A vector space model of semantics using Canonical Correlation Analysis

Abstract

We present an efficient method that uses canonical correlation analysis (CCA) between words and their contexts (i.e., the neighboring words) to estimate a real-valued vector for each word that characterizes its “hidden state” or “meaning”. The use of CCA allows us to prove theorems characterizing how accurately we can estimate this hidden state. Recently developed algorithms for computing the required singular eigenvectors make it easy to compute models for billions of words of text with vocabularies in the hundreds of thousands. Experiments on the Google-ngram collection show that CCA between words and their contexts provides a mapping from each word to a low dimensional feature vector that captures information about the part of speech and meanings of the word. Unlike latent semantic analysis, which uses PCA, our method takes advantage of the information implicit in the word sequences.

1 The problem of state estimation

Many people have clustered words based on their distributionally similarity (see, among many articles, (Pereira et al., 1993)), but such clustering ignores the many different dimensions that similarity could be computed on. We instead, characterize words using a vector space model (Turney and Pantel, 2010). We present a method for learning language models that estimates a “state” or latent variable representation for words based on their context. The vector learned for each word captures a wide variety of information about it, allowing us to predict the word’s part of speech, linguist features such as animacy, membership in wide variety of semantic categories (foods, drinks, colors, males, females ...) and direction of sentiment (happy vs. sad).

More precisely, 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.

Our CCA-based multi-view learning method can be thought of as a generalization of widely used latent semantic analysis (LSA) methods. LSA (sometimes called LSI, when used for indexing) uses a principle component analysis (PCA) of the co-occurrence matrix between words and the documents they occur in to learn a latent vector representation of each word type (Landauer et al., 2008). We extend this method in two ways: (1) By looking at the correlation between words and the words in their context, we find a latent vector for each word that depends upon the sequence of nearby words, unlike the bag-of-words model standardly used in LSA. (2) We rescale the covariance matrix between words and context to give a correlation matrix (i.e., use CCA), allowing us to prove theorems about how accurately we can estimate the state. Unlike PCA, in CCA the dominant singular vectors are guaranteed to capture the key state information. Importantly, our method scales well, handling gigaword corpora with vocabu-
laries of tens of thousands of words on a small computer.

Using CCA to estimate a latent vector for each word based on the contexts it appears in, gives a significantly different (and as we show below, often more useful) representation than, for example, taking the PCA of the same data used to generate the CCA.

We show below how to efficiently compute a vector that characterizes each word type by using the right 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 attribute dictionary for words, as it associates with every word a vector that captures that word’s attributes. As will be made clear below, the attribute 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.

Characterizing words by a low dimensional (e.g., 50 dimensional) vector as we do is valuable in that words can be partitioned in many ways. They can be grouped based on part of speech, or by many aspects of their “meaning” including features such as animacy, gender, whether the word describes something that is good to eat or drink, whether it refers to a year or to a small number, etc. Such features can be used, for example, as features for word sense disambiguation or named entity disambiguation, or to help improve parsing algorithms.

This paper:

- presents a simple CCA-based spectral method for estimating state in language models,
- gives an efficient implementation of our CCA method,
- proves that we provide an optimal reconstruction of the state associated with a word (given, of course, some assumptions), and
- applies this method to the Google n-gram collection and gives some examples showing what features are captured by the state.

2 CCA-based language modeling

The basic data we use is a matrix, \(A\), of co-occurrence counts between word and context, where each column corresponds to a word in the vocabulary (size \(v\)) and each row corresponds to a word in the vocabulary at each location relative to the target word. For example, for bigrams, \(A_{2v \times v}\) would have rows for each possible word immediately before the target word and for each word immediately after the target word. Five-grams give a similar matrix, \(A_{4v \times v}\).

The key idea is to compute the singular value decomposition (SVD) of \(A = \Psi \Lambda \Phi^T\) or, as described below, a rescaled version of \(A\) which contains the correlations between words and their contexts, rather than the covariances (the counts). We then use the left and right singular vectors \(\Psi\) and \(\Phi\) to estimate the hidden state associated with each word.

The \(k\) “largest” right singular vectors \(\phi_i\) of \(A^T A\), i.e., the solutions of \(A^T A \phi_i = \lambda_i \phi_i\) with the largest singular values \(\lambda_i\), form the matrix \(\Phi\) (our “attribute dictionary”), where each of the \(v\) rows is a \(k\)-dimensional estimate of a latent vector associated with one of the words in the vocabulary.

The \(k\) “largest” left singular vectors \(\psi_i\) of \(A A^T\), i.e., the solutions of \(A A^T \psi_i = \lambda_i \psi_i\) form the matrix \(\Psi\), which gives a mapping from a context to the state associated with it. (That is, an estimate of the state of the target word inside the context.)

When \(A\) is taken to be the correlation matrix between word and context, both of these estimates are optimal in a sense which is made precise below.

2.1 Theoretical properties

We now discuss how well the hidden state can be estimated from the target word. (A similar, but slightly more technical 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.

We start with a document consisting of \(n\) tokens \(w_1, w_2, ..., w_n\) drawn from a vocabulary of \(v\) words (actually, it is the concatenation of a vast number of documents). Let \(W\) and \(C\) be matrices in which the \(i^{th}\) row describes either the \(i^{th}\) token, \(w_i\) (for \(W\)) or its context – the words to its right and left – (for \(C\)). In \(W\), we represent the presence of the \(j^{th}\) word type in the \(i^{th}\) position in a document by setting matrix element \(w_{ij} = 1\). \(C\) is similar, but has columns for each word in each position in the
Let $\beta$ be a linear function of the context (i.e., $Y_t = \alpha^T C_t$)

2. $\beta^T W_i$ is the best linear estimator of $Y_t$ given $W_i$, namely $\beta$ minimizes $E (Y_t - \beta^T W_i)^2$

3. $\operatorname{Var}(Y_t) \leq 1$.

Proof of Theorem 1:

We can write $\alpha$ and $\beta$ in this coordinate system.

By orthogonality we now have $\beta_i = \rho_i \alpha_i$. So, $E(Y - \beta W)^2 = \sum (\alpha_i - \beta_i \rho_i)^2$. This 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 $(\hat{Y} - \beta^TW)^2 = \sum_{i=k+1}^\infty \beta_i^2$.

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

q.e.d.

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 dimensional. Instead, we can first use CCA to map high dimensional vectors $w$ to lower dimensional vectors $\phi$, from which $y$ can be predicted. For example with a few labeled examples of the form $(w, y)$, we can recover the $\gamma_i$ parameters using linear regression. The $\phi$ subspace is guaranteed to hold a good approximation. A special case of interest occurs when estimating a label $Z$ which is $\alpha^T C_T$ plus zero mean noise. In this case, one can pick $Y = E(Z)$ 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 (i.e., not normalized by $A_{cc}^{-1/2}$ and $A_{ww}^{-1/2}$, and hence computed the canonical covariance rather than the canonical 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

Define $\phi_i$ to be the $i^{\text{th}}$ right singular vector for the SVD of Eq. 1 with $A_{cc} = E(CC^T)$, $A_{cw} = E(CW^T)$, and $A_{ww} = E(WW^T)$ where $(W, C)$ are drawn from the marginal distribution $D(w, c)$. 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 $\hat{Y}_t = \sum_{i=1}^k \gamma_i \phi_i$ is a good approximation to $Y_t$ in the sense that $E(\hat{Y}_t - \beta^TW)^2 \leq \epsilon$.
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. In practice, one can often still get good estimates for our method without the normalization.

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 left 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. To keep this paper simple, we will not discuss this approach, but instead focus on the simpler use of right canonical covariates to map each word type to a $k$ dimensional vector. Below, we show examples illustrating the information captured by this state estimate.

3 Efficient estimation algorithm

Recent advances in algorithms for computing SVD decompositions of matrices make it easy to compute the singular values of large matrices (Halko et al., 2011), thus making spectral methods easy to use on data with very large vocabularies, such as those in the Google n-gram collection. The key idea is to find a lower dimensional basis for $A$, 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 $A$ is $vd \times v$, and we seek a state of dimension $k$, we start with a $(k+l) \times vd$ matrix $\Omega$ of random numbers, where $l$ is number of “extra” basis vectors between 0 and $k$. We then project $A$ onto this matrix and take the SVD decomposition of the resulting matrix $\Omega A = U_1 D_1 V_1^T$. Since $\Omega A$ is $(k + l) \times vd$, this is much cheaper than working on the original matrix $A$. We keep the largest $k$ components of $U_1$ and of $V_1$, which form a left and a right basis for $A$ respectively. The rank $k$ matrices $U^T U A$ and $AV^T V$ from good approximations to $A$. The matrix $V$ approximates the dominant left singular vectors of $A$, $\Phi$, and $U$ approximates the right singular vectors, $\Psi$. Better approximations can be found by iterating.

We take two more iterations, as summarized in algorithm 1. First find the SVD of $AV_1^T = U_2 D_2 V_2^T$. $U_2$ and $V_2^T$ form improved estimates for the left and right singular vectors of $A$. Then find the SVD of $AV_2^T = U_3 D_3 V_3^T$ to find yet better estimates. This iteration rapidly converges.

(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 4 and 5 produces an approximation whose error is bounded by a small polynomial factor times the size of the largest singular value whose singular vectors are not part of the approximation, $\sigma_{k+1}$. 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, $\sigma_{k+1}$.

A practical question to be answered is how to normalize the data to generate the $A$ matrix. Normalizing $A_{cw}$ by postmultiplying by $A_{cw}^{-1/2}$ is trivial, since the later matrix is diagonal. Normalizing $A_{cw}$ by $A_{cw}^{-1/2}$ is, however, not practical for large data sets, as it requires inverting a prohibitively large matrix. In practice, one can either replace this matrix by the identity, as is often done in CCA (see e.g., (Witten and Tibshirani, 2009)) or one can use similar dimensionality reduction techniques to those described above, and take advantage of the fact that the

<table>
<thead>
<tr>
<th>Algorithm 1 Randomized singular value decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: <strong>Input</strong>: matrix $A$ of size $c \times w$, the desired hidden state dimension $k$, and the number of “extra” singular vectors, $l$</td>
</tr>
<tr>
<td>2: Generate a $(k + l) \times c$ random matrix $\Omega$</td>
</tr>
<tr>
<td>3: Find the SVD $U_1 D_1 V_1^T$ of $\Omega A$, and keep the $k$ components of $V_1$ with the largest singular values</td>
</tr>
<tr>
<td>4: Find the SVD $U_2 D_2 V_2^T$ of $AV_1^T$</td>
</tr>
<tr>
<td>5: Find the SVD $U_3 D_3 V_3^T$ of $AV_2^T$</td>
</tr>
<tr>
<td>6: <strong>Output</strong>: The canonical covariates $\Psi = U_3$, $\Phi = V_3$</td>
</tr>
</tbody>
</table>
inverse of $A_{cc}$ can be well approximated by finding its (approximate) SVD composition $UDV'$ and then inverting the diagonal matrix $D$.

4 Experimental Results

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.

In the results presented below, we started with the Google n-gram collection which has counts of the most frequent 1 to 5-grams on the web. The collection is extensive, having a vocabulary of thirteen million words and over a billion 5-grams derived from a trillion words of text from the web.

4.1 Qualitative Evaluation

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 “attribute dictionary” values, which are simply the right canonical correlates calculated from equation 1. (The first principle component generally just separates the selected words from other words, and so is less interesting here.)

In all of the following examples, a 30-dimensional state vector was estimated using as context the one word proceeding and one word after each word in the 3-gram collection of the Google n-grams. The middle (second) word of each trigram is then the word being correlated with the context. Note that this procedure is fully unsupervised, although we could easily train a classifier based on the attribute dictionary to separate out different classes of words using only a few hand-labeled examples.

To make clear the contrast with PCA, we take exactly the same trigrams, and do PCA on their them. (To make this look like LSA, think of each trigram being a document containing three words. Since a given trigram will occur many times, there are then many copies of each “document”.)

Figure 1 shows plots for three different sets of words. The left column uses the attribute dictionary from CCA, while the right column uses the corresponding latent vectors derived using PCA on the same data. In all cases, the 30-dimensional vectors have been projected onto two dimensions (using PCA) so that they can be visualized.

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. Good separation is achieved with no supervision. 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 second row in Figure 1 shows the CCA and PCA attribute dictionaries for a few pronouns and possessives. In the CCA plot on the left, we can see the nominative pronouns (he, she, they) in the lower right corner, the possessives (his, her) on the lower left of the figure. Third person singular pairs (he/she and his/her) are particularly close together. Note that two dimensions are not sufficient to fully separate the different parts of speech here; different projections of the data would separate out those parts of speech. Again PCA fails to give clear separation.

When one picks sets of words that are highly similar, such as names of people, the projections reveal more subtle features of those words. The third row shows a fairly good separation of male and female names along the diagonal from lower left to upper right. Closer inspection of the plot reveals that the other dimension separates names based on formality, with more complete names such as Joseph and Thomas on the lower right and shorter versions like Joe and Tom on the upper left.

The latent state as estimated by CCA thus captures a wide variety of attributes of the words: singular or plural, formal or informal, and as shown below, good to eat or not, happy or sad, successful or failed – all from a single run of doing CCA.

---

\[\text{http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html}\]
Figure 1: Projections onto two dimension of selected words in different categories using both CCA (left) and PCA (Right), all using the same trigrams from Google n-grams.
Figure 2 shows attribute dictionary projections for a few foods and drinks. (Due to space limitations, only the CCA is shown.) One can see drinks in the upper left, carrots and lettuce on the lower right, and a variety of dishes that one might eat for a meal on the lower left. (I know, that implies that one wouldn’t eat carrots or lettuce; let’s just note that different people or animals eat them.)

![Figure 2: Projections of the attribute dictionary from CCA on the Google tri-grams of words representing items to eat or drink.](image)

We have also plotted (not shown) names of numbers or the numerals representing numbers. Numbers that are close to each other in value tend to be close in the plot, thus suggesting that state captures not just classifications, but also more continuous hidden features.

The theme running through these figures is that the attribute dictionary, computed using the right canonical correlates (or singular vectors), captures a rich set of features characterizing the words. These vectors could be used in many applications to generalize beyond the particular words they represent.

### 4.2 Quantitative Evaluation

The attribute dictionary which we learn forms a low rank representation of each of the words in the vocabulary. These vectors can then be used as features in supervised models to predict various labels on the words, such as part of speech, word sense, or entity type.

In addition to the unsupervised results shown in the above pictures, we also evaluated our vector models of words on a number of supervised binary classification problems. The word lists for all these problems were gotten from Wikipedia or from Martin Seligman’s PERMA lexicon (Seligman, 2011), and the state vector for each word was learned using Google n-gram data as described above. (“PERMA” is a positive psychology construct that measures how much people have positive emotion, engagement, relationships, meaning, and a sense of achievement in their lives. These attributes constitute measures of subjective well-being.)

In all the experiments we used 75% of the data (See Table 1) chosen randomly for training and remaining data for testing and the numbers reported are averaged over 10 such runs. We used a SVM (Chang and Lin, 2001) with RBF kernel for the binary classification task. The test-set accuracies for CCA and PCA are shown in Table 2 and as can be seen CCA is significantly better in all but one binary classification tasks. The only exception is male vs. female names classification task, where using the non-sequential PCA models works as well as CCA.

### 5 Related work

There is, of course, a very long tradition of using vector space models of language, almost universally based on PCA. See, for example, the excellent recent review in (Turney and Pantel, 2010). However, none of the methods cited in that paper use CCA or its non-whitened cousin, canonical covariance estimation. PCA, which is by far the most widely used vector space model, tries to capture all of the vari-

<table>
<thead>
<tr>
<th>Word sets</th>
<th>Number of observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive emotion or not</td>
<td>Class I: 81, Class II: 162</td>
</tr>
<tr>
<td>Meaningful life or not</td>
<td>Class I: 246, Class II: 46</td>
</tr>
<tr>
<td>Achievement or not</td>
<td>Class I: 159, Class II: 70</td>
</tr>
<tr>
<td>Engagement or not</td>
<td>Class I: 208, Class II: 93</td>
</tr>
<tr>
<td>Relationship or not</td>
<td>Class I: 236, Class II: 204</td>
</tr>
<tr>
<td>Male vs. Female name</td>
<td>Class I: 447, Class II: 330</td>
</tr>
<tr>
<td>Number vs. Color</td>
<td>Class I: 33, Class II: 34</td>
</tr>
<tr>
<td>Animal vs. Bird</td>
<td>Class I: 129, Class II: 20</td>
</tr>
</tbody>
</table>

Table 1: Description of the datasets used. All the data was collected either from Wikipedia or PERMA lexicon. We plan to make the data available publicly after the conference.
Table 2: Test set accuracies for each binary classification problem. “Majority baseline” always uses the more common label. PCA and CCA are computed as described above using the Google n-grams to estimate the word vectors and then using these attribute vectors as features in an SVM. The first five categories are from the “PERMA” lexicon (see text). Note: The numbers in bold are statistically significant at 5% level over the 10 runs.

<table>
<thead>
<tr>
<th>Word sets</th>
<th>Majority Baseline</th>
<th>PCA</th>
<th>CCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive emotion or not</td>
<td>66.1</td>
<td>95.3 ± 0.2</td>
<td>97.6 ± 0.1</td>
</tr>
<tr>
<td>Engagement or not</td>
<td>68.4</td>
<td>94.9 ± 0.2</td>
<td>97.4 ± 0.2</td>
</tr>
<tr>
<td>Relationship or not</td>
<td>53.7</td>
<td>96.1 ± 0.4</td>
<td>98.0 ± 0.3</td>
</tr>
<tr>
<td>Meaningful life or not</td>
<td>83.8</td>
<td>95.9 ± 0.1</td>
<td>97.7 ± 0.3</td>
</tr>
<tr>
<td>Achievement or not</td>
<td>69.0</td>
<td>97.3 ± 0.3</td>
<td>99.1 ± 0.4</td>
</tr>
<tr>
<td>Number vs. Color</td>
<td>50.0</td>
<td>95.8 ± 0.2</td>
<td>99.7 ± 0.4</td>
</tr>
<tr>
<td>Male vs. Female name</td>
<td>56.9</td>
<td>84.2 ± 0.2</td>
<td>84.3 ± 0.3</td>
</tr>
<tr>
<td>Animal vs. Bird</td>
<td>86.9</td>
<td>97.1 ± 0.1</td>
<td>98.2 ± 0.2</td>
</tr>
</tbody>
</table>

There has been some recent work on semi-supervised sequence labeling (Jiao and et al, 2006; Mann and McCallum, 2010), but these models generally use domain-specific constraints, and are much more cumbersome and slow to train than CCA-based methods.

In contrast, our CCA-based method scales extremely well, is guaranteed to find its single global optimal solution, has provable computational complexity and approximation accuracy.

Closer in style to our CCA-based approach, there is a recent resurgence of interest in using CCA-style spectral methods for estimating HMMs (Hsu et al., 2009) or similar linear dynamical systems (Siddiqi et al., 2010; Song et al., 2010). These methods, while attractive in, like us, making use of sequence information, tend to be slightly more complex and difficult to analyze than the algorithm presented in this paper. The more complex spectral sequence methods have also not been demonstrated to work on real language problems.

The major application CCA to language has been in the field of machine translation, where it has been recognized that CCA in which the two views are composed of corresponding text in different languages (instead of our context and word pairs) can be used to extract latent vectors with the shared meaning between the languages. (Hardoon and Shawe-Taylor, 2008; Haghighi et al., 2008)

6 Discussion

We have argued that for many problems, CCA can give better feature vectors for words than PCA. CCA, in our application, finds the components of maximum correlation between the context words, taking into account their location in the context, with the word of interest, unlike PCA on n-grams, which finds treats all words in the n-gram equivalently.

PCA and CCA share deep similarities, not just in both being spectral methods. If the word co-
occurrence matrix for PCA is normalized by scaling each word by dividing by the square of its overall frequency, then in the special case of bigrams, PCA and CCA become identical. In this special case, the context and the target word covariances $C' C$ and $W' W$ become (after normalization) the identity matrix. Since CCA scales by the inverse of these covariance matrices, if they are identity matrices, PCA and CCA will have identical singular vectors.

In the more common case of a larger context, PCA will devote more degrees of freedom to finding the structure within the context, while CCA will focus on finding the correlation between context and target word. If we could afford to compute and use larger state spaces, this would not be too serious, but because we are working with large corpora and large vocabularies, even a five-fold reduction in the number of components that is kept matters.

More broadly, we have argued that a single vector for each word can capture a wide range of attributes of that word including, part of speech, animacy, sex, edibility, etc. One could instead have clustered words based on distributional similarity using, e.g., (Pereira et al., 1993) or (Brown et al., 1992), but one would need some complex multifaceted hierarchical clustering scheme to come close to capturing the different dimensions represented in the attribute vectors. For example, should “he” and “she” be in the same or different clusters? The words are very similar on many dimensions, but opposed on at least one. Using vector models also has advantages over categories in allowing word meaning to sit on a continuum, rather than being binned into discrete categories. There is substantial evidence from human studies that word meanings are often interpreted on a graded scale (Erk and McCarthy, 2009), rather than categorically.

This paper has focused on the right canonical covariates (CCs), which give vectors characterizing each word type. We showed that these vectors characterize both the part of speech and the "meaning" of words. These right CCs, that we called the "attribute dictionary", have the disadvantage that for words with multiple parts of speech or meanings, they will be a weighted average of these meanings. In such cases, one could use the attribute dictionary to map the context words down to the low dimensional state space and then predict properties of a particular token based on the reduced dimensional representation of its context. Alternatively, one could use the left CCs, which map from the context each token to its associated state. Either method gives different state estimates for the same word in different contexts and can be used for problem such as POS tagging, word sense disambiguation, and named entity disambiguation. Under the assumption that the data are generated from a Hidden Markov Model (HMM), one can give proofs of power of the left CCs that are similar in flavor to Theorem 1 or to the HMM estimation schemes in (Hsu et al., 2009).

We are currently extending this work in two directions. We are exploring the use of larger contexts than the n-gram models we used above, and we are starting to use similar CCA-based state estimation methods to learn probabilistic context free grammars. We believe that characterizing words using a vector-valued attribute dictionary will offer advantages over the (hard) word clusters used in some recent lexicalized parsers such as (Koo et al., 2008).

References


