A Spectral Algorithm for Learning Class-Based n-gram Models of Natural Language

Karl Stratos[†] **Do-kyum Kim**[‡] **Michael Collins**[†] **Daniel Hsu**[†]
[†]Department of Computer Science, Columbia University, New York, NY 10027
[‡]Department of Computer Science and Engineering, University of California–San Diego, La Jolla, CA 92093

Abstract

The Brown clustering algorithm (Brown *et al.*, 1992) is widely used in natural language processing (NLP) to derive lexical representations that are then used to improve performance on various NLP problems. The algorithm assumes an underlying model that is essentially an HMM, with the restriction that each word in the vocabulary is emitted from a single state. A greedy, bottom-up method is then used to find the clustering; this method does not have a guarantee of finding the correct underlying clustering. In this paper we describe a new algorithm for clustering under the Brown *et al.* model. The method relies on two steps: first, the use of canonical correlation analysis to derive a low-dimensional representation of words; second, a bottom-up hierarchical clustering over these representations. We show that given a sufficient number of training examples sampled from the Brown *et al.* model, the method is guaranteed to recover the correct clustering. Experiments show that the method recovers clusters of comparable quality to the algorithm of Brown *et al.* (1992), but is an order of magnitude more efficient.

1 INTRODUCTION

There has recently been great interest in the natural language processing (NLP) community in methods that derive lexical representations from large quantities of unlabeled data (Brown *et al.*, 1992; Pereira *et al.*, 1993; Ando and Zhang, 2005; Liang, 2005; Turian *et al.*, 2010; Dhillon *et al.*, 2011; Collobert *et al.*, 2011; Mikolov *et al.*, 2013a,b). These representations can be used to improve accuracy on various NLP problems, or to give significant reductions in the number of training examples required for learning. The Brown clustering algorithm (Brown *et al.*, 1992) is one of the most widely used algorithms for this task. Brown clustering representations have been shown to be useful in a diverse set of problems including named-entity recognition (Miller *et al.*, 2004; Turian *et al.*, 2010), syntactic chunking (Turian *et al.*, 2010), parsing (Koo *et al.*, 2008), and language modeling (Kneser and Ney, 1993; Gao *et al.*, 2001).

The Brown clustering algorithm assumes a model that is essentially a hidden Markov model (HMM), with a restriction that each word in the vocabulary can only be emitted from a single state in the HMM (i.e, there is a deterministic mapping from words to underlying states). The algorithm uses a greedy, bottom-up method in deriving the clustering. This method is a heuristic, in that there is no guarantee of recovering the correct clustering. In practice, the algorithm is quite computationally expensive: for example in our experiments, the implementation of Liang (2005) takes over 22 hours to derive a clustering from a dataset with 205 million tokens and 300,000 distinct word types.

This paper introduces a new algorithm for clustering under the Brown *et al.* model (henceforth, the Brown model). Crucially, under an assumption that the data is generated from the Brown model, our algorithm is guaranteed to recover the correct clustering when given a sufficient number of training examples (see the theorems in Section 5). The algorithm draws on ideas from canonical correlation analysis (CCA) and agglomerative clustering, and has the following simple form:

1. Estimate a normalized covariance matrix from a corpus and use singular value decomposition (SVD) to derive low-dimensional vector representations for word types (Figure 4).
2. Perform a bottom-up hierarchical clustering of these vectors (Figure 5).

In our experiments, we find that our clusters are comparable to the Brown clusters in improving the performance of a supervised learner, but our method is significantly faster. For example, both our clusters and Brown clusters improve the F1 score in named-entity recognition (NER) by 2-3 points, but the runtime of our method is around 10 times faster than the Brown algorithm (Table 3).

The paper is structured as follows. In Section 2, we discuss

Input: corpus with N tokens of n distinct word types $w^{(1)}, \ldots, w^{(n)}$ ordered by decreasing frequency; number of clusters m.
Output: hierarchical clustering of $w^{(1)}, \ldots, w^{(n)}$.

1. Initialize active clusters $\mathcal{C} = \{\{w^{(1)}\}, \ldots, \{w^{(m)}\}\}$.
2. For $i = m + 1$ to $n + m - 1$:
   (a) If $i \leq n$: set $\mathcal{C} = \mathcal{C} \cup \{\{w^{(i)}\}\}$.
   (b) Merge $c, c' \in \mathcal{C}$ that cause the smallest decrease in the likelihood of the corpus.

Figure 1: A standard implementation of the Brown clustering algorithm.

related work. In Section 3, we establish the notation we use throughout. In Section 4, we define the Brown model. In Section 5, we present the main result and describe the algorithm. In Section 6, we report experimental results.

2 BACKGROUND

2.1 THE BROWN CLUSTERING ALGORITHM

The Brown clustering algorithm (Brown *et al.*, 1992) has been used in many NLP applications (Koo *et al.*, 2008; Miller *et al.*, 2004; Liang, 2005). We briefly describe the algorithm below; a part of the description was taken from Koo *et al.* (2008).

The input to the algorithm is a corpus of text with N tokens of n distinct word types. The algorithm initializes each word type as a distinct cluster, and repeatedly merges the pair of clusters that cause the smallest decrease in the likelihood of the corpus according to a discrete hidden Markov model (HMM). The observation parameters of this HMM are assumed to satisfy a certain disjointedness condition (Assumption 4.1). We will explicitly define the model in Section 4.

At the end of the algorithm, one obtains a hierarchy of word types which can be represented as a binary tree as in Figure 2. Within this tree, each word is uniquely identified by its path from the root, and this path can be compactly represented with a bit string. In order to obtain a clustering of the words, we select all nodes at a certain depth from the root of the hierarchy. For example, in Figure 2 we might select the four nodes at depth 2 from the root, yielding the clusters {apple,pear}, {Apple,IBM}, {bought,run}, and {of,in}. Note that the same clustering can be obtained by truncating each word's bit string to a 2-bit prefix. By using prefixes of various lengths, we can produce clusterings of different granularities.

A naive implementation of this algorithm has runtime $O(n^5)$. Brown *et al.* (1992) propose a technique to reduce the runtime to $O(n^3)$. Since this is still not acceptable

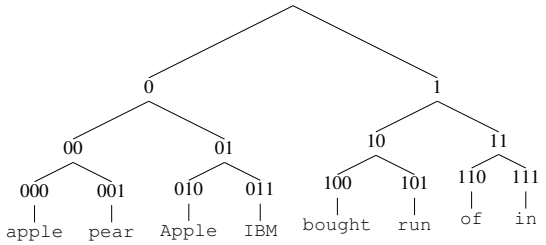


Figure 2: An example of a Brown word-cluster hierarchy taken from Koo *et al.* (2008). Each node in the tree is labeled with a bit string indicating the path from the root node to that node, where 0 indicates a left branch and 1 indicates a right branch.

for large values of n, a common trick used for practical implementation is to specify the number of active clusters $m \ll n$, for example, $m = 1000$. A sketch of this implementation is shown in Figure 1. Using this technique, it is possible to achieve $O(N + nm^2)$ runtime. We note that our algorithm in Figure 5 has a similar form and asymptotic runtime, but is empirically much faster. We discuss this issue in Section 6.3.1.

In this paper, we present a very different algorithm for deriving a word hierarchy based on the Brown model. In all our experiments, we compared our method against the highly optimized implementation of the Brown algorithm in Figure 1 by Liang (2005).

2.2 CCA AND AGGLOMERATIVE CLUSTERING

Our algorithm in Figure 4 operates in a fashion similar to the mechanics of CCA. CCA is a statistical technique used to maximize the correlation between a pair of random variables (Hotelling, 1936). A central operation in CCA to achieve this maximization is SVD; in this work, we also critically rely on SVD to recover the desired parameters.

Recently, it has been shown that one can use CCA-style algorithms, so-called spectral methods, to learn HMMs in polynomial sample/time complexity (Hsu *et al.*, 2012). These methods will be important to our goal since the Brown model can be viewed as a special case of an HMM.

We briefly note that one can view our approach from the perspective of spectral clustering (Ng *et al.*, 2002). A spectral clustering algorithm typically proceeds by constructing a graph Laplacian matrix from the data and performing a standard clustering algorithm (e.g., k-means) on reduced-dimensional points that correspond to the top eigenvalues of the Laplacian. We do not make use of a graph Laplacian, but we do make use of spectral methods for dimensionality reduction before clustering.

Agglomerative clustering refers to hierarchical grouping of n points using a bottom-up style algorithm (Ward Jr, 1963; Shanbehzadeh and Ogunbona, 1997). It is commonly used for its simplicity, but a naive implementation

requires $O(dn^3)$ time where d is the dimension of a point. Franti *et al.* (2000) presented a faster algorithm that requires $O(\gamma dn^2)$ time where γ is a data-dependent quantity which is typically much smaller than n. In our work, we use a variant of this last approach that has runtime $O(\gamma dmn)$ where $m \ll n$ is the number of active clusters we specify (Figure 5). We also remark that under our derivation, the dimension d is always equal to m, thus we express the runtime simply as $O(\gamma nm^2)$.

3 NOTATION

Let $[n]$ denote the set $\{1, \ldots, n\}$. Let $[[\Gamma]]$ denote the indicator of a predicate Γ, taking value 1 if Γ is true and 0 otherwise. Given a matrix M, we let $\sqrt{M}$ denote its element-wise square-root and M^+ denote its Moore-Penrose pseudoinverse. Let $I_{m\times m} \in \mathbb{R}^{m\times m}$ denote the identity matrix. Let $\text{diag}(v)$ denote the diagonal matrix with the vector $v \in \mathbb{R}^m$ appearing on its diagonal. Finally, let $\|v\|$ denote the Euclidean norm of a vector v, and $\|M\|$ denote the spectral norm of a matrix M.

4 BROWN MODEL DEFINITION

A Brown model is a 5-tuple (n, m, π, t, o) for integers n, m and functions π, t, o where

- $[n]$ is a set of states that represent word types.
- $[m]$ is a set of states that represent clusters.
- $\pi(c)$ is the probability of generating $c \in [m]$ in the first position of a sequence.
- $t(c'|c)$ is the probability of generating $c' \in [m]$ given $c \in [m]$.
- $o(x|c)$ is the probability of generating $x \in [n]$ given $c \in [m]$.

In addition, the model makes the following assumption on the parameters $o(x|c)$. This assumption comes from Brown *et al.* (1992) who require that the word clusters partition the vocabulary.

Assumption 4.1 (Brown *et al.* assumption)**.** *For each $x \in [n]$, there is a unique $C(x) \in [m]$ such that $o(x|C(x)) > 0$ and $o(x|c) = 0$ for all $c \neq C(x)$.*

In other words, the model is a discrete HMM with a many-to-one deterministic mapping $C : [n] \to [m]$ from word types to clusters. Under the model, a sequence of N tokens $(x_1, \ldots, x_N) \in [n]^N$ has probability

$$p(x_1, \ldots, x_N) = \pi(C(x_1)) \times \prod_{i=1}^{N} o(x_i|C(x_i)) \times \prod_{i=1}^{N-1} t(C(x_{i+1})|C(x_i))$$

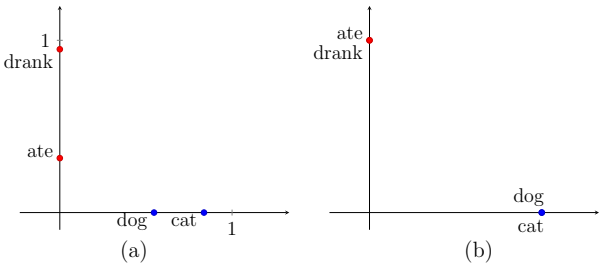


Figure 3: Illustration of our clustering scheme. (a) Original rows of $\sqrt{O}$. (b) After row-normalization.

An equivalent definition of a Brown model is given by organizing the parameters in matrix form. Under this definition, a Brown model has parameters (π, T, O) where $\pi \in \mathbb{R}^m$ is a vector and $T \in \mathbb{R}^{m\times m}, O \in \mathbb{R}^{n\times m}$ are matrices whose entries are set to:

- $\pi_c = \pi(c)$ for $c \in [m]$
- $T_{c',c} = t(c'|c)$ for $c, c' \in [m]$
- $O_{x,c} = o(x|c)$ for $c \in [m], x \in [n]$

Throughout the paper, we will assume that T, O have rank m. The following is an equivalent reformulation of Assumption 4.1 and will be important to the derivation of our algorithm.

Assumption 4.2 (Brown *et al.* assumption)**.** *Each row of O has exactly one non-zero entry.*

5 CLUSTERING UNDER THE BROWN MODEL

In this section, we develop a method for clustering words based on the Brown model. The resulting algorithm is a simple two-step procedure: an application of SVD followed by agglomerative hierarchical clustering in Euclidean space.

5.1 AN OVERVIEW OF THE APPROACH

Suppose the parameter matrix O is known. Under Assumption 4.2, a simple way to recover the correct word clustering is as follows:

1. Compute $\bar{M} \in \mathbb{R}^{n\times m}$ whose rows are the rows of $\sqrt{O}$ normalized to have length 1.
2. Put words x, x' in the same cluster iff $\bar{M}_x = \bar{M}_{x'}$, where $\bar{M}_x$ is the x-th row of $\bar{M}$.

This works because Assumption 4.2 implies that the rows of $\sqrt{O}$ corresponding to words from the same cluster lie along the same coordinate-axis in $\mathbb{R}^m$. Row-normalization puts these rows precisely at the standard basis vectors. See Figure 3 for illustration.

In Section 5.2, we prove that the rows of $\sqrt{O}$ can be recovered, up to an orthogonal transformation $Q \in \mathbb{R}^{m\times m}$, just from unigram and bigram word probabilities (which can be estimated from observed sequences). It is clear that the correctness of the above procedure is unaffected by the orthogonal transformation. Let M denote the row-normalized form of $\sqrt{O}Q^\top$: then M still satisfies the property that $M_x = M_{x'}$ iff x, x' belong to the same cluster. We give an algorithm to estimate this M from a sequence of words in Figure 4.

5.2 SPECTRAL ESTIMATION OF OBSERVATION PARAMETERS

To derive a method for estimating the observation parameter $\sqrt{O}$ (up to an orthogonal transformation), we first define the following random variables to model a single random sentence. Let $(X_1, \ldots, X_N) \in [n]^N$ be a random sequence of tokens drawn from the Brown model, along with the corresponding (hidden) cluster sequence $(C_1, \ldots, C_N) \in [m]^N$; independently, pick a position $I \in [N-1]$ uniformly at random. Let $B \in \mathbb{R}^{n\times n}$ be a matrix of bigram probabilities, $u, v \in \mathbb{R}^n$ vectors of unigram probabilities, and $\tilde{\pi} \in \mathbb{R}^m$ a vector of cluster probabilities:

$$\begin{aligned} B_{x,x'} &:= P(X_I = x, X_{I+1} = x') && \forall x, x' \in [n] \\ u_x &:= P(X_I = x) && \forall x \in [n] \\ v_x &:= P(X_{I+1} = x) && \forall x \in [n] \\ \tilde{\pi}_c &:= P(C_I = c) && \forall c \in [m]. \end{aligned}$$

We assume that $\mathrm{diag}(\tilde{\pi})$ has rank m; note that this assumption is weaker than requiring $\mathrm{diag}(\pi)$ to have rank m. We will consider a matrix $\Omega \in \mathbb{R}^{n\times n}$ defined as

$$\Omega := \mathrm{diag}(u)^{-1/2} B \mathrm{diag}(v)^{-1/2} \tag{1}$$

Theorem 5.1. *Let $U \in \mathbb{R}^{n\times m}$ be the matrix of m left singular vectors of Ω corresponding to nonzero singular values. Then there exists an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ such that $U = \sqrt{O}Q^\top$.*

To prove Theorem 5.1, we need to examine the structure of the matrix Ω. The following matrices $A, \tilde{A} \in \mathbb{R}^{n\times m}$ will be important for this purpose:

$$\begin{aligned} A &= \mathrm{diag}(O\tilde{\pi})^{-1/2} O \mathrm{diag}(\tilde{\pi})^{1/2} \\ \tilde{A} &= \mathrm{diag}(OT\tilde{\pi})^{-1/2} OT \mathrm{diag}(\tilde{\pi})^{1/2} \end{aligned}$$

The first lemma shows that Ω can be decomposed into A and $\tilde{A}^\top$.

Lemma 5.1. $\Omega = A\tilde{A}^\top$.

Proof. It can be algebraically verified from the definition of B, u, v that $B = O\mathrm{diag}(\tilde{\pi})(OT)^\top$, $u = O\tilde{\pi}$, and $v = OT\tilde{\pi}$. Plugging in these expressions in Eq. (1), we have

$$\begin{aligned} \Omega &= \mathrm{diag}(O\tilde{\pi})^{-1/2} O \mathrm{diag}(\tilde{\pi})^{1/2} \\ &\quad \left(\mathrm{diag}(OT\tilde{\pi})^{-1/2} OT \mathrm{diag}(\tilde{\pi})^{1/2}\right)^\top = A\tilde{A}^\top. \quad \square \end{aligned}$$

The second lemma shows that A is in fact the desired matrix. The proof of this lemma crucially depends on the disjoint-cluster assumption of the Brown model.

Lemma 5.2. *$A = \sqrt{O}$ and $A^\top A = I_{m\times m}$.*

Proof. By Assumption 4.2, the x-th entry of $O\tilde{\pi}$ has value $O_{x,C(x)} \times \tilde{\pi}_{C(x)}$, and the $(x, C(x))$-th entry of $O\mathrm{diag}(\tilde{\pi})^{1/2}$ has value $O_{x,C(x)} \times \sqrt{\tilde{\pi}_{C(x)}}$. Thus the $(x, C(x))$-th entry of A is

$$A_{x,C(x)} = \frac{O_{x,C(x)}\sqrt{\tilde{\pi}_{C(x)}}}{\sqrt{O_{x,C(x)}\tilde{\pi}_{C(x)}}} = \sqrt{O_{x,C(x)}}$$

The columns of A have disjoint supports since A has the same sparsity pattern as O. Furthermore, the l_2 (Euclidean) norm of any column of A is the l_1 norm of the corresponding column of O. This implies $A^\top A = I_{m\times m}$ $\square$

Now we give a proof of the main theorem.

Proof of Theorem 5.1. The orthogonal projection matrix onto range(Ω) is given by $UU^\top$ and also by $\Omega(\Omega^\top\Omega)^+\Omega^\top$. Hence from Lemma 5.1 and 5.2, we have

$$\begin{aligned} UU^\top &= \Omega(\Omega^\top\Omega)^+\Omega^\top \\ &= (A\tilde{A}^\top)(\tilde{A}A^\top A\tilde{A}^\top)^+(A\tilde{A}^\top)^\top \\ &= (A\tilde{A}^\top)(\tilde{A}\tilde{A}^\top)^+(A\tilde{A}^\top)^\top = A\Pi A^\top \end{aligned}$$

where $\Pi = \tilde{A}(\tilde{A}^\top\tilde{A})^+\tilde{A}^\top$ is the orthogonal projection matrix onto range$(\tilde{A})$. But since $\tilde{A}$ has rank m, range$(\tilde{A}) = \mathbb{R}^m$ and thus $\Pi = I_{m\times m}$. Then we have $UU^\top = AA^\top$ where both U and A have orthogonal columns (Lemma 5.2). This implies that there is an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ such that $U = AQ^\top$. $\square$

5.3 ESTIMATION FROM SAMPLES

In Figure 4, we give an algorithm for computing an estimate of M from a sample of words $(x_1, \ldots, x_N) \in [n]^N$ (where M is described in Section 5.1). The algorithm estimates unigram and bigram word probabilities u, v, B to form a plug-in estimate $\hat{\Omega}$ of Ω (defined in Eq. (1)), computes a low-rank SVD of a sparse matrix, and normalizes the rows of the resulting left singular vector matrix.

The following theorem implies the consistency of our algorithm, assuming the consistency of $\hat{\Omega}$.

Theorem 5.2. *Let $\varepsilon := \|\hat{\Omega} - \Omega\|/\sigma_m(\Omega)$, where $\sigma_m(\Omega)$ is the m-th largest singular value of Ω. If $\varepsilon \le 0.07 \min_{x\in[n]}\{O_{x,C(x)}^{1/2}\}$, then the word embedding $f : x \mapsto \hat{M}_x$ (where $\hat{M}_x$ is the x-th row of $\hat{M}$) satisfies the following property: for all $x, x', x'' \in [n]$,*

$$\begin{aligned} C(x) = C(x') \neq C(x'') \\ \implies \|f(x) - f(x')\| < \|f(x) - f(x'')\|; \end{aligned}$$

Input: sequence of $N \geq 2$ words $(x_1, \ldots, x_N) \in [n]^N$; number of clusters m; smoothing parameter κ.
Output: matrix $\hat{M} \in \mathbb{R}^{n \times m}$ defining $f : x \mapsto \hat{M}_x \; \forall x \in [n]$.

1. Compute $\hat{B} \in \mathbb{R}^{n \times n}$, $\hat{u} \in \mathbb{R}^n$, and $\hat{v} \in \mathbb{R}^n$ where

$$\hat{B}_{x,x'} := \frac{1}{N-1} \sum_{i=1}^{N-1} [[x_i = x, x_{i+1} = x']] \qquad \forall x, x' \in [n]$$

$$\hat{u}_x := \frac{1}{N-1} \sum_{i=1}^{N-1} [[x_i = x]] + \frac{\kappa}{N-1} \qquad \forall x \in [n]$$

$$\hat{v}_x := \frac{1}{N-1} \sum_{i=1}^{N-1} [[x_{i+1} = x]] + \frac{\kappa}{N-1} \qquad \forall x \in [n]$$

2. Compute rank-m SVD of the sparse matrix

$$\hat{\Omega} := \text{diag}(\hat{u})^{-1/2} \; \hat{B} \; \text{diag}(\hat{v})^{-1/2}.$$

   Let $\hat{U} \in \mathbb{R}^{n \times m}$ be a matrix of m left singular vectors of $\hat{\Omega}$ corresponding to the m largest singular values.

3. Let $\hat{M}$ be the result of normalizing every row of $\hat{U}$ to have length 1.

Figure 4: Estimation of M from samples.

Input: vectors $\mu^{(1)}, \ldots, \mu^{(n)} \in \mathbb{R}^m$ corresponding to word types $[n]$ ordered in decreasing frequency.
Output: hierarchical clustering of the input vectors.
Tightening: Given a set of clusters $\mathcal{C}$, the subroutine tighten(c) for $c \in \mathcal{C}$ consists of the following three steps:

$$\text{nearest}(c) := \arg\min_{c' \in \mathcal{C} : c' \neq c} d(c, c')$$

$$\text{lowerbound}(c) := \min_{c' \in \mathcal{C} : c' \neq c} d(c, c')$$

$$\text{tight}(c) := \texttt{True}$$

Main body:

1. Initialize active clusters $\mathcal{C} = \{\{\mu^{(1)}\}, \ldots, \{\mu^{(m)}\}\}$ and call tighten(c) for all $c \in \mathcal{C}$.
2. For $i = m + 1$ to $n + m - 1$:
   (a) If $i \leq n$: let $c := \{\mu^{(i)}\}$, call tighten(c), and let $\mathcal{C} := \mathcal{C} \cup \{c\}$.
   (b) Let $c^* := \arg\min_{c \in \mathcal{C}} \text{lowerbound}(c)$.
   (c) While tight(c^*) is `False`,
      i. Call tighten(c^*).
      ii. Let $c^* := \arg\min_{c \in \mathcal{C}} \text{lowerbound}(c)$.
   (d) Merge c^* and nearest(c^*) in $\mathcal{C}$.
   (e) For each $c \in \mathcal{C}$: if nearest$(c) \in \{c^*, \text{nearest}(c^*)\}$, set tight$(c) :=$ `False`.

Figure 5: Variant of Ward's algorithm from Section 5.4.

(i.e., the embedding of any word x is closer to that of other words x' from the same cluster than it is to that of any word x'' from a different cluster).

The property established by Theorem 5.2 (proved in the appendix) allows many distance-based clustering algorithms to recover the correct clustering (e.g., single-linkage, average-linkage; see Balcan *et al.*, 2008). Moreover, it is possible to establish the finite sample complexity bounds for the estimation error of $\hat{\Omega}$ (and we do so for a simplified scenario in the (supplementary) Appendix C).

In practice, it is important to regularize the estimates $\hat{u}$ and $\hat{v}$ using a smoothing parameter $\kappa \geq 0$. This can be viewed as adding pseudocounts to alleviate the noise from infrequent words, and has a significant effect on the resulting representations. The practical importance of smoothing is also seen in previous methods using CCA (Cohen *et al.*, 2013; Hardoon *et al.*, 2004).

Another practical consideration is the use of richer context. So far, the context used for the token X_I is just the next token X_{I+1}; hence, the spectral estimation is based just on unigram and bigram probabilities. However, it is straightforward to generalize the technique to use other context—details are in the appendix. For instance, if we use the previous and next tokens (X_{I-1}, X_{I+1}) as context, then we form $\hat{\Omega} \in \mathbb{R}^{n \times 2n}$ from $\hat{B} \in \mathbb{R}^{n \times 2n}, \hat{u} \in \mathbb{R}^n, \hat{v} \in \mathbb{R}^{2n}$; however, we still extract $\hat{M} \in \mathbb{R}^{n \times m}$ from $\hat{\Omega}$ in the same way to form the word embedding.

5.4 AGGLOMERATIVE CLUSTERING

As established in Theorem 5.2, the word embedding obtained by mapping words to their corresponding rows of $\hat{M}$ permits distance-based clustering algorithms to recover the correct clustering. However, with small sample sizes and model approximation errors, the property from Theorem 5.2 may not hold exactly. Therefore, we propose to compute a hierarchical clustering of the word embeddings, with the goal of finding the correct clustering (or at least a good clustering) as some pruning of the resulting tree. Simple agglomerative clustering algorithms can provably recover the correct clusters when idealized properties (such as that from Theorem 5.2) hold (Balcan *et al.*, 2008), and can also be seen to be optimizing a sensible objective regardless (Dasgupta and Long, 2005). These algorithms also yield a hierarchy of word types—just as the original Brown clustering algorithm.

We use a form of average-linkage agglomerative clustering called Ward's algorithm (Ward Jr, 1963), which is particularly suited for hierarchical clustering in Euclidean spaces. In this algorithm, the cost of merging clusters c and c' is defined as

$$d(c, c') = \frac{|c||c'|}{|c| + |c'|} \; ||\mu_c - \mu_{c'}||^2 \tag{2}$$

where $|c|$ refers to the number of elements in cluster c and $\mu_c = |c|^{-1} \sum_{u \in c} u$ is the mean of cluster c. The algorithm starts with every point (word) in its own cluster, and repeat-

edly merges the two clusters with cheapest merge cost.

Figure 5 sketches a variant of Ward's algorithm that only considers merges among (at most) $m+1$ clusters at a time. The initial $m+1$ (singleton) clusters correspond to the $m+1$ most frequent words (according to $\hat{u}$); after a merge, the next most frequent word (if one exists) is used to initialize a new singleton cluster. This heuristic is also adopted by the original Brown algorithm, and is known to be very effective.

Using an implementation trick from Franti *et al.* (2000), the runtime of the algorithm is $O(\gamma nm^2)$, where γ is a data-dependent constant often much smaller than m, as opposed to $O(nm^3)$ in a naive implementation in which we search for the closest pair among $O(m^2)$ pairs at every merge.

The basic idea of Franti *et al.* (2000) is the following. For each cluster, we keep an estimation of the lower bound on the distance to the nearest cluster. We also track if this lower bound is tight; in the beginning, every bound is tight. When searching for the nearest pair, we simply look for a cluster with the smallest lower bound among m clusters instead of $O(m^2)$ cluster pairs. If the cluster has a tight lower bound, we merge it with its nearest cluster. Otherwise, we tighten its bound and again look for a cluster with the smallest bound. Thus γ is the effective number of searches at each iteration. At merge, the bound of a cluster whose nearest cluster is either of the two merged clusters becomes loose. We report empirical values of γ in our experimental study (see Table 3).

6 EXPERIMENTS

To evaluate the effectiveness of our approach, we used the clusters from our algorithm as additional features in supervised models for NER. We then compared the improvement in performance and also the time required to derive the clusters against those of the Brown clustering algorithm. Additionally, we examined the mutual information (MI) of the derived clusters on the training corpus:

$$\sum_{c,c'} \frac{\text{count}(c,c')}{N} \log \frac{\text{count}(c,c')N}{\text{count}(c)\text{count}(c')} \tag{3}$$

where N is the number of tokens in the corpus, $\text{count}(c)$ is the number of times cluster c appears, and $\text{count}(c,c')$ is the number of times clusters c, c' appear consecutively. Note that this is the quantity the Brown algorithm directly maximizes (Brown *et al.*, 1992).

6.1 EXPERIMENTAL SETTINGS

For NER experiments, we used the scripts provided by Turian *et al.* (2010). We used the greedy perceptron for NER experiments (Ratinov and Roth, 2009) using the standard features as our baseline models. We used the CoNLL 2003 dataset for NER with the standard train/dev/test split.

Table 1: Performance gains in NER.

	vocab	context	dev	test
Baseline	—	—	90.03	84.39
Spectral	50k	LR1	92	86.72
($\kappa = 200$)	300k	LR2	92.31	87.76
Brown	50k	—	92	88.56
	300k		92.68	88.76

Table 2: Mutual information computed as in Eq. (3) on the RCV1 corpus.

	vocab size	context	MI
Spectral	50k	LR2	1.48
($\kappa = 5000$)	300k	LR2	1.54
Brown	50k	—	1.52
	300k	—	1.6

For the choice of unlabeled text data, we used the Reuters-RCV1 corpus which contains 205 million tokens with 1.6 million distinct word types. To keep the size of the vocabulary manageable and also to reduce noise from infrequent words, we used only a selected number of the most frequent word types and replaced all other types in the corpus with a special token. For the size of the vocabulary, we used 50,000 and 300,000.

Our algorithm can be broken down into two stages: the SVD stage (Figure 4) and the clustering stage (Figure 5). In the SVD stage, we need to choose the number of clusters m and the smoothing parameter κ. As mentioned, we can easily define Ω to incorporate information beyond one word to the right. We experimented with the following configurations for context:

1. R1 ($\Omega \in \mathbb{R}^{n \times n}$): 1 word to the right. This is the version presented in Figure 4.
2. LR1 ($\Omega \in \mathbb{R}^{n \times 2n}$): 1 word to the left/right.
3. LR2 ($\Omega \in \mathbb{R}^{n \times 4n}$): 2 words to the left/right.

6.2 COMPARISON TO THE BROWN ALGORITHM: QUALITY

There are multiple ways to evaluate the quality of clusters. We considered the improvement in the F1 score in NER from using the clusters as additional features. We also examined the MI on the training corpus. For all experiments in this section, we used 1,000 clusters for both the spectral algorithm (i.e., $m = 1000$) and the Brown algorithm.

6.2.1 NER

In NER, there is significant improvement in the F1 score from using the clusters as additional features (Table 1).

Table 3: Speed and performance comparison with the Brown algorithm for different numbers of clusters and vocabulary sizes. In all the reported runtimes, we exclude the time to read and write data. We report the F1 scores on the NER dev set; for the spectral algorithm, we report the best scores.

m	vocab	Spectral runtime				Brown runtime	Ratio (%)	Spectral F1	Brown F1
		γ	SVD	cluster	total				
200	50k	3.35	4m24s	13s	4m37s	10m37s	43.48	91.53	90.79
400		5.17	6m39s	1m8s	7m47s	37m16s	20.89	91.73	91.21
600		9.80	5m29s	3m1s	8m30s	1h33m55s	9.05	91.68	91.79
800		12.64	9m26s	6m59s	16m25s	2h20m40s	11.67	91.81	91.83
1000		12.68	11m10s	10m25s	21m35s	3h37m	9.95	92.00	92.00
1000	300k	13.77	59m38s	1h4m37s	2h4m15s	22h19m37s	9.28	92.31	92.68

The dev F1 score is improved from 90.03 to 92 with either spectral or Brown clusters using 50k vocabulary size; it is improved to 92.31 with the spectral clusters and to 92.68 with the Brown clusters using 300k vocabulary size. The spectral clusters are a little behind the Brown clusters in the test set results. However, we remark that the well-known discrepancy between the dev set and the test set in the CoNLL 2003 dataset makes a conclusive interpretation difficult. For example, Turian *et al.* (2010) report that the F1 score using the embeddings of Collobert and Weston (2008) is higher than the F1 score using the Brown clusters on the dev set (92.46 vs 92.32) but lower on the test set (87.96 vs 88.52).

6.2.2 MI

Table 2 shows the MI computed as in Eq. (3) on the RCV1 corpus. The Brown algorithm optimizes the MI directly and generally achieves higher MI scores than the spectral algorithm. However, the spectral algorithm also achieves a surprisingly respectable level of MI scores even though the MI is not its objective. That is, the Brown algorithm specifically merges clusters in order to maximize the MI score in Eq. (3). In contrast, the spectral algorithm first recovers the model parameters using SVD and perform hierarchical clustering according to the parameter estimates, without any explicit concern for the MI score.

6.3 COMPARISON TO THE BROWN ALGORITHM: SPEED

To see the runtime difference between our algorithm and the Brown algorithm, we measured how long it takes to extract clusters from the RCV1 corpus for various numbers of clusters. In all the reported runtimes, we exclude the time to read and write data. We report results with 200, 400, 600, 800, and 1,000 clusters. All timing experiments were done on a machine with dual-socket, 8-core, 2.6GHz Intel Xeon E5-2670 (Sandy Bridge). The implementations for both algorithms were written in C++. The spectral algorithm also made use of Matlab for matrix calculations such as the SVD calculation.

Table 3 shows the runtimes required to extract these clusters as well as the F1 scores on the NER dev set obtained with these clusters. The spectral algorithm is considerably faster than the Brown algorithm while providing comparable improvement in the F1 scores. The runtime difference becomes more prominent as the number of clusters increases. Moreover, the spectral algorithm scales much better with larger vocabulary size. With 1,000 clusters and 300k vocabulary size, the Brown algorithm took over 22 hours whereas the spectral algorithm took 2 hours, 4 minutes, and 15 seconds—less than 10% of the time the Brown algorithm takes.

We also note that for the Brown algorithm, the improvement varies significantly depending on how many clusters are used; it is 0.76 with 200 clusters but 1.97 with 1,000 clusters. For the spectral algorithm, this seems to be less the case; the improvement is 1.5 with 200 clusters and 1.97 with 1,000 clusters.

6.3.1 Discussion on Runtimes

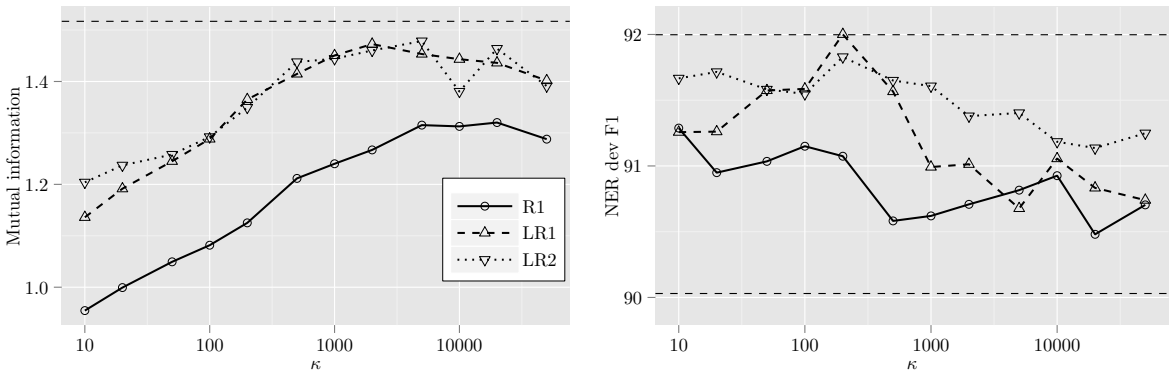
The final asymptotic runtime is $O(N+\gamma nm^2)$ for the spectral algorithm and $O(N + nm^2)$ for the Brown algorithm, where N is the size of the corpus, n is the number of distinct word types, m is the number of clusters, and γ is a data-dependent constant. Thus it may be puzzling why the spectral algorithm is significantly faster in practice. We explicitly discuss the issue in this section.

The spectral algorithm proceeds in two stages. First, it constructs a scaled covariance matrix in $O(N)$ time and performs a rank-m SVD of this matrix. Table 3 shows that SVD scales well with the value of m and the size of the corpus.

Second, the algorithm performs hierarchical clustering in $O(\gamma nm^2)$ time. This stage consists of $O(\gamma nm)$ calls to an $O(m)$ time function that computes Eq. (2), that is,

$$d(c, c') = \frac{|c||c'|}{|c| + |c'|} ||\mu_c - \mu_{c'}||^2$$

This function is quite simple: it calculates a scaled distance



(a) (b)

Figure 6: Effect of the choice of κ and context on (a) MI and (b) NER dev F1 score. We used 1,000 clusters on RCV1 with vocabulary size 50k. In (a), the horizontal line is the MI achieved by Brown clusters. In (b), the top horizontal line is the F1 score achieved with Brown clusters and the bottom horizontal line is the baseline F1 score achieved without using clusters.

$$
\begin{aligned}
&\texttt{computeL2usingOld}(s,t,u,v,w) = \texttt{L2}[v][w] \\
&- \texttt{q2}[v][s] - \texttt{q2}[s][v] - \texttt{q2}[w][s] - \texttt{q2}[s][w] \\
&- \texttt{q2}[v][t] - \texttt{q2}[t][v] - \texttt{q2}[w][t] - \texttt{q2}[t][w] \\
&+ (\texttt{p2}[v][s] + \texttt{p2}[w][s]) * \log((\texttt{p2}[v][s] + \texttt{p2}[w][s])/((\texttt{p1}[v] + \texttt{p1}[w]) * \texttt{p1}[s])) \\
&+ (\texttt{p2}[s][v] + \texttt{p2}[s][w]) * \log((\texttt{p2}[s][v] + \texttt{p2}[s][w])/((\texttt{p1}[v] + \texttt{p1}[w]) * \texttt{p1}[s])) \\
&+ (\texttt{p2}[v][t] + \texttt{p2}[w][t]) * \log((\texttt{p2}[v][t] + \texttt{p2}[w][t])/((\texttt{p1}[v] + \texttt{p1}[w]) * \texttt{p1}[t])) \\
&+ (\texttt{p2}[t][v] + \texttt{p2}[t][w]) * \log((\texttt{p2}[t][v] + \texttt{p2}[t][w])/((\texttt{p1}[v] + \texttt{p1}[w]) * \texttt{p1}[t])) \\
&+ \texttt{q2}[v][u] + \texttt{q2}[u][v] + \texttt{q2}[w][u] + \texttt{q2}[u][w] \\
&- (\texttt{p2}[v][u] + \texttt{p2}[w][u]) * \log((\texttt{p2}[v][u] + \texttt{p2}[w][u])/((\texttt{p1}[v] + \texttt{p1}[w]) * \texttt{p1}[u])) \\
&- (\texttt{p2}[u][v] + \texttt{p2}[u][w]) * \log((\texttt{p2}[u][v] + \texttt{p2}[u][w])/((\texttt{p1}[v] + \texttt{p1}[w]) * \texttt{p1}[u]))
\end{aligned}
$$

Figure 7: A $O(1)$ function that is called $O(nm^2)$ times in Liang's implementation of the Brown algorithm, accounting for over 40% of the runtime. Similar functions account for the vast majority of the runtime. The values in the arrays $\texttt{L2}, \texttt{q2}, \texttt{p2}, \texttt{p1}$ are precomputed. $\texttt{p2}[v][w] = p(v,w)$, i.e, the probability of cluster v being followed by cluster w; $\texttt{p1}[v] = p(v)$ is the probability of cluster v; $\texttt{q2}[v][w] = p(v,w)\log((p(v)p(w))^{-1}p(v,w))$ is the contribution of the mutual information between clusters v and w. The function recomputes $\texttt{L2}[v][w]$, which is the loss in log-likelihood if clusters v and w are merged. The function updates $\texttt{L2}$ after clusters s and t have been merged to form a new cluster u. There are many operations involved in this calculation: 6 divisions, 12 multiplications, 36 additions (26 additions and 10 subtractions), and 6 $\log$ operations.

between two vectors in $\mathbb{R}^m$. Moreover, it avails itself readily to existing optimization techniques such as vectorization.[1] Finally, we found that the empirical value of γ was typically small: it ranged from 3.35 to 13.77 in our experiments reported in Table 3 (higher m required higher γ).

In contrast, while the main body of the Brown algorithm requires $O(N + nm^2)$ time, the constant factors are high due to fairly complex book-keeping that is required. For example, the function in Figure 7 (obtained from Liang's implementation) is invoked $O(nm^2)$ times in total: specifically, whenever two clusters s and t are merged to form a new cluster u (this happens $O(n)$ times), the function is called $O(m^2)$ times, for all pairs of clusters v, w such that v and w are not equal to s, t, or u. The function recomputes the loss in likelihood if clusters v and w are merged, after s and t are merged to form u. It requires a relatively large number of arithmetic operations, leading to high constant factors. Calls to this function alone take over 40% of the runtime for the Brown algorithm; similar functions account for the vast majority of the algorithm's runtime. It is not clear that this overhead can be reduced.

[1]Many linear algebra libraries automatically support vectorization. For instance, the `Eigen` library in our implementation enables vectorization by default, which gave a 2-3 time speedup in our experiments.

6.4 EFFECT OF THE CHOICE OF κ AND CONTEXT

Figure 6 shows the MI and the F1 score on the NER dev set for various choices of κ and context. For NER, around 100-200 for the value of κ gives good performance. For the MI, the value of κ needs to be much larger.

LR1 and LR2 perform much better than R1 but are very similar to each other across the results, suggesting that words in the immediate vicinity are necessary and nearly sufficient for these tasks.

7 CONCLUSION

In this paper, we have presented a new and faster alternative to the Brown clustering algorithm. Our algorithm has a provable guarantee of recovering the underlying model parameters. This approach first uses SVD to consistently estimate low-dimensional representations of word types that reveal their originating clusters by exploiting the implicit disjoint-cluster assumption of the Brown model. Then agglomerative clustering is performed over these representations to build a hierarchy of word types. The resulting clusters give a competitive level of improvement in performance in NER as the clusters from the Brown algorithm,

but the spectral algorithm is significantly faster.

There are several areas for the future work. One can try to speed up the algorithm even more via a top-down rather than bottom-up approach for hierarchical clustering, for example recursively running the 2-means algorithm. Experiments with the clusters in tasks other than NER (e.g., dependency parsing), as well as larger-scale experiments, can help further verify the quality of the clusters and highlight the difference between the spectral algorithm and the Brown algorithm.

Acknowledgments

We thank Dean Foster, Matus Telgarsky, and Terry Koo for helpful discussions. This work was made possible by a research grant from Bloomberg's Knowledge Engineering team. Kim was also supported by a Samsung Scholarship.

A INCORPORATING RICHER CONTEXT

We assume a function ϕ such that for $i \in [N]$, it returns a set of positions other than i. For example, we may define $\phi(i) = \{i-1, i+1\}$ to look at one position to the left and to the right. Let $s = |\phi(i)|$ and enumerate the elements of $\phi(i)$ as $j_1, \ldots, j_s$. Define $B^{(j)} \in \mathbb{R}^{n \times n}, v^{(j)} \in \mathbb{R}^n$ for all $j \in \phi(i)$ as follows:

$$B^{(j)}_{x,x'} = P(X_i = x, X_j = x') \qquad \forall x, x' \in [n]$$
$$v^{(j)}_x = P(X_j = x) \qquad \forall x \in [n]$$

The new definitions of $B \in \mathbb{R}^{n \times ns}, v \in \mathbb{R}^{ns}$ are given by $B = [B^{(j_1)}, \ldots, B^{(j_s)}]$ and $v = [(v^{(j_1)})^\top, \ldots, (v^{(j_s)})^\top]^\top$. Letting $\Omega \in \mathbb{R}^{n \times ns}$ as in Eq. (1), it is easy to verify Theorem 5.1 using similar techniques.

B PROOF OF THEOREM 5.2

Write the rank-m SVD of Ω as $\Omega = USV^\top$, and similarly write the rank-m SVD of $\hat{\Omega}$ as $\hat{U}\hat{S}\hat{V}^\top$. Since Ω has rank m, it follows by Eckart-Young that

$$\|\hat{U}\hat{S}\hat{V}^\top - \hat{\Omega}\| \leq \|\Omega - \hat{\Omega}\|.$$

Therefore, by the triangle inequality,

$$\|\hat{U}\hat{S}\hat{V}^\top - USV^\top\| \leq 2\|\Omega - \hat{\Omega}\| = 2\varepsilon\sigma_m(\Omega).$$

This implies, via applications of Wedin's theorem and Weyl's inequality,

$$\|U_\perp^\top \hat{U}\| \leq 2\varepsilon \quad \text{and} \quad \|\hat{U}_\perp^\top U\| \leq \frac{2\varepsilon}{1-2\varepsilon} \tag{4}$$

where $U_\perp \in \mathbb{R}^{n \times (n-m)}$ is a matrix whose columns form an orthonormal basis for the orthogonal complement of the range of U, and $\hat{U}_\perp \in \mathbb{R}^{n \times (n-m)}$ is similarly defined (and note that $\varepsilon < 1/2$ by assumption).

Recall that by Theorem 5.1, there exists an orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ such that $U = \sqrt{O}Q^\top$. Define $\hat{Q} := \hat{U}^\top\sqrt{O} = \hat{U}^\top UQ$, and, for all $c \in [m]$, $\hat{q}_c := \hat{Q}e_c$. The fact that $\|UQe_c\| = 1$ implies

$$\|\hat{q}_c\| = \sqrt{1 - \|\hat{U}_\perp\hat{U}_\perp^\top UQe_c\|^2} \leq 1.$$

Therefore, by Eq. (4),

$$1 \geq \|\hat{q}_c\| \geq \|\hat{q}_c\|^2 \geq 1 - \left(\frac{2\varepsilon}{1-2\varepsilon}\right)^2. \tag{5}$$

We also have, for $c \neq c'$,

$$\hat{q}_c^\top \hat{q}_{c'} \leq \|\hat{U}_\perp^\top UQe_c\|\|\hat{U}_\perp^\top UQe_{c'}\| \leq \left(\frac{2\varepsilon}{1-2\varepsilon}\right)^2, \tag{6}$$

where the first inequality follows by Cauchy-Schwarz, and the second inequality follows from (4). Therefore, by Eq. (5) and Eq. (6), we have for $c \neq c'$,

$$\|\hat{q}_c - \hat{q}_{c'}\|^2 \geq 2\left(1 - 2\left(\frac{2\varepsilon}{1-2\varepsilon}\right)^2\right). \tag{7}$$

Let $\bar{o}_x := O^{1/2}_{x,C(x)}$. Recall that $\sqrt{O}^\top e_x = \bar{o}_x e_{C(x)} \in \mathbb{R}^m$, so $\hat{Q}\sqrt{O}^\top e_x = \bar{o}_x \hat{q}_{C(x)}$ and $\|\hat{Q}\sqrt{O}^\top e_x\| = \bar{o}_x\|q_{C(x)}\|$. By the definition of $\hat{Q}$, we have

$$\hat{U} - \sqrt{O}\hat{Q}^\top = \hat{U} - UU^\top\hat{U} = U_\perp U_\perp^\top \hat{U}$$

This implies, for any $x \in [n]$,

$$\begin{aligned}\|\hat{U}^\top e_x - \bar{o}_x\hat{q}_{C(x)}\| &= \|(\hat{U} - \sqrt{O}\hat{Q}^\top)^\top e_x\| \\ &= \|\hat{U}^\top U_\perp U_\perp^\top e_x\| \leq 2\varepsilon \end{aligned} \tag{8}$$

by Eq. (4). Moreover, by the triangle inequality,

$$|\|\hat{U}^\top e_x\| - \bar{o}_x\|q_{C(x)}\|| \leq 2\varepsilon. \tag{9}$$

Since $\hat{M}^\top e_x = \|\hat{U}^\top e_x\|^{-1}\hat{U}^\top e_x$, we have

$$\begin{aligned}\|\hat{M}^\top e_x - \hat{q}_{C(x)}\| &= \left\|\frac{1}{\|\hat{U}^\top e_x\|}\hat{U}^\top e_x - \hat{q}_{C(x)}\right\| \\ &\leq \frac{1}{\bar{o}_x}\|\hat{U}^\top e_x - \bar{o}_x\hat{q}_{C(x)}\| + |1 - \|\hat{q}_{C(x)}\|| \\ &\quad + \frac{|\bar{o}_x\|\hat{q}_{C(x)}\| - \|\hat{U}^\top e_x\||}{\bar{o}_x} \\ &\leq \frac{4\varepsilon}{\bar{o}_x} + \left(\frac{2\varepsilon}{1-2\varepsilon}\right)^2, \end{aligned} \tag{10}$$

where the first inequality follow by the triangle inequality and norm homogeneity, and the second inequality uses Eq. (8), Eq. (9), and Eq. (5). Using Eq. (10), we may upper bound the distance $\|\hat{M}^\top e_x - \hat{M}^\top e_{x'}\|$ when $C(x) = C(x')$; using Eq. (7) and Eq. (10), we may lower bound the distance $\|\hat{M}^\top e_x - \hat{M}^\top e_{x''}\|$ when $C(x) \neq C(x'')$. The theorem then follows by invoking the assumption on ε.

References

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

Balcan, M.-F., Blum, A., and Vempala, S. (2008). A discriminative framework for clustering via similarity functions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 671–680.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, **18**(4), 467–479.

Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. (2013). Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Dasgupta, S. and Long, P. M. (2005). Performance guarantees for hierarchical clustering. *J. Comput. Syst. Sci.*, **70**(4), 555–569.

Dhillon, P. S., Foster, D., and Ungar, L. (2011). Multi-view learning of word embeddings via CCA. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24.

Franti, P., Kaukoranta, T., Shen, D.-F., and Chang, K.-S. (2000). Fast and memory efficient implementation of the exact pnn. *Image Processing, IEEE Transactions on*, **9**(5), 773–777.

Gao, J., Goodman, J., Miao, J., *et al.* (2001). The use of clustering techniques for language modeling–application to asian languages. *Computational Linguistics and Chinese Language Processing*, **6**(1), 27–60.

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

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, **28**(3/4), 321–377.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, **78**(5), 1460–1480.

Kneser, R. and Ney, H. (1993). Improved clustering techniques for class-based statistical language modelling. In *Third European Conference on Speech Communication and Technology*.

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Liang, P. (2005). *Semi-Supervised Learning for Natural Language*. Master's thesis, Massachusetts Institute of Technology.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751.

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342. Citeseer.

Ng, A. Y., Jordan, M. I., Weiss, Y., *et al.* (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, **2**, 849–856.

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Shanbehzadeh, J. and Ogunbona, P. (1997). On the computational complexity of the lbg and pnn algorithms. *Image Processing, IEEE Transactions on*, **6**(4), 614–616.

Tropp, J. A. (2011). User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, pages 1–46.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, **58**(301), 236–244.