more standard sets of positive, negative, and comparative adjectives.

Psychologists, in particular, have created many such hand curated lists of words, such as the widely used LIWC collection (Pennebaker et al., 2001), which has a heterogeneous set of word lists ranging from "positive emotion" to "pronouns," "swear words" and "body parts." In the example below, we use words from a more homogeneous psychology collection, a set of five dimensions that have been identified in positive psychology under the acronym PERMA (Seligman, 2011):

- *Positive emotion* (aglow, awesome, bliss, ...),
- *Engagement* (absorbed, attentive, busy, ...),
- *Relationships* (admiring, agreeable, ...),
- *Meaning* (aspire, belong, ...)
- *Achievement* (accomplish, achieve, attain, ...).

Word sets	Number of observations	
	Class I	Class II
Positive emotion or not	81	162
Meaningful life or not	246	46
Achievement or not	159	70
Engagement or not	208	93
Relationship or not	236	204

Table 5: Description of the data sets used. All the data was collected from the PERMA lexicon.

For each of these five categories, we have both positive words – ones that connote, for example, *achievement*, and negative words, for example, *un-achievement* (amateurish, blundering, bungling, ...). We would hope (and we show below that this is in fact true), that we can use eigenwords not only to distinguish between different PERMA categories, but also to address the harder task of distinguishing between positive and negative terms in the same category. (The latter task is harder because words that are opposites, such as "large" and "small," often are distributionally similar.)

The description of the PERMA data sets is given in Table 5 and Table 6 shows results for the five PERMA categories. As earlier, we used logistic regression for the supervised binary classification.

As can be seen from the plots, the eigenwords perform significantly (5% significance level in a paired t-test) better than all other embeddings in 3/5 cases and for the remaining 2 cases they perform significantly better than all embeddings except word2vec.

9.4 Named Entity Recognition (NER) & Chunking

In this section we present the experimental results of eigenwords on Named Entity Recognition (NER) and chunking. For the previous evaluation tasks we were performing classification of individual words in isolation, however NER and chunking tasks involve assigning tasks to running text. This allows us to induce context specific embeddings i.e. a different embedding for a word based on its context.

9.4.1 Datasets and Experimental Setup

For the NER experiments we used the data from CoNLL 2003 shared task and for chunking experiments we used the CoNLL 2000 shared task data[12] with standard training, development and testing set splits. The CoNLL '03 and the CoNLL '00 data sets had $\sim 204K/51K/46K$ and $\sim 212K/-/47K$ tokens respectively for Train/Dev./Test sets.

Named Entity Recognition (NER): We use the same set of baseline features as used by (Zhang and Johnson, 2003; Turian et al., 2010) in their experiments. The detailed list of features is as below:

- Current Word w_i; Its type information: all-capitalized, is-capitalized, all-digits and so on; Prefixes and suffixes of w_i

12. More details about the data and competition are available at `http://www.cnts.ua.ac.be/conll2003/ner/` and `http://www.cnts.ua.ac.be/conll2000/chunking/`.

	eigenwords (OSCCA) ($\mu \pm \sigma$)	eigenwords (TSCCA) ($\mu \pm \sigma$)	eigenwords (LR-MVL(I)) ($\mu \pm \sigma$)	eigenwords (LR-MVL(II)) ($\mu \pm \sigma$)	PCA ($\mu \pm \sigma$)	Turian (C&W) ($\mu \pm \sigma$)	Turian (HLBL) ($\mu \pm \sigma$)	SENNA ($\mu \pm \sigma$)	word2vec (SK) ($\mu \pm \sigma$)	word2vec (CB) ($\mu \pm \sigma$)
Positive	25.8± 6.9	24.5± 6.3	26.4± 7.0	29.9± 6.5	33.1 ± 5.8	32.6 ± 6.3	30.0 ± 6.4	29.9 ± 5.1	24.5± 8.3	27.6 ± 7.0
Engagement	18.8 ± 5.0	16.1± 4.4	17.4 ± 4.5	19.6 ± 4.5	29.6 ± 5.2	25.9 ± 5.1	23.2 ± 5.1	20.9 ±5.1	17.2 ± 5.0	20.4 ± 5.3
Relationship	16.3 ± 3.9	**12.2± 3.4**	15.6 ± 4.1	15.9 ± 3.8	46.6 ± 5.4	36.1 ± 5.0	28.3 ± 4.4	18.9 ± 3.4	14.9 ± 4.0	15.0 ± 3.9
Meaningful	10.9 ± 3.9	**8.9± 3.7**	9.5 ± 3.7	9.9 ± 4.0	15.7 ± 3.9	16.1 ± 4.0	15.9 ± 4.0	14.6 ± 3.5	11.1 ± 4.1	14.2 ± 4.5
Achievement	15.7 ± 5.4	**14.6± 5.3**	17.5 ± 5.7	19.0 ± 6.0	30.4 ± 6.0	29.2 ± 6.2	23.0 ± 5.7	20.4 ± 4.9	23.2 ±7.7	27.6 ± 6.8

Table 6: Binary Classification % test errors ($\sum_{i=1}^{n} \frac{\mathbb{I}[y_i \neq \hat{y}_i]}{n}$) averaged over 100 random 80/20 train/test splits for sentiment classification. Bold (3/5 cases) indicates the cases where eigenwords are significantly better (5% level in a paired t-test) compared to all other embeddings. In the remaining 2/5 cases eigenwords are significantly better than all embeddings except word2vec.

- Word tokens in window of 2 around the current word i.e. $d = (w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2})$; and capitalization pattern in the window.

- Previous two predictions y_{i-1} and y_{i-2} and conjunction of d and y_{i-1}

- Embedding features (eigenwords, C&W, HLBL, Brown etc.) in a window of 2 around the current word including the current word (when applicable).

Following (Ratinov and Roth, 2009) we use a regularized averaged perceptron model with the above set of baseline features for the NER task. We also used their BILOU text chunk representation and fast greedy inference, as it was shown to give superior performance.

We also augment the above set of baseline features with gazetteers, as is standard practice in NER experiments. We also benchmark the performance of eigenwords on MUC7 out-of-domain dataset which had 59K words. MUC7 uses a different annotation and has some different Named Entity types that are not present in the CoNLL '03 dataset, so it provides a good test bed for eigenwords. As earlier, we performed the same preprocessing for this dataset as done by (Turian et al., 2010).

Chunking: For our chunking experiments we use a similar base set of features as above:

- Current Word w_i and word tokens in window of 2 around the current word i.e. $d = (w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2})$;

- POS tags t_i in a window of 2 around the current word.

- Word conjunction features $w_i \cap w_{i+1}$, $i \in \{-1, 0\}$ and Tag conjunction features $t_i \cap t_{i+1}$, $i \in \{-2, -1, 0, 1\}$ and $t_i \cap t_{i+1} \cap t_{i+2}$, $i \in \{-2, -1, 0\}$.

- Embedding features in a window of 2 around the current word including the current word (when applicable).

Since the CoNLL '00 chunking data does not have a development set, we randomly sampled 1000 sentences from the training data (8936 sentences) for development. So, we trained our chunking models on 7936 training sentences and evaluated their F1 score on the 1000 development sentences and used a CRF[13] as the supervised classifier. We tuned the magnitude of the ℓ_2 regularization penalty in CRF on the development set. The regularization penalty that gave best performance on development set was 2. Finally, we trained the CRF on the entire ("original") training data i.e. 8936 sentences.

9.4.2 Results

The results for NER and chunking are shown in Tables 7 and 8, respectively, which show that eigenwords perform significantly better than state-of-the-art competing methods on both NER and chunking tasks.

9.5 Cross Lingual Word Sense Disambiguation: SEMEVAL 2013

In cross-lingual word sense disambiguation (WSD) tasks, ambiguous English words are given in context as input, and translations of these words into one or more target languages are

13. `http://www.chokkan.org/software/crfsuite/`

		F1-Score		
Embedding/Model		Dev. Set	Test Set	MUC7
Baseline	No Gazetteers	90.03	84.39	67.48
Brown 1000 clusters		92.32	88.52	78.84
Turian (C&W)		92.46	87.46	75.51
Turian (HLBL)		92.00	88.13	75.25
SENNA		-	88.67	-
word2vec (SK)		92.54	89.40	76.21
word2vec (CB)		92.08	89.20	76.55
eigenwords (OSCCA)		92.94	89.67	79.85
eigenwords (TSCCA)		**93.19**	**89.99**	**80.99**
eigenwords (LR-MVL(I))		92.82	89.85	78.60
eigenwords (LR-MVL(II))		92.73	89.87	78.71
Brown, 1000 clusters	With Gazetteers	93.25	89.41	**82.71**
Turian (C&W)		92.98	88.88	81.44
Turian (HLBL)		92.91	89.35	79.29
SENNA		-	89.59	-
word2vec (SK)		92.99	89.69	79.55
word2vec (CB)		92.93	89.89	79.94
eigenwords (OSCCA)		93.21	90.28	81.59
eigenwords (TSCCA)		**93.96**	**90.59**	82.42
eigenwords (LR-MVL(I))		93.50	90.33	81.15
eigenwords (LR-MVL(II))		93.49	90.10	80.34

Table 7: NER Results. **Note:** F1-score= Harmonic Mean of Precision and Recall. Note that the numbers reported for eigenwords here are different than those in (Dhillon et al., 2011) as we use a different vocabulary size and different dimensionality than there.

Embedding/Model	Test Set F1-Score
Baseline	93.79
Brown 3200 Clusters	94.11
Turian (HLBL)	94.00
Turian (C&W)	94.10
SENNA	93.94
word2vec (SK)	94.02
word2vec (CB)	94.16
eigenwords (OSCCA)	94.02
eigenwords (TSCCA)	**94.23**
eigenwords (LR-MVL(I))	93.97
eigenwords (LR-MVL(II))	94.13

Table 8: Chunking Results. Note that the numbers reported for eigenwords here are different than those in (Dhillon et al., 2011) as we use a different vocabulary size and different dimensionality than there.

produced as output. This can be seen in contrast with more traditional monolingual WSD tasks, in which word senses are instead chosen from a pre-determined sense inventory such as WordNet (Fellbaum, 1998). By framing the problem in a multilingual setting, several important issues are addressed at once. First, by using foreign words rather than human-defined sense labels to resolve ambiguities, WSD systems can more directly be integrated into machine translation and multilingual information retrieval systems, two major areas of application. Moreover, such systems are generalizable to any languages for which sufficient parallel data exists, and do not require the manual construction of sense inventories or sense-tagged corpora for training.

9.5.1 Task Description

We focus on the SemEval 2013 cross-lingual WSD task (Lefever and Hoste, 2013), for which 20 English nouns were chosen for disambiguation. This was framed as an unsupervised task, in which the only provided training data was a sentence-aligned subset of the Europarl parallel corpus (Koehn, 2005). Six languages were included: the source language, English, and the five target languages, namely Spanish, Dutch, German, Italian, and French.

To evaluate a system's output, its answers were compared against the gold standard translations, and corresponding precision and recall scores were computed.

Two evaluation schemes were used in this Semeval task: a Best evaluation metric and an Out-of-Five evaluation metric. For the Best metric, systems could propose multiple sense labels, but the resulting scores were divided by the number of guesses. For the Out-of-Five metric, systems could propose up to five translations without penalty. Further details about this task's evaluation metric can be found in Section 4.1 of Lefever and Hoste (2013).

9.5.2 System Description

Our baseline system was an adaptation of the layer one (L1) classifier described in Section 2 of Rudnick et al. (2013), which was one of the top-scoring systems in the SemEval 2013 cross-lingual WSD task. This system used a maximum entropy model trained on monolingual features from the English source text, incorporating words, lemmas, parts of speech, etc. within a small window of the ambiguous word being classified (Please see Figure 1 of Rudnick et al. (2013) for a detailed list of features). Training instances were extracted programmatically from the provided Europarl subcorpus, using the code made publicly available on the group's GitHub repository[14].

The MEGA Model Optimization Package (MegaM) (Daumé III, 2004) and its NLTK interface (Bird et al., 2009) were used for training the models and producing output for the test sentences.

Using the L1 classifier as a starting point, we began by making two minor modifications to make the system more amenable to further changes. First, regularization was introduced in the form of a Gaussian prior by setting the `sigma` parameter in NLTK's MegaM interface to a nonzero value. Second, "always-on" features were enabled, allowing the classifier to explicitly model the prior probabilities of each output label. Building on this system, we then introduced a variety of embeddings to accompany the existing lexical features. Each

14. https://github.com/hltdi/semeval2013

Best	Spanish	Dutch	German	Italian	French
Most-Frequent Baseline	23.23	20.66	17.43	20.21	25.74
Original L1 System	28.67	21.37	20.64	23.34	27.75
C&W	29.76	**25.17**	22.47	23.59	30.20
HLBL	28.34	24.60	22.35	23.13	29.54
SENNA	**30.78**	24.06	22.39	**25.28**	30.13
Word2Vec (CB)	29.59	25.07	22.73	23.34	30.23
Word2Vec (SK)	29.34	25.04	22.49	23.64	30.09
eigenwords (OSCCA)	30.10	24.58	22.79	24.53	30.37
eigenwords (TSCCA)	30.76	24.56	22.68	24.61	**30.55**
eigenwords (LR-MVL(I))	30.36	24.51	22.92	24.17	30.30
eigenwords (LR-MVL(II))	30.72	24.83	**22.97**	24.85	30.39

Table 9: Best metric F-scores averaged over the twenty English test words.

class of features was included independently of the others in a separate experiment to allow for a direct comparison of the results.

9.5.3 Results

Our experiments were performed using the trial and test data sets from the SemEval 2010 competition, which were released as the trial data for the SemEval 2013 competition. Since the same ambiguous English nouns were tested in both competitions, few changes to the training process were required. The SemEval 2010 trial data was used to select appropriate regularization parameters, and the SemEval 2010 test data was used for the final evaluations.

We used the most frequent translation of an ambiguous word in the training corpus to obtain a baseline score for the Best evaluation metric, and the five most frequent translations to obtain a baseline score for the Out-of-Five evaluation metric. These scores are presented alongside the results of the original L1 classifier and its extensions in Tables 9 and 10. All reported scores are macro averages of the F-scores for the twenty test words from the SemEval 2010 test data. The best score in each category is bolded for emphasis.

We observe that in all cases, the top-scoring system includes some form of vector word embeddings, indicating that these features indeed provide useful information beyond the lexical features from which they are derived. Moreover, the systems using eigenword embeddings outperform the other systems in a majority of cases for both the Best and Out-of-Five evaluation metrics.

9.5.4 Context Specific Embeddings?

The embeddings that we used above for the tasks of NER, Chunking and cross-lingual WSD were the context "oblivious" embeddings i.e. we just used the $\phi_{\mathbf{w}}$ matrix. As described in Section 5 one could induce context specific embeddings also, which help in disambiguating polysemous words. However it turns out that for the tasks of NER, Chunking and WSD they did not give any additional improvement in accuracy. This is due to the fact that in addition to the embedding of the current word we also use the embeddings of words in a window of 2 around the current word as features. They serve as a proxy for the context

Out-of-Five	Spanish	Dutch	German	Italian	French
Most-Frequent Baseline	53.07	43.59	38.86	42.63	51.36
Original L1 System	60.93	46.12	43.40	51.89	57.91
C&W	62.07	48.81	45.06	55.42	63.21
HLBL	61.11	47.25	44.51	55.16	61.19
SENNA	62.88	49.15	45.22	55.92	62.28
Word2Vec (CB)	62.32	48.74	45.51	56.04	62.64
Word2Vec (SK)	61.97	48.35	45.42	56.04	62.55
eigenwords (OSCCA)	62.46	**49.85**	46.34	56.36	62.98
eigenwords (TSCCA)	62.99	49.53	46.60	55.91	63.37
eigenwords (LR-MVL(I))	62.81	49.63	**47.03**	**56.40**	63.12
eigenwords (LR-MVL(II))	**63.05**	49.58	46.86	56.23	**63.51**

Table 10: Out-of-Five metric F-scores averaged over the twenty English test words.

specific embeddings and capture similar discriminative context information as the context specific embeddings do. However, if one only uses the embeddings of the current word as features, then context specific embeddings give improved performance compared to the context oblivious embeddings and the improvement is similar to using the context oblivious embeddings and the embeddings of words in a window of 2 around that word as features.

9.6 Google Semantic and Syntactic Relations Task

(Mikolov et al., 2013a,b) present new syntactic and semantic relation data sets composed of analogous word pairs. The syntactic relations dataset contains word pairs that are different syntactic forms of a given word e.g. write : writes :: eat : eats There are nine such different kinds of relations: adjective-adverb, opposites, comparative, superlative, present participle, nation-nationality, past tense, plural nouns and plural verbs

The semantic relations dataset contains pairs of tuples of word relations that follow a common semantic relation e.g. in Athens : Greece :: Canberra : Australia, where the two given pairs of words follow the country-capital relation. There are three other such kinds of relations: country-currency, man-woman, city-in-state and overall 8869 such pairs of words. The task here is to find a word d that best fits the following relationship: a : b :: c : d given a, b and c. They use the vector offset method, which assumes that the words can be represented as vectors in vector space and computes the offset vector: $y_d = e_a - e_b + e_c$ where e_a, e_b and e_c are the vector embeddings for the words a, b and c. Then, the best estimate of d is the word in the entire vocabulary whose embedding has the highest cosine similarity with y_d. Note that this is a hard problem as it is a v class problem, where v is the vocabulary size.

Table 11 shows the performance of various embeddings for semantic and syntactic relation tasks. Here, as earlier, we trained eigenwords on a Reuters RCV1 with a window size of 2, however as can be seen it performed significantly better compared to all the embeddings except word2vec. We conjectured that it could be due to the fact that we were taking too small a context window which mostly captures syntactic information, which was sufficient for the earlier tasks. So, we experimented with a window size of 10 with the hope that a broader context window should be able to capture semantic and topic information. For this configuration, the eigenwords' performance was comparable to word2vec and as we had intuited most of the improvement in performance took place on the semantic relation task [15].

Embedding/Model	Semantic Relation	Syntactic Relation	Total Accuracy
Turian (C&W)	1.41	2.20	1.84
Turian (HLBL)	3.33	13.21	8.80
SENNA	9.33	12.35	10.98
eigenwords (Window size= 2) (Best)	12.41	30.27	22.28
word2vec (Window size= 10) (SK)	33.91	32.81	33.30
word2vec (Window size= 10) (CB)	31.05	**36.21**	**33.90**
eigenwords (Window size= 10) (OSCCA)	34.79	31.01	32.70
eigenwords (Window size= 10) (TSCCA)	6.06	10.19	8.34
eigenwords (Window size= 10) (LR-MVL(I))	**35.43**	32.12	33.60
eigenwords (LR-MVL(II))	5.41	19.20	13.03

Table 11: Accuracies for Semantic, Syntactic Relation Tasks and total accuracies.

9.6.1 Which Eigenword Embeddings to Use?

We proposed four algorithms for learning word embeddings and from a practitioners point of view it is natural to ask: Which embedding do I use for my supervised NLP task? Based on the experiments and our experience we found that OSCCA = TSCCA > LR-MVL (I) > LR-MVL(II). In other words, OSCCA and TSCCA work remarkably well out-of-the-box and are robust to the choice of the hidden state dimensionality (k) or the context size (h). Also, since they are not iterative algorithms, they are faster to run than the LR-MVL algorithms. LR-MVL(I) trails the OSCCA and TSCCA algorithms only slightly (not significantly) in terms of performance and sometimes gave better performance than them e.g. on the Google analogy tasks.

The LR-MVL algorithms are different in spirit than OSCCA and TSCCA as they involve an iterative procedure. Unfortunately, since the algorithms involve a CCA operation, they are non-convex and hence there are no convergence guarantees. It might be possible to borrow some theoretical machinery from the alternating minimization literature (Netrapalli et al., 2013) to get convergence bounds, but it is beyond the scope of this paper and we leave it for future work. That said, empirically we never faced any issues regarding multiple local-optima, convergence or matrix inversions. We repeated the process 20 times and both the LR-MVL algorithms gave similar answers.

We found the LR-MVL(II) algorithm to be the least robust and highly sensitive to the values and amounts of smooths used. Its behavior can be explained by its genesis and our motivation for proposing it. LR-MVL(II)) is based on modeling language data using time-series models (in fact exponential smoothing is an ARIMA(0,1,1) process). So, from a modeling perspective LR-MVL(II) has a mature story but still empirically it performs worse than simpler models like OSCCA and TSCCA. This, itself sheds some light on the task of word embedding learning in that simple models work really well and are hard to

15. Note that here TSCCA's performance is significantly worse than other algorithms. This should not be entirely surprising as the theoretical analysis of TSCCA assumes squared loss and those guarantees need not hold after performing vector arithmetic.

beat. Perhaps, its so because the text data is not fully amenable to exponential smoothing, like financial or economic time series data and too small or too big smooths scramble the signal provided by the Zipfian distributed words. Also, since it performs smoothing on one document at a time and is iterative, it can be significantly slower to run.

10. Conclusion & Future Work

In this paper we made two main contributions. First, we proposed four algorithms for learning word embeddings (eigenwords) which are fast to train, have strong theoretical properties, can induce context specific embeddings and have better sample complexity for rare words. All the algorithms had a Canonical Correlation Analysis (CCA) style eigen-decomposition at their core. We performed a thorough evaluation of *eigenwords* learned using these algorithms, and showed that they were comparable to or better than other state-of-the-art algorithms when used as features in a set of NLP classification tasks. Eigenwords are able to capture nuanced syntactic and semantic information about the words. They also have a clearer theoretical foundation than the competing algorithms, which allows us to bound their error rate in recovering the true hidden state under linearity assumptions.

Second, we showed that linear models help us attain state-of-the-art performance on text applications and there is no need to move to more complex non-linear models, e.g. Deep Learning based models. In addition, spectral learning methods are highly scalable and parallelizable and can incorporate the latest advances in numerical linear algebra as black-box routines.

There are many open avenues for future research building on the above spectral methods.

1. Our word embeddings are based on modeling individual words based on their contexts; it will be interesting to induce embeddings for entire phrases or sentences. There are multiple possibilities here. One could directly model phrases by considering a phrase as a "unit" rather than a word, perhaps taking the context of a word or phrase from connected elements in a dependency or constituency parse tree. Another possibility is to learn embeddings for individual words but then combine them in some manner to get an embedding for a phrase or a sentence; some relevant work on this problem has been done by (Socher et al., 2012, 2013).

2. Closely related is the idea of semantic composition. Recent advances in spectral learning for tree structures e.g. (Dhillon et al., 2012a; Cohen et al., 2012) may be able to be extended to provide scalable principled alternative methods to the recursive neural networks of (Socher et al., 2012, 2013).

3. Also it will be fruitful to study embeddings where the contexts are left and right dependencies of a word rather than the neighboring words in the surface structure of the sentence. This might give more precise embeddings with smaller data sets.

4. It will also be interesting to incorporate more domain knowledge into the learning of eigenwords. For example, one could envision using ontologies like WordNet (Fellbaum, 1998) as priors in an otherwise data-driven embedding learning.

Appendix A.

CCA by SVD. Proof of Equation 1:

Proof Assuming $\mathbf{W}$ is the $n \times v$ word matrix and $\mathbf{C}$ is the $n \times hv$ context matrix where n is the number of tokens in the corpus, h is the context size and v is the vocabulary size. Further $\mathbf{C}_{wc} = \mathbf{W}^\top \mathbf{C}$, $\mathbf{C}_{cc} = \mathbf{C}^\top \mathbf{C}$ and $\mathbf{C}_{ww} = \mathbf{W}^\top \mathbf{W}$. The CCA objective is to find vectors ϕ_w and ϕ_c such that the linear combinations $s_w = \phi_w^\top \mathbf{W}$ and $s_{cc} = \phi_c^\top \mathbf{C}$ are maximally correlated i.e.

$$\max_{\phi_w,\phi_c} \frac{{\phi_w}^\top \mathbf{C}_{wc}\phi_c}{\sqrt{{\phi_w}^\top \mathbf{C}_{ww}\phi_w}\sqrt{{\phi_c}^\top \mathbf{C}_{cc}\phi_c}}.$$

This is equivalent to

$$\max_{\phi_w,\phi_c} {\phi_w}^\top \mathbf{C}_{wc}\phi_c,$$

subject to unit-norm constraints ${\phi_w}^\top \mathbf{C}_{ww}\phi_w = I$ and ${\phi_c}^\top \mathbf{C}_{cc}\phi_c = I$.

Then, performing full SVD on $\mathbf{C}_{ww}$ and $\mathbf{C}_{cc}$, we get

$$\begin{aligned} \mathbf{C}_{ww} &= \mathbf{V}_w \mathbf{\Lambda}_w \mathbf{V}_w^\top, \\ \mathbf{C}_{cc} &= \mathbf{V}_c \mathbf{\Lambda}_c \mathbf{V}_c^\top, \end{aligned}$$

where $\mathbf{V}_w^\top \mathbf{V}_w = \mathbf{I}_{v\times v}$ and $\mathbf{V}_c^\top \mathbf{V}_c = \mathbf{I}_{hv \times hv}$.

Define change of basis as

$$\begin{aligned} u_w &= \mathbf{\Lambda}_w^{-1/2} \mathbf{V}_w^\top \mathbf{W}, \\ u_{cc} &= \mathbf{\Lambda}_c^{-1/2} \mathbf{V}_c^\top \mathbf{C}, \end{aligned}$$

Now, in this new transformed basis:

$\mathbb{E}[u_w^\top u_w] = \mathbf{\Lambda}_w^{-1/2} \mathbf{V}_w^\top \mathbf{W} \mathbf{V}_w^\top \mathbf{\Lambda}_w \mathbf{V}_w \mathbf{V}_w \mathbf{\Lambda}_w^{-1/2} = \mathbf{I}_{v\times v}$ and similarly $\mathbb{E}[u_{cc}^\top u_{cc}] = \mathbf{I}_{hv\times hv}$, as desired.

Transform the coefficients ϕ_w and ϕ_c, so that s_w and s_{cc} can be expressed as linear combination in the new basis:

$$\begin{aligned} s_w &= \phi_w^\top \mathbf{W} = g_{\phi_w}^\top u_w \\ s_{cc} &= \phi_c^\top \mathbf{C} = g_{\phi_c}^\top u_{cc} \end{aligned}$$

where $g_{\phi_w} = \mathbf{\Lambda}_w \mathbf{V}_w \phi_w$ and $g_{\phi_c} = \mathbf{\Lambda}_c \mathbf{V}_c \phi_c$.

So, the CCA optimization problem can be cast as the following maximization criteria

$$\max_{g_{\phi_w}, g_{\phi_c}} g_{\phi_w}^\top \mathbf{D}_{wc} g_{\phi_c}$$

subject to unit-norm constraints $g_{\phi_w}^\top g_{\phi_w} = \mathbf{I}$ and $g_{\phi_c}^\top g_{\phi_c} = \mathbf{I}$, where $\mathbf{D}_{wc} = \mathbf{\Lambda}_w^{-1/2} \mathbf{V}_w^\top \mathbf{C}_{wc} \mathbf{V}_c \Lambda_c^{-1/2}$.

The solution to above is nothing but the SVD of $\mathbf{D}_{wc}$.

Finally, we can construct the original coefficient matrices $\boldsymbol{\phi}_w$ and $\boldsymbol{\phi}_c$ as $\boldsymbol{\phi}_w = \mathbf{V}_w\boldsymbol{\Lambda}_w^{-1/2}\mathbf{G}_{\phi_w}$ and $\boldsymbol{\phi}_c = \mathbf{V}_c\boldsymbol{\Lambda}_c^{-1/2}\mathbf{G}_{\phi_c}$, where $\mathbf{G}_{\phi_w}$ and $\mathbf{G}_{\phi_c}$ are the matrices corresponding to the vectors g_{ϕ_w} and g_{ϕ_c} respectively.

Now, in our case $\mathbf{C}_{ww} = \mathbf{W}^\top\mathbf{W}$ is the diagonal word occurrence matrix with the words counts in the corpus on the diagonal, so $\boldsymbol{\Lambda}_w^{-1/2}$ is nothing but $\mathbf{C}_{ww}^{-1/2}$ and $\mathbf{V}_w = \mathbf{I}$.

The context matrix $\mathbf{C}_{cc} = \mathbf{C}^\top\mathbf{C}$, though is not diagonal but it can be approximated by its diagonal. One could also approximate it as a diagonal matrix plus its first order Taylor's expansion, but it would make the resulting matrix substantially more dense and hence the computations intense. In our experiments we found no improvement in prediction accuracy by adding the first order Taylor's term, so we approximate $\mathbf{C}_{cc}$ just by its diagonal. ■

Proof of Theorem 1:

Proof Without loss of generality, we can assume that $\mathbf{W}$ and $\mathbf{C}$ have been transformed to their canonical correlations coordinate space. So $Var(\mathbf{W})$ is the identity and $Var(\mathbf{C})$ is the identity, and the $Cov(\mathbf{W}, \mathbf{C})$ is a diagonal with non-increasing values ρ_i on the diagonal (namely the correlations / singular values). We can write α and β in this coordinate system. By orthogonality we now have $\beta_i = \rho_i\alpha_i$. So, $\mathbb{E}(Y - \beta\mathbf{W})^2$ is simply $\sum(\alpha_i - \beta_i\rho_i)^2$. Which is $\sum\alpha_i^2(1-\rho_i^2)$. Our estimator will then have $\gamma_i = \beta_i$ for i smaller than k and $\gamma_i = 0$ otherwise. Hence $(\hat{Y} - \beta^\top\mathbf{W})^2 = \sum_{i=k+1}^{\infty}\beta_i^2$.

So if we pick k to include all terms which have $\rho_i \geq \sqrt{\epsilon}$ our error will be less than $\epsilon\sum_{i=k+1}^{\infty}\alpha_i^2 \leq \epsilon$. ■

Proof of Theorem 2:

Proof The key is that CCA can be understood using the same machinery as is used for analyzing linear regression. In this context we want to recover the word of length v given its context which can be expressed in terms of regression. For a more in-depth discussion of how CCA relates to regression, see (Glahn, 1968), for example. Thus, consider the case of predicting a vector $\mathbf{y}$ of length v (the word) from a vector $\mathbf{x}$ (the context, which is of dimension $2hv$ in the one step CCA case and dimension $2k$ in the two step CCA). We consider the linear model

$$y = \mathbf{x}\beta + \epsilon.$$

Note that, we are predicting only one dimension of our v-dimensional vector $\mathbf{y}$ at a time.

We want to understand the variance of our prediction of a word given the context. As is typical in regression, we calculate a standard error for each coefficient in our contexts, $\approx O(\frac{1}{\sqrt{n}})$. In the one step CCA, $\mathbf{X} = [\mathbf{L} \quad \mathbf{R}]$, and running a regression we will get a prediction error on order of $\frac{hv}{n}$, but since we have v such y's we get a total prediction error on the order of $\frac{hv^2}{n}$.

For the two-step case we take $\mathbf{X} = [\mathbf{L}\phi_{\mathbf{L}} \quad \mathbf{R}\phi_{\mathbf{R}}]$. As mentioned earlier, note that now we are working with about 2k predictors instead of 2hv predictors. If we knew the true $\phi_{\mathbf{L}}$ and $\phi_{\mathbf{R}}$, and thus the true subspace covered by our predictors, the regression error would be on the order of $\frac{kv}{n}$ (again, since there are v entries in our vector $\mathbf{y}$). Instead, we have an estimation of $\phi_{\mathbf{L}}$ and $\phi_{\mathbf{R}}$. If these were computed on infinite amounts of data (and hence we would be arbitrarily close to the true subspace)–we would be done. However since they come from a sample, we are using $\widehat{\phi_{\mathbf{L}}}$ and $\widehat{\phi_{\mathbf{R}}}$ which are approximation to the ideal $\phi_{\mathbf{L}}$ and $\phi_{\mathbf{R}}$. So our task is to understand the error introduced by this sample approximation of the true CCA. First, we develop some notation and concepts found in (Stewart, 1990).

Consider two subspaces $\mathcal{V}$ and $\hat{\mathcal{V}}$ and respective matrices containing an orthonormal basis for these subspaces $\mathbf{V}$ and $\hat{\mathbf{V}}$. Let $\gamma_1, \gamma_2, \ldots$ be the singular values of the matrix $\mathbf{V}^\top \hat{\mathbf{V}}$, then define

$$\theta_i = \cos^{-1} \gamma_i,$$

and define the canonical angle matrix $\mathbf{\Theta} = \mathrm{diag}(\theta_{\mathbf{1}}, \ldots, \theta_{\mathbf{k}})$.

These values of $\mathbf{\Theta}$ capture the effect of using estimated singular vectors, $\hat{\mathbf{V}}$ to form an underlying subspace, as compared to the true subspace formed by the true singular vectors $\mathbf{V}$ stemming from infinite data. The largest canonical angle captures the largest angle between any two vectors- one from the perturbed subspace and one from the true subspace. The second largest canonical angle captures the second largest angle between any two vectors given that they are orthogonal to the original two, and so on. In this proof we will only make use of the largest canonical angle to provide a loose upper bound on the error stemming from the imperfect estimation of the true subspace.

Now, consider a matrix $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{E}$ and take the thin singular value decomposition of $\mathbf{A}$ and $\hat{\mathbf{A}}$ (and here we take the liberty of applying diag in a block matrix sense)

$$\mathbf{A} = [\mathbf{U_1 U_2}]\mathrm{diag}(\mathbf{\Lambda_1}, \mathbf{\Lambda_2})[\mathbf{V_1 V_2}]^\top$$
$$\hat{\mathbf{A}} = [\hat{\mathbf{U}}_\mathbf{1} \hat{\mathbf{U}}_\mathbf{2}]\mathrm{diag}(\hat{\mathbf{\Lambda}}_\mathbf{1}, \hat{\mathbf{\Lambda}}_\mathbf{2})[\hat{\mathbf{V}}_\mathbf{1} \hat{\mathbf{V}}_\mathbf{2}]^\top$$

In our case we have that $\lambda_i = 0$ for all $\lambda_i \in \mathbf{\Lambda_2}$.

From (Stewart and Sun, 1990), we have that

$$\max\{||\sin \mathbf{\Theta}||_\mathbf{2}, ||\sin \mathbf{\Psi}||_\mathbf{2}\} \leq \mathbf{c}||\mathbf{E}||_\mathbf{2}, \tag{3}$$

for some constant c where here $\mathbf{\Theta}$ is the matrix of canonical angles formed from the subspaces of $\mathbf{U}$ and $\hat{\mathbf{U}}$, and $\mathbf{\Psi}$ is the matrix of canonical angles formed between the subspaces of $\mathbf{V}$ and $\hat{\mathbf{V}}$. Note that since $\mathbf{\Theta}$ and $\mathbf{\Psi}$ are diagonal matrices the induced norms $||\cdot||_2$ recover the largest canonical angle of each subspace, and hence we can simultaneously derive an upper bound for the largest canonical angle of either subspace.

We have now developed the machinery we need to analyze the two step CCA.

Without loss of generality, assume that $\mathbf{L}^\top\mathbf{L} = \mathbf{R}^\top\mathbf{R} = \mathbf{I}$ (Even if it is not, we can always rotate $\mathbf{L}$ and $\mathbf{R}$ such that $\mathbf{L}^\top\mathbf{L} = \mathbf{R}^\top\mathbf{R} = \mathbf{I}$ and since PCA/CCA are only identifiable up to a rotation, we would get the same answer.), then ultimately we are interested in projection onto the subspace spanned by $\mathbf{B} = [\mathbf{LU_1} \;\; \mathbf{RV_1}]$. Note that because of our assumption the projection onto $\mathbf{LU_1}$ is $\mathbf{LU_1U_1^\top L^\top}$ and similarly for $\mathbf{RV_1}$. Furthermore, note from our assumptions that $\mathbf{LU_1}$ forms an orthonormal basis for the space spanned by $\mathbf{LU_1}$ (since

$$(\mathbf{LU_1})^\top(\mathbf{LU_1}) = \mathbf{U_1^\top L^\top LU_1} = \mathbf{I},$$

and similarly for $\mathbf{L\hat{U}_1}$, $\mathbf{RV_1}$, and $\mathbf{R\hat{V}_1}$).

Lastly, and critically, the singular values of $\mathbf{U_1^\top L^\top L\hat{U}_1}$ are identical to those of $\mathbf{U_1^\top\hat{U}_1}$ (similarly for $\mathbf{RV_1}$ etc.) and so from above we have that the matrix of canonical angles between the subspaces $\mathbf{LU_1}$ and $\mathbf{L\hat{U}_1}$ are identical to $\mathbf{\Theta}$, the matrix of canonical angles between $\mathbf{U_1}$ and $\mathbf{\hat{U}_1}$, and likewise the matrix of canonical angles between the subspaces $\mathbf{RV_1}$ and $\mathbf{R\hat{V}_1}$ are identical to $\mathbf{\Psi}$, the matrix of canonical angles between $\mathbf{V_1}$ and $\mathbf{\hat{V}_1}$, and thus the maximal angle enjoys the same bound derived above. If we can get a handle on the spectral norm of $\mathbf{E}$, which will come directly from random matrix theory, then we can bound the largest canonical angle of our two subspaces.

We know that $\mathbf{E}$ is a random matrix of iid Gaussian entries with variance $\frac{1}{n}$, and that the largest singular value of a matrix is the spectral norm of the matrix. From random matrix theory we know that the square of the spectral norm of $\mathbf{E}$ is $O(\frac{\sqrt{hv}}{\sqrt{n}})$, from say (Rudelson and Vershynin, 2010).

The strategy will be to divide the variance in the prediction of $\mathbf{y}$ into two separate parts. First the variance that comes from predicting using the incorrect subspace, and then the variance from regression (as stated above) if we had the correct subspace.

Let $\mathbf{\hat{X}} = [\mathbf{L}\hat{\phi}_\mathbf{L} \quad \mathbf{R}\hat{\phi}_\mathbf{R}]$ (i.e. the incorrect subspace) and $\mathbf{X} = [\mathbf{L}\phi_\mathbf{L} \quad \mathbf{R}\phi_\mathbf{R}]$ (the true version). To get a handle on predicting with the incorrect subspace (we will consider the subspaces $\mathbf{L}\phi_\mathbf{L}$ and $\mathbf{R}\phi_\mathbf{R}$ separately here, but note that from (3) the angles between the subspaces and their respective perturbed subspaces are bounded by a common bound) we note that, for the regression of $\mathbf{Y}$ on $\mathbf{X}$ we have

$$\beta|\mathbf{\hat{X}} = \frac{\mathrm{Cov}(\mathbf{Y}, \mathbf{\hat{X}})}{\mathrm{Var}(\mathbf{\hat{X}})},$$

and

$$\beta|\mathbf{X} = \frac{\mathrm{Cov}(\mathbf{Y}, \mathbf{X})}{\mathrm{Var}(\mathbf{X})},$$

and

$$\mathrm{Cov}(\mathbf{Y}, \mathbf{X}) = \mathrm{Cov}(\mathbf{Y}, \mathbf{\hat{X}}),$$

so trivially,

$$\begin{aligned}\beta|\mathbf{\hat{X}} &= \beta|\mathbf{X} * \frac{\mathrm{Var}(\mathbf{X})}{\mathrm{Var}\mathbf{\hat{X}}} \\ &= \beta|\mathbf{X} * \frac{\mathrm{Var}(\mathbf{X})}{\mathrm{Var}(\mathbf{X}) + \mathrm{Var}(\mathbf{X} - \mathbf{\hat{X}})}.\end{aligned}$$

Let $\hat{y}$ be the the estimate of y from the true subspace, and $\hat{\hat{y}}$ be the estimate from the perturbed subspace. For the first part of our strategy, bounding the error that comes from predicting with the incorrect subspace, we want to bound $\mathbb{E}(\hat{y} - \hat{\hat{y}})^2$.

We have,

$$\begin{aligned}
\left[\hat{y} - \hat{\hat{y}}\right]^2 &= \left[\beta|\mathbf{X} * \mathbf{x} - \beta|\hat{\mathbf{X}} * \mathbf{x}\right]^2, \\
&= \left[(\beta|\mathbf{X} - \beta|\hat{\mathbf{X}}) * \mathbf{x}\right]^2, \\
&= \left[\left(\beta|\mathbf{X} - \beta|\mathbf{X}\frac{\mathrm{Var}(\mathbf{X})}{\mathrm{Var}(\mathbf{X}) + \mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}\right) * \mathbf{x}\right]^2, \\
&= \left[\beta|\mathbf{X}\left(\mathbf{1} - \frac{\mathrm{Var}(\mathbf{X})}{\mathrm{Var}(\mathbf{X}) + \mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}\right) * \mathbf{x}\right]^2, \\
&= \left[\beta|\mathbf{X} * \mathbf{x}\left(\frac{\mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}{\mathrm{Var}(\mathbf{X}) + \mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}\right)\right]^2, \\
&= \left[\hat{y} * \left(\frac{\mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}{\mathrm{Var}(\mathbf{X}) + \mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}\right)\right]^2. \qquad (4)
\end{aligned}$$

Because we are working with a ratio of variances instead of actual variances, then without loss of generality we can set $\mathrm{Var}(\hat{\mathbf{X}}) = 1$ for all predictors.

Now, we don't really care what the exact 'true' $\mathbf{X}$'s are (formed with the true singular vectors), because we only care about predicting y and not actually recovering the true β's associated with our SVD. This means we do not suffer from the usual constraints imposed on the erratic behavior of singular vectors. Usually one must handle this kind of error with respect to the entire subspace since singular vectors are highly unstable. In our case, however, we are free to compare to any 'true' vectors we like from the correct subspace, as long as they span the entire true subspace (and nothing more).

We will define a theoretical set of predictors to compare with, then. We are doing this to obtain an upper bound for the total possible variance of $\mathrm{Var}(x - \hat{x})$ for any acceptable set of x's in the true underlying subspace (where we take acceptable to mean that the x's span the true subspace and nothing more).

We handle each subspace $\mathbf{L}\hat{\mathbf{U}}_\mathbf{1}$ and $\mathbf{R}\hat{\mathbf{V}}_\mathbf{1}$ separately. The construction is to take our first vector and choose a vector from the true subspace that lies such that the angle between the two vectors is the maximal canonical angle between the true and perturbed subspaces.

We proceed to our second predictor and choose a vector from the true subspace such the second 'true' predictor is orthogonal to the first. Note that the angle between our second observed $\hat{x}$ and the second chosen x is at most the maximal canonical angle by assumption. Again, because we don't care about the β's associated with our true singular vectors, but only about prediction quality of our perturbed subspace, we need not be worried that our chosen vectors might not be the true singular vectors. We continue in this manner until we have expired all of our predictors from both sets of spaces.

We know from above that the sine of the maximal angle of of both sets of subspaces is $O\left(\frac{\sqrt{hv}}{\sqrt{n}}\right)$ and so we have that the maximal variation

$$\frac{\mathrm{Var}(\mathbf{X} - \hat{\mathbf{X}})}{\mathrm{Var}(\hat{\mathbf{X}})} \sim O\left(\frac{\sqrt{hv}}{\sqrt{n}}\right),$$

and so from 4 we have

$$\mathbb{E}(\hat{y}-\hat{\hat{y}})^2 = \mathbb{E}\left[\hat{y} * O\left(\frac{\sqrt{hv}}{\sqrt{n}}\right)\right]^2$$
$$\approx O\left(\frac{hv}{n} * \frac{1}{v}\right) = O\left(\frac{h}{n}\right).$$

We have v of these to predict, so we have a total error attributable to subspace estimation on the order of $\frac{hv}{n}$. Adding regression error as we did from above, which is on the order of $\frac{kv}{n}$ we get a total error of $\frac{(h+k)v}{n}$. We recall that the error from the one step CCA is on the order of $\frac{hv^2}{n}$ which yields an error ratio of $\frac{h+k}{hv}$. ■

Proof of Lemma 3 and Theorem 4:

Proof Our goal is to find a $v \times k$ matrix $\boldsymbol{\phi}_w$ that maps each of the v words in the vocabulary to a k-dimensional state vector. We will show that the $\boldsymbol{\phi}_w$ we find preserves the information in our data and allows a significant data reduction.

Let $\mathbf{L}$ be an $n \times hv$ matrix giving the words in the left context of each of the n tokens, where the context is of length h, $\mathbf{R}$ be the corresponding $n \times hv$ matrix for the right context, and $\mathbf{W}$ be an $n \times v$ matrix of indicator functions for the words themselves. Also, let l, r and w be the underlying multivariate random variables from which the "observed" matrices $\mathbf{L}$, $\mathbf{R}$ and $\mathbf{W}$ were generated by the data generating process.

We will use three assumptions at various points in our proof:

Assumption 1 *l, r and w come from a rank k HMM i.e it has a rank k observation matrix and a rank k transition matrix both of which have the same domain.*

For example, if the dimension of the hidden state is k and the vocabulary size is v then the observation matrix, which is $k \times v$, has rank k. This rank condition is similar to the one used by (Siddiqi et al., 2010).

Assumption 1A 1 *For the three views, l, r and w assume that there exists a k dimensional "hidden state $\hbar$", such that $\mathbb{E}(l|\hbar) = \hbar\boldsymbol{\beta}_l^\top$ and $\mathbb{E}(r|\hbar) = \hbar\boldsymbol{\beta}_r^\top$ and $\mathbb{E}(w|\hbar) = \hbar\boldsymbol{\beta}_w^\top$ where all $\boldsymbol{\beta}$'s are of rank k.*

This assumption actually follows from the previous one.

Assumption 2 *$\rho(l, w)$, $\rho(l, r)$ and $\rho(w, r)$ all have rank k, where $\rho(a, b)$ is the expected correlation between the random vectors a and b.*

This is a rank condition similar to that in (Hsu et al., 2009).

Assumption 3 *$\rho([l \quad r], w)$ has k distinct singular values.*

This assumption just makes the proof a little cleaner, since if there are repeated singular values, then the singular vectors are not unique. Without it, we would have to phrase results in terms of subspaces with identical singular values.

We also need to define the CCA function that computes the left and right singular vectors for a pair of matrices:

Definition 1 (CCA) *Compute the CCA between multivariate random vectors z and x. Let $\boldsymbol{\phi}_z$ be a matrix containing the d largest singular vectors for z (sorted from the largest on down) and likewise for x. Define the function $CCA(z, x) \equiv [\boldsymbol{\phi}_z, \boldsymbol{\phi}_x]$. When we want just one of these ϕ's, we will use $CCA(z, x)_{left} = \boldsymbol{\phi}_z$ for the left singular vectors and $CCA(z, x)_{right} = \boldsymbol{\phi}_x$ for the right singular vectors.*

Note that the resulting singular vectors, $[\boldsymbol{\phi}_z, \boldsymbol{\phi}_x]$ can be used to give two redundant estimates, $z\boldsymbol{\phi}_z$ and $x\boldsymbol{\phi}_x$ of the "hidden" state relating z and x, if such a hidden state exists.

Lemma 3 *Define $\boldsymbol{\phi}_w$ by the following right singular vectors:*

$$CCA([l \quad r], w)_{right} \equiv \boldsymbol{\phi}_w.$$

Under assumptions 2, 3 and 1A, such that if $CCA(l, r) \equiv [\boldsymbol{\phi}_l, \boldsymbol{\phi}_r]$ then we have

$$CCA([l\boldsymbol{\phi}_l \quad r\boldsymbol{\phi}_r], w)_{right} = \boldsymbol{\phi}_w.$$

This lemma shows that instead of finding the CCA between the full context and the words, we can take the CCA between the Left and Right contexts, estimate a k dimensional state from them, and take the CCA of that state with the words and get the same result.
Proof:
Proof By Assumption 1A, we see that:

$$\mathbb{E}(l\boldsymbol{\beta}_l|\hbar) = \hbar\boldsymbol{\beta}_l^\top\boldsymbol{\beta}_l,$$

and

$$\mathbb{E}(r\boldsymbol{\beta}_r|\hbar) = \hbar\boldsymbol{\beta}_r^\top\boldsymbol{\beta}_r,$$

Since, again by assumption 1Aboth of the $\boldsymbol{\beta}$ matrices have full rank, $\boldsymbol{\beta}_l^\top\boldsymbol{\beta}_l$ is a $k \times k$ matrix of rank k, and likewise for $\boldsymbol{\beta}_r^\top\boldsymbol{\beta}_r$. So

$$\mathbb{E}(\boldsymbol{\beta}_r^\top r^\top l\boldsymbol{\beta}_l|\hbar) = \boldsymbol{\beta}_r^\top\boldsymbol{\beta}_r\hbar^\top\hbar\boldsymbol{\beta}_L\boldsymbol{\beta}_r^\top,$$

i.e.,

$$\boldsymbol{\beta}_r^\top\mathbb{E}(r^\top l)\boldsymbol{\beta}_l = \boldsymbol{\beta}_r^\top\boldsymbol{\beta}_r\mathbb{E}(\hbar^\top\hbar)\boldsymbol{\beta}_l\boldsymbol{\beta}_l^\top,$$

since $\boldsymbol{\beta}_r^\top\boldsymbol{\beta}_r$, $\mathbb{E}(\hbar^\top\hbar)$ and $\boldsymbol{\beta}_l^\top\boldsymbol{\beta}_l$ are all $k \times k$ full rank matrices, $\boldsymbol{\beta}_r$ and $\boldsymbol{\beta}_l$ span the same subspace as the singular values of the CCA between l and r since by Assumption 2 they also have rank k. Similar arguments hold when relating l with w and when relating r with w. Thus if $CCA([l \quad r], w) \equiv [\phi_l, \phi_r]$,

$$CCA(l\boldsymbol{\phi}_l, r\boldsymbol{\phi}_r)_{right} = CCA([l\boldsymbol{\beta}_l \quad r\boldsymbol{\beta}_r], w)_{right},$$

(where we have used Assumption 3 to ensure that not only are the subspaces the same, but that the actual singular vectors are the same.)

Finally by Assumption 3 we know that the rank of $CCA([l \quad r], w)_{right}$ is k, we see that

$$CCA([l\boldsymbol{\beta}_l \quad r\boldsymbol{\beta}_r], w)_{right} = CCA([l \quad r], w)_{right}.$$

Calling this common equality $\boldsymbol{\phi}_w$ yields our result. ■

Let $\boldsymbol{\phi}_w^h$ denote a matrix formed by stacking h copies of $\boldsymbol{\phi}_w$ on top of each other. Right multiplying l or r by $\boldsymbol{\phi}_w^h$ projects each of the words in that context into the k-dimensional reduced rank space.

The following theorem addresses the core of the LR-MVL(II) algorithm, showing that there is an $\boldsymbol{\phi}_w$ which gives the desired dimensionality reduction. Specifically, it shows that the previous lemma also holds in the reduced rank space.

Theorem 4 *Under assumptions 1, 2 and 3 there exists a unique matrix $\boldsymbol{\phi}_w$ such that if*

$$[\boldsymbol{\phi}_l^h, \boldsymbol{\phi}_r^h] \equiv CCA(l\boldsymbol{\phi}_w^h, r\boldsymbol{\phi}_w^h),$$

then

$$\boldsymbol{\phi}_w = CCA([l\boldsymbol{\phi}_w^h\boldsymbol{\phi}_l^h \quad r\boldsymbol{\phi}_w^h\boldsymbol{\phi}_l^h], w)_{right},$$

where $\boldsymbol{\phi}_w^h$ is the stacked form of $\boldsymbol{\phi}_w$.

Proof: We start by noting that Assumption 1 implies Assumption 1A. Thus, the previous lemma follows. So, we know

$$CCA([l \quad r], w)_{right} = CCA([l\boldsymbol{\phi}_l \quad r\boldsymbol{\phi}_r], w)_{right}.$$

Let's define this common quantity as $\boldsymbol{\phi}_w$. This $\boldsymbol{\phi}_w$ has the property that the rank of $CCA(w\boldsymbol{\phi}_w, \hbar)_{left}$ is the same as $CCA(w, \hbar)_{left}$ where $\hbar$ is the hidden state process associated with our data. Hence anything which is not in the domain of $\boldsymbol{\phi}_w$ won't have any correlation with $\hbar$ and hence no correlation with other observed states. So l and $l\boldsymbol{\phi}_w^h$ have the same "information" (predictive power of a linear estimator based on them). More precisely, $[\boldsymbol{\phi}_w^h\boldsymbol{\phi}_l^h, \boldsymbol{\phi}_w^h\boldsymbol{\phi}_r^h] = CCA(l, r)$. Putting this together with the first equation gives the desired result. ■

References

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta sinta(c)tica: a treebank for portuguese. In *Proceedings of LREC*, pages 1698–1703, 2002.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

F. Anscombe. The Transformation of Poisson, Binomial and Negative-Binomial data. *Biometrika*, pages 246–254, 1948.

F. Bach and M. Jordan. A probabilistic interpretation of canonical correlation analysis. In *TR 688, University of California, Berkeley*, 2005.

M. Bansal, K. Gimpel, and K. Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*, 2014.

S. Bird and E. Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python.* O'Reilly Media, 2009.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *In Proceedings of COLT*, pages 92–100, 1998.

P. Brown, P.. deSouza, R. Mercer, V. Della Pietra, and J. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, December 1992. ISSN 0891-2017.

S. Cohen, K. Stratos, M. Collins, D. Foster, and L. Ungar. Spectral learning of latent-variable pcfgs. In *Proceedings of the ACL: Long Papers-Volume 1*, pages 223–231. Association for Computational Linguistics, 2012.

R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12: 2493–2537, 2011.

H. Daumé III. Notes on CG and LM-BFGS Optimization of Logistic Regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`, August 2004.

P. Dhillon, D. Foster, and L. Ungar. Multi-view learning of word embeddings via CCA. In *Proceedings of NIPS*, volume 24, 2011.

P. Dhillon, J. Rodu, M. Collins, D. Foster, and L. Ungar. Spectral dependency parsing with latent variables. In *Proceedings of EMNLP-CoNLL*, 2012a.

P. Dhillon, J. Rodu, D. Foster, and L. Ungar. Two Step CCA: A New Spectral Method for Estimating Vector Models of Words. In *Proceedings of ICML*, 2012b.

S. Dumais, G. Furnas, T. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, pages 281–285. ACM, 1988.

C. Fellbaum. *WordNet.* Wiley Online Library, 1998.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *Proceedings of WWW*, pages 406–414. ACM, 2001.

D. Foster, S. Kakade, and T. Zhang. Multi-view dimensionality reduction via canonical correlation analysis. Technical report, Technical Report TR-2008-4, TTI-Chicago, 2008.

H. Glahn. Canonical Correlation and Its Relationship to Discriminant Analysis and Multiple Regression. *Journal of the Atmospheric Sciences*, 25(1):23–31, January 1968.

N. Halko, P-G. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 2011.

D. Hardoon and J. Shawe-Taylor. Sparse cca for bilingual word generation. In *EURO Mini Conference, Continuous Optimization and Knowledge-Based Technologies*, 2008.

D. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

H. Hotelling. Canonical correlation analysis (CCA). *Journal of Educational Psychology*, 1935.

D. Hsu, S. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. In *Proceedings of COLT*, 2009.

E. Huang, R. Socher, C. Manning, and A. Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of ACL:Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

F. Huang and A. Yates. Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of ACL*, pages 495–503, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9.

F. Huang, A. Ahuja, D. Downey, Y. Yang, Y. Guo, and A. Yates. Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*, 2013.

D. Jurafsky and J. Martin. *Speech & Language Processing*. Pearson Education India, 2000.

S. Kakade and D. Foster. Multi-view regression via canonical correlation analysis. In Nader H. Bshouty and Claudio Gentile, editors, *In Proceedings of COLT*, volume 4539 of *Lecture Notes in Computer Science*, pages 82–96. Springer, 2007.

P. Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT. URL `http://mt-archive.info/MTS-2005-Koehn.pdf`.

T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, 2008.

M. Kromann. The Danish Dependency Treebank and the Underlying Linguistic Theory. In *Proceedings of LREC*, pages 217–220, 2003.

M. Lamar, Y. Maron, M. Johnson, and E. Bienenstock. Svd and clustering for unsupervised pos tagging. In *Proceedings of ACL Short*, pages 215–219, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

T. Landauer, P. Foltz, and D. Laham. An introduction to latent semantic analysis. In *Discourse processes*, 2008.

E. Lefever and V. Hoste. SemEval-2013 Task 10: Cross-Lingual Word Sense Disambiguation. In *In Proceedings of SemEval 2013*, Atlanta, USA, 2013.

M. Marcus, M. Marcinkiewicz, and B. Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, June 1993. ISSN 0891-2017.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, 2013b.

A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of ICML*, pages 641–648, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: http://doi.acm.org/10.1145/1273496.1273577. URL `http://doi.acm.org/10.1145/1273496.1273577`.

P. Netrapalli, P. Jain, and S. Sanghavi. Phase Retrieval using Alternating Minimization. In *Proceedings of NIPS*, pages 2796–2804, 2013.

A. Parikh, S. Cohen, and E. Xing. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of ACL*, 2014.

J. Pennebaker, M. Francis, and R. Booth. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001, 2001.

F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *In Proceedings of ACL*, pages 183–190, 1993.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of CONLL*, pages 147–155, 2009.

T. Rose, M. Stevenson, and M. Whitehead. The reuters corpus volume 1-from yesterday's news to tomorrow's language resources. In *Proceedings of LREC*, volume 2, pages 827–832, 2002.

M. Rudelson and R. Vershynin. Non-asymptotic theory of random matrices: extreme singular values, 2010. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:1003.2990`.

A. Rudnick, C. Liu, and M. Gasser. HLTDI: CL-WSD Using Markov Random Fields for SemEval-2013 Task 10. In *Proceedings of SemEval 2013*, 2013.

M. Seligman. *Flourish: A Visionary New Understanding of Happiness and Well-being*. Free Press, 2011.

S. Siddiqi, B. Boots, and G. J. Gordon. Reduced-rank hidden Markov models. In *Proceedings of AISTATS*, 2010.

K. Simov, P. Osenova, M. Slavcheva, S. Kolkovska, E. Balabanova, D. Doikoff, K. Ivanova, A. Simov, E. Simov, and M. Kouylekov. Building a linguistically interpreted corpus of bulgarian: the bultreebank. In *Proceedings of LREC*, 2002.

N. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL*, pages 354–362. Association for Computational Linguistics, 2005.

R. Socher, B. Huval, C. Manning, and A. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the EMNLP-CoNLL*, pages 1201–1211. Association for Computational Linguistics, 2012.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642. Citeseer, 2013.

G. Stewart. Perturbation theory for the singular value decomposition. In *SVD and Signal Processing, II: Algoritms, Analysis and Applications*, pages 99–109. Elsevier, 1990.

G. Stewart and J. Sun. *Matrix perturbation theory*. Computer science and scientific computing. Academic Press, 1990. ISBN 9780126702309. URL http://books.google.com/books?id=l78PAQAAMAAJ.

J. Suzuki and H. Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL*, 2008.

O. Täckström, R. McDonald, and J. Uszkoreit. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of NAACL-HLT*, pages 477–487. Association for Computational Linguistics, 2012.

S. Teufel. *The structure of scientific articles*. CSLI Publications, 2010.

J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1858681.1858721.

P. Turney and P. Pantel. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

T. Zhang and D. Johnson. A robust risk minimization based named entity recognition system. In *Proceedings of CONLL*, pages 204–207, 2003.