Latent-Descriptor Clustering for Unsupervised POS Induction

Michael Lamar
Department of Mathematics and Computer Science
Saint Louis University
220 N. Grand Blvd.
St.Louis, MO 63103, USA
mlamar@slu.edu

Yariv Maron
Gonda Brain Research Center
Bar-Ilan University
Ramat-Gan 52900, Israel
syarivm@yahoo.com

Elie Bienenstock
Division of Applied Mathematics
and Department of Neuroscience
Brown University
Providence, RI 02912, USA
elie@brown.edu

Abstract

We present a novel approach to distributional-only, fully unsupervised, POS tagging, based on an adaptation of the EM algorithm for the estimation of a Gaussian mixture. In this approach, which we call Latent-Descriptor Clustering (LDC), word types are clustered using a series of progressively more informative descriptor vectors. These descriptors, which are computed from the immediate left and right context of each word in the corpus, are updated based on the previous state of the cluster assignments. The LDC algorithm is simple and intuitive. Using standard evaluation criteria for unsupervised POS tagging, LDC shows a substantial improvement in performance over state-of-the-art methods, along with a several-fold reduction in computational cost.

1 Introduction

Part-of-speech (POS) tagging is a fundamental natural-language-processing problem, and POS tags are used as input to many important applications. While state-of-the-art supervised POS taggers are more than 97% accurate (Toutanova et al., 2003; Tsuruoka and Tsujii, 2005), unsupervised POS taggers continue to lag far behind. Several authors addressed this gap using limited supervision, such as a dictionary of tags for each word (Goldwater and Griffiths, 2007; Ravi and Knight, 2009), or a list of word prototypes for each tag (Haghighi and Klein, 2006). Even in light of all these advancements, there is still interest in a completely unsupervised method for POS induction for several reasons. First, most languages do not have a tag dictionary. Second, the preparation of such resources is error-prone. Third, while several widely used tag sets do exist, researchers do not agree upon any specific set of tags across languages or even within one language. Since tags are used as basic features for many important NLP applications (e.g. Headden et al. 2008), exploring new, statistically motivated, tag sets may also be useful. For these reasons, a fully unsupervised induction algorithm has both a practical and a theoretical value.

In the past decade, there has been a steady improvement on the completely unsupervised version of POS induction (Schütze, 1995; Clark, 2001; Clark, 2003; Johnson, 2007; Gao and Johnson, 2008; Graça et al., 2009; Abend et al., 2010; Lamar et al., 2010; Reichart et al., 2010; Berg-Kirkpatrick et al., 2010). Some of these methods use morphological cues (Clark, 2001; Clark, 2003; Abend et al., 2010; Reichart et al., 2010; Berg-Kirkpatrick et al., 2010), but all rely heavily on distributional information, *i.e.*, bi-

gram statistics. Two recent papers advocate *non-disambiguating* models (Abend et al., 2010; Lamar et al., 2010): these assign the same tag to all tokens of a word type, rather than attempting to disambiguate words in context. Lamar et al. (2010) motivate this choice by showing how removing the disambiguation ability from a state-of-the-art disambiguating model results in increasing its accuracy.

In this paper, we present a novel approach to non-disambiguating, distributional-only, fully unsupervised, POS tagging. As in all non-disambiguating distributional approaches, the goal, loosely stated, is to assign the same tag to words whose contexts in the corpus are similar. Our approach, which we call Latent-Descriptor Clustering, or LDC, is an iterative algorithm, in the spirit of the *K*-means clustering algorithm and of the EM algorithm for the estimation of a mixture of Gaussians.

In conventional *K*-means clustering, one is given a collection of N objects described as N data points in an r-dimensional Euclidean space, and one seeks a clustering that minimizes the sum of intra-cluster squared distances, *i.e.*, the sum, over all data points, of the squared distance between that point and the centroid of its assigned cluster. In LDC, we similarly state our goal as one of finding a tagging, *i.e.*, cluster assignment, A, that minimizes the sum of intra-cluster squared distances. However, unlike in conventional *K*-means, the N objects to be clustered are themselves described by vectors—in a suitable manifold—that depend on the clustering A. We call these vectors *latent descriptors*.

Specifically, each object to be clustered, *i.e.*, each word type w, is described in terms of its *left-tag context* and *right-tag context*. These context vectors are the counts of the K different tags occurring, under tagging A, to the left and right of tokens of word type w in the corpus. We normalize each of these context vectors to unit length, producing, for each word type w, two points $L_A(w)$ and $R_A(w)$ on the $(K–1)$-dimensional unit sphere. The latent descriptor for w consists of the pair $(L_A(w), R_A(w))$—more details in Section 2.

A straightforward approach to this latent-descriptor *K*-means problem is to adapt the classical iterative *K*-means algorithm so as to handle the latent descriptors. Specifically, in each iteration, given the assignment A obtained from the previous iteration, one first computes the latent descriptors for all word types as defined above, and then proceeds in the usual way to update cluster centroids and to find a new assignment A to be used in the next iteration.

For reasons to be discussed in Section 5, we instead prefer a *soft-assignment* strategy, inspired from the EM algorithm for the estimation of a mixture of Gaussians. Thus, rather than the hard assignment A, we use a soft-assignment matrix P. P_{wk}, interpreted as the probability of assigning word w to cluster k, is, essentially, proportional to $\exp\{-d_{wk}^2/2\sigma^2\}$, where d_{wk} is the distance between the latent descriptor for w and the centroid, *i.e.*, Gaussian mean, for k. Unlike the Gaussian-mixture model however, we use the same mixture coefficient and the same Gaussian width for all k. Further, we let the Gaussian width σ decrease gradually during the iterative process. As is well-known, the EM algorithm for Gaussian mixtures reduces in the limit of small σ to the simpler *K*-means clustering algorithm. As a result, the last few iterations of LDC effectively implement the hard-assignment *K*-means-style algorithm outlined in the previous paragraph. The soft assignment used earlier in the process lends robustness to the algorithm.

The LDC approach is shown to yield substantial improvement over state-of-the-art methods for the problem of fully unsupervised, distributional only, POS tagging. The algorithm is conceptually simple and easy to implement, requiring less than 30 lines of Matlab code. It runs in a few seconds of computation time, as opposed to hours or days for the training of HMMs.

2 Notations and Statement of Problem

The LDC algorithm is best understood in the context of the latent-descriptor *K*-means optimization problem. In this section, we set up our notations and define this problem in detail. For simplicity, induced tags are henceforth referred to as *labels*, while *tags* will be reserved for the gold-standard tags, to be used later for evaluation.

Let W denote the set of word types $w_1,\ldots,w_N$, and let T denote the set of labels, *i.e.*, induced

tags. The sizes of these sets are $|W| = N$ and $|T| = K$. In the experiments presented in Section 4, N is 43,766 and K is either 50 or 17. For any word token t in the corpus, we denote the word type of t by $w(t)$. The frequency of word type w in the corpus is denoted $f(w)$; thus, $\sum_w f(w) = 1$.

For a word type w_1, the *left-word context* of w_1, $L(w_1)$, is defined as the N-dimensional vector whose n-th component is the number of bigrams, *i.e.*, pairs of consecutive tokens (t_{i-1}, t_i) in the corpus, such that $w(t_i) = w_1$ and $w(t_{i-1}) = n$. Similarly, we define the *right-word context* of w_1, $R(w_1)$, as the N-dimensional vector whose n-th component is the number of bigrams (t_i, t_{i+1}) such that $w(t_i) = w_1$ and $w(t_{i+1}) = n$. We let L (resp. R) be the $N \times N$ matrix whose w-th row is $L(w)$ (resp. $R(w)$).

S^{K-1} is the unit sphere in the K-dimensional Euclidean space $\mathbb{R}^K$. For any $x \in \mathbb{R}^K$, we denote by $\lambda(x)$ the projection of x on S^{K-1}, *i.e.*, $\lambda(x) = x/||x||$.

A labeling is a map $A: W \to T$. Given a labeling A, we define $\widetilde{L}_A(w_1)$, the *left-label context* of word type w_1, as the K-dimensional vector whose k-th component is the number of bigrams (t_{i-1}, t_i) in the corpus such that $w(t_i) = w_1$ and $A(w(t_{i-1})) = k$. We define the *left descriptor* of word type w as:

$$L_A(w) = \lambda(\widetilde{L}_A(w)).$$

We similarly define the right-label context of w_1, $\widetilde{R}_A(w_1)$, as the K-dimensional vector whose k-th component is the number of bigrams (t_i, t_{i+1}) such that $w(t_i) = w_1$ and $A(w(t_{i+1})) = k$, and we define the *right descriptor* of word type w as:

$$R_A(w) = \lambda(\widetilde{R}_A(w)).$$

In short, any labeling A defines two maps, L_A and R_A, each from W to S^{K-1}.

For any function $g(w)$ defined on W, $<g(w)>$ will be used to denote the average of $g(w)$ weighted by the frequency of word type w in the corpus:

$$<g(w)> = \sum_w f(w)g(w).$$

For any label k, we define:

$$C_L(k) = \lambda(< L_A(w): A(w) = k >).$$

Thus, $C_L(k)$ is the projection on S^{K-1} of the weighted average of the left descriptors of the word types labeled k. We sometimes refer to $C_L(k)$ as the left centroid of cluster k. Note that $C_L(k)$ depends on A in two ways, first in that the average is taken on words w such that $A(w) = k$, and second through the global dependency of L_A on A. We similarly define the right centroids:

$$C_R(k) = \lambda(< R_A(w): A(w) = k >).$$

Informally, we seek a labeling A such that, for any two word types w_1 and w_2 in W, w_1 and w_2 are labeled the same if and only if $L_A(w_1)$ and $L_A(w_2)$ are close to each other on S^{K-1} and so are $R_A(w_1)$ and $R_A(w_2)$. Formally, our goal is to find a labeling A that minimizes the objective function:

$$F(A) = <||L_A(w) - C_L(A(w))||^2 + ||R_A(w) - C_R(A(w))||^2>.$$

Note that, just as in conventional K-means clustering, $F(A)$ is the sum of the intra-cluster squared distances. However, unlike conventional K-means clustering, the descriptors of the objects to be clustered depend themselves on the clustering. We accordingly refer to L_A and R_A as *latent descriptors*, and to the method described in the next section as **Latent-Descriptor Clustering**, or LDC.

Note, finally, that we do not seek the *global* minimum of $F(A)$. This global minimum, 0, is obtained by the trivial assignment that maps all word types to a unique label. Instead, we seek a minimum under the constraint that the labeling be non-trivial. As we shall see, this constraint need not be imposed explicitly: the iterative LDC algorithm, when suitably initialized and parameterized, converges to non-trivial local minima of $F(A)$—and these are shown to provide excellent taggers.

3 Methods

Recall that a mixture of Gaussians is a generative model for a random variable taking values

in a Euclidean space $\mathbb{R}^r$. With K Gaussians, the model is parameterized by:

- K mixture parameters, *i.e.*, K non-negative numbers adding up to 1;
- K means, *i.e.*, K points $\mu_1,\ldots,\mu_K$ in $\mathbb{R}^r$;
- K variance-covariance $d\times d$ matrices.

The collection of all parameters defining the model is denoted by θ. EM is an iterative algorithm used to find a (local) maximizer of the likelihood of N observed data points $x_1,\ldots,x_N \in \mathbb{R}^r$. Each iteration of the algorithm includes an E phase and an M phase. The E phase consists of computing, based on the current θ, a probabilistic assignment of each of the N observations to the K Gaussian distributions. These probabilistic assignments form an $N\times K$ stochastic matrix P, *i.e.*, a matrix of non-negative numbers in which each row sums to 1. The M phase consists of updating the model parameters θ, based on the current assignments P. For more details, see, e.g., Bishop (2006).

The structure of the LDC algorithm is very similar to that of the EM algorithm. Thus, each iteration of LDC consists of an E phase and an M phase. As observations are replaced by latent descriptors, an iteration of LDC is best viewed as starting with the M phase. The M phase first starts by building a pair of latent-descriptor matrices L_P and R_P, from the soft assignments obtained in the previous iteration. Note that these descriptors are now indexed by P, the matrix of probabilistic assignments, rather than by hard assignments A as in the previous section.

L_P and R_P are obtained by a straightforward adaptation of the definition given in the previous section to the case of probabilistic assignments. Thus, the latent descriptors consist of the left-word and right-word contexts (recall that these are given by matrices L and R), mapped into left-label and right-label contexts through multiplication by the assignment matrix P, and scaled to unit length:

$$L_P = \lambda(LP)$$
$$R_P = \lambda(RP).$$

With these latent descriptors in hand, we proceed with the M phase of the algorithm as usual. Thus, the left mean $\mu^L{}_k$ for Gaussian k is the weighted average of the left latent descriptors $L_P(w)$, scaled to unit length. The weight used in this weighted average is $P_{wk}\times f(w)$ (remember that $f(w)$ is the frequency of word type w in the corpus). Note that the definition of the Gaussian mean $\mu^L{}_k$ parallels the definition of the cluster centroid $C_L(k)$ given in the previous section; if the assignment P happens to be a hard assignment, $\mu^L{}_k$ is actually identical to $C_L(k)$. The right Gaussian mean $\mu^R{}_k$ is computed in a similar fashion. As mentioned, we do not estimate any mixture coefficients or variance-covariance matrices.

The E phase of the iteration takes the latent descriptors and the Gaussian means, and computes a new $N\times K$ matrix of probabilistic assignments P. These new assignments are given by:

$$P_{wk} = \frac{1}{Z}\exp\{-[\| L_P(w) - \mu_k^L \|^2 + \| R_P(w) - \mu_k^R \|^2]/2\sigma^2\}$$

with Z a normalization constant such that $\sum_k P_{wk} = 1$. σ is a parameter of the model, which, as mentioned, is gradually decreased to enforce convergence of P to a hard assignment.

The description of the M phase given above does not apply to the first iteration, since the M phase uses P from the previous iteration. To initialize the algorithm, *i.e.*, create a set of left and right descriptor vectors in the M phase of the first iteration, we use the left-word and right-word contexts L and R. These matrices however are of very high dimension ($N\times N$), and thus sparse and noisy. We therefore reduce their dimensionality, using reduced-rank singular-value decomposition. This yields two $N\times r_1$ matrices, L_1 and R_1. A natural choice for r_1 is $r_1 = K$, and this was indeed used for $K = 17$. For $K = 50$, we also use $r_1 = 17$. The left and right descriptors for the first iteration are obtained by scaling each row of matrices L_1 and R_1 to unit length. The Gaussian centers $\mu^L{}_k$ and $\mu^R{}_k$, $k = 1,\ldots,K$, are set equal to the left and right descriptors of the K

most frequent words in the corpus. This completes the description of the LDC algorithm.[1]

While this algorithm is intuitive and simple, it does not easily lend itself to mathematical analysis; indeed there is no *a priori* guarantee that it will behave as desired. Even for the simpler, hard-assignment, K-means-style version of LDC outlined in the previous section, there is no equivalent to the statement—valid for the conventional K-means algorithm—that each iteration lowers the intra-cluster sum of squared distances $F(A)$; this is a mere consequence of the fact that the descriptors themselves are updated on each iteration. The soft-assignment version of LDC does not directly attempt to minimize $F(A)$, nor can it be viewed as likelihood maximization—as is EM for a Gaussian mixture—since the use of latent descriptors precludes the definition of a generative model for the data. This theoretical difficulty is compounded by the use of a variable σ.

Empirically however, as shown in the next section, we find that the LDC algorithm is very well behaved. Two simple tools will be used to aid in the description of the behavior of LDC.

The first tool is an objective function $G(P)$ that parallels the definition of $F(A)$ for hard assignments. For a probabilistic assignment P, we define $G(P)$ to be the weighted average, over all w and all k, of $\|L_P(w) - \mu^L_k\|^2 + \|R_P(w) - \mu^R_k\|^2$; the weight used in this average is $P_{wk} \times f(w)$, just as in the computation of the Gaussian means. Clearly, G is identical to F on any P that happens to be a hard assignment. Thus, G is actually an extension of the objective function F to soft assignments.

The second tool will allow us to compute a tagging accuracy for soft assignments. For this purpose, we simply create, for any probabilistic assignment P, the obvious labeling $A = A^*(P)$ that maps w to k with highest P_{wk}.

4 Results

In order to evaluate the performance of LDC, we apply it to the *Wall Street Journal* portion of the Penn Treebank corpus (1,173,766 tokens, all lower-case, resulting in $N = 43{,}766$ word types). We compare the induced labels with two gold-standard tagsets:

- **PTB45**, the standard 45-tag PTB tagset. When using PTB45 as the gold standard, models induce 50 labels, to allow comparison with Gao and Johnson (2008) and Lamar et al. (2010).

- **PTB17**, the PTB tagset coarse-grained to 17 tags (Smith and Eisner 2005). When using PTB17 as the gold standard, models induce 17 labels.

In order to compare the labels generated by the unsupervised model with the tags of each tagset, we use two map-based criteria:

- **MTO**: many-to-one tagging accuracy, *i.e.*, fraction of correctly-tagged tokens in the corpus under the so-called many-to-one mapping, which takes each induced tag to the gold-standard POS tag with which it co-occurs most frequently. This is the most prevalent metric in use for unsupervised POS tagging, and we find it the most reliable of all criteria currently in use. Accordingly, the study presented here emphasizes the use of MTO.

- **OTO**: best tagging accuracy achievable under a so-called one-to-one mapping, *i.e.*, a mapping such that at most one induced tag is sent to any POS tag. The optimal one-to-one mapping is found through the Hungarian algorithm[2].

[1] The LDC code, including tagging accuracy evaluation, is available at http://www.dam.brown.edu/people/elie/code/.

[2] Code by Markus Beuhren is available at http://www.mathworks.com/matlabcentral/fileexchange/6543-functions-for-the-rectangular-assignment-problem

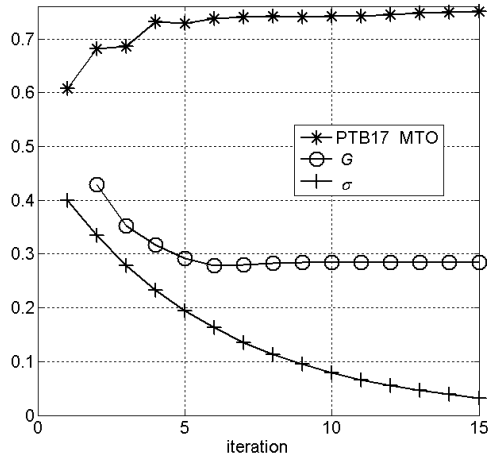

Figure 1: Convergence of LDC with $K = 17$. Bottom curve: σ -schedule, *i.e.*, sequence of Gaussian widths employed. Middle curve: Objective function $G(P)$ (see Section 3). Top curve: Many-to-one tagging accuracy of labeling $A^*(P)$, evaluated against PTB17.

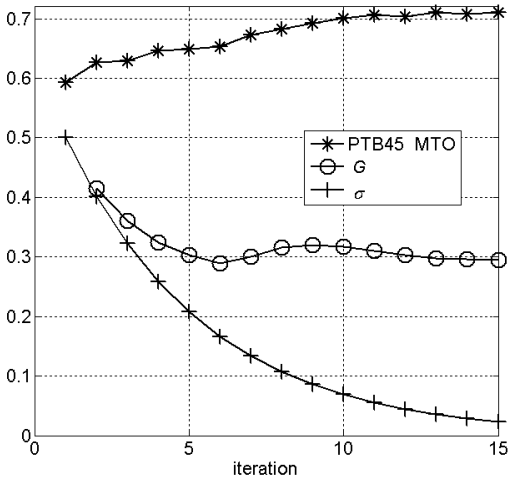

Figure 2: Same as Figure 1 but with $K = 50$. Top curve shows the MTO accuracy of the labeling evaluated against PTB45.

Figures 1 and 2 show the behavior of the LDC algorithm for $K = 17$ and $K = 50$ respectively. From the G curves as well as from the MTO scoring curves (using the labeling $A^*(P)$ defined at the end of Section 3), it is clear that the algorithm converges. The figures show only the first 15 iterations, as little change is observed after that. The schedule of the σ parameter was given the simple form $\sigma(t) = \sigma_1 \exp\{-c(t-1)\}$, $t = 1,2,\ldots$, and the parameters σ_1 and c were adjusted so as to get the best MTO accuracy. With the σ-schedules used in these experiments, P typically converges to a hard assignment in about 45 iterations, σ being then 10^{-5}.

While the objective function $G(P)$ mostly decreases, it does show a hump for $K = 50$ around iteration 9. This may be due to the use of latent descriptors, or of a variable σ, or both. The MTO score sometimes decreases by a small fraction of a percent, after having reached its peak around the 15th iteration.

Note that we start σ at 0.4 for $K = 17$, and at 0.5 for $K = 50$. Although we chose two slightly different σ schedules for the two tagsets in order to achieve optimal performance on each tagset, an identical sequence of σ can be used for both with only a 1% drop in PTB17 score.

As the width of the Gaussians narrows, each vector is steadily pushed toward a single choice of cluster. This forced choice, in turn, produces more coherent descriptor vectors for all word types, and yields a steady increase in tagging accuracy.

Criterion	Model	PTB17	PTB45
MTO	LDC	**0.751**	**0.708**
	SVD2	0.740	0.658
	HMM-EM	0.647	0.621
	HMM-VB	0.637	0.605
	HMM-GS	0.674	0.660
OTO	LDC	**0.593**	0.483
	SVD2	0.541	0.473
	HMM-EM	0.431	0.405
	HMM-VB	0.514	0.461
	HMM-GS	0.466	**0.499**

Table 1. Tagging accuracy comparison between several models for two tagsets and two mapping criteria. Note that LDC significantly outperforms all HMMs (Gao and Johnson, 2008) in every case except PTB45 under the OTO mapping. LDC also outperforms SVD2 (Lamar et al., 2010) in all cases.

Table 1 compares the tagging accuracy of LDC with several recent models of Gao and Johnson (2008) and Lamar et al. (2010).

The LDC results shown in the top half of the table, which uses the MTO criterion, were obtained with the same σ-schedules as used in Figures 1 and 2. Note that the LDC algorithm is deterministic. However, the randomness in the sparse-matrix implementation of reduced-rank SVD used in the initialization step causes a small variability in performance (the standard deviation of the MTO score is 0.0004 for PTB17 and 0.003 for PTB45). The LDC results reported are averages over 20 runs. Each run was halted at iteration 15, and the score reported uses the labeling $A^*(P)$ defined at the end of Section 3.

The LDC results shown in the bottom half of the table, which uses the OTO criterion, were obtained with a variant of the LDC algorithm, in which the M phase estimates not only the Gaussian means but also the mixture coefficients. Also, different σ-schedules were used,[3]

For both PTB17 and PTB45, and under both criteria, LDC's performance nearly matches or exceeds (often by a large margin) the results achieved by the other models. We find the large increase achieved by LDC in the MTO performance under the PTB45 tagset particularly compelling. It should be noted that Abend et al. (2010) report 71.6% MTO accuracy for PTB45, but they treat all punctuation tags differently in their evaluation and therefore these results cannot be directly compared. Berg-Kirkpatrick et al. (2010) report 75.5% MTO accuracy for PTB45 by incorporating other features such as morphology; Table 1 is limited to distributional-only methods.

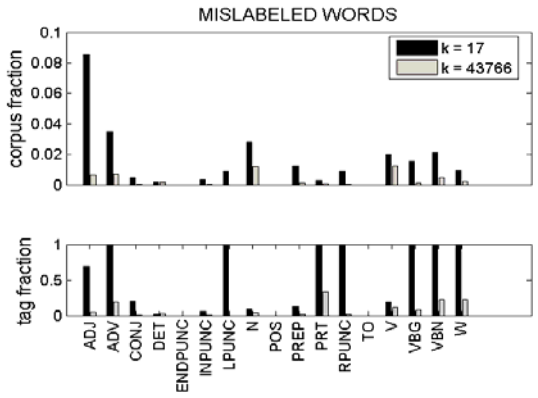

Figure 3: Mislabeled words per tag, using the PTB17 tagset. Black bars indicate mislabeled words when 17 clusters are used. Gray bars indicate words that continue to be mislabeled even when every word type is free to choose its own label, as if each type were in its own cluster—which defines the theoretically best possible non-disambiguating model. Top: fraction of the corpus mislabeled, broken down by gold tags. Bottom: fraction of tokens of each tag that are mislabeled. Many of the infrequent tags are 100% mislabeled because no induced label is mapped to these tags under MTO.

Figure 3 demonstrates the mistakes made by LDC under the MTO mapping. From the top graph, it is clear that the majority of the missed tokens are open-class words – most notably adjectives and adverbs. Over 8% of the tokens in the corpus are mislabeled adjectives – roughly one-third of all total mislabeled tokens (25.8%). Furthermore, the corresponding bar in the bottom graph indicates that over half of the adjectives are labeled incorrectly. Similarly, nearly 4% of the mislabeled tokens are adverbs, but *every* adverb in the corpus is mislabeled because no label is mapped to this tag – a common oc-

[3] All details are included in the code available at http://www.dam.brown.edu/people/elie/code/.

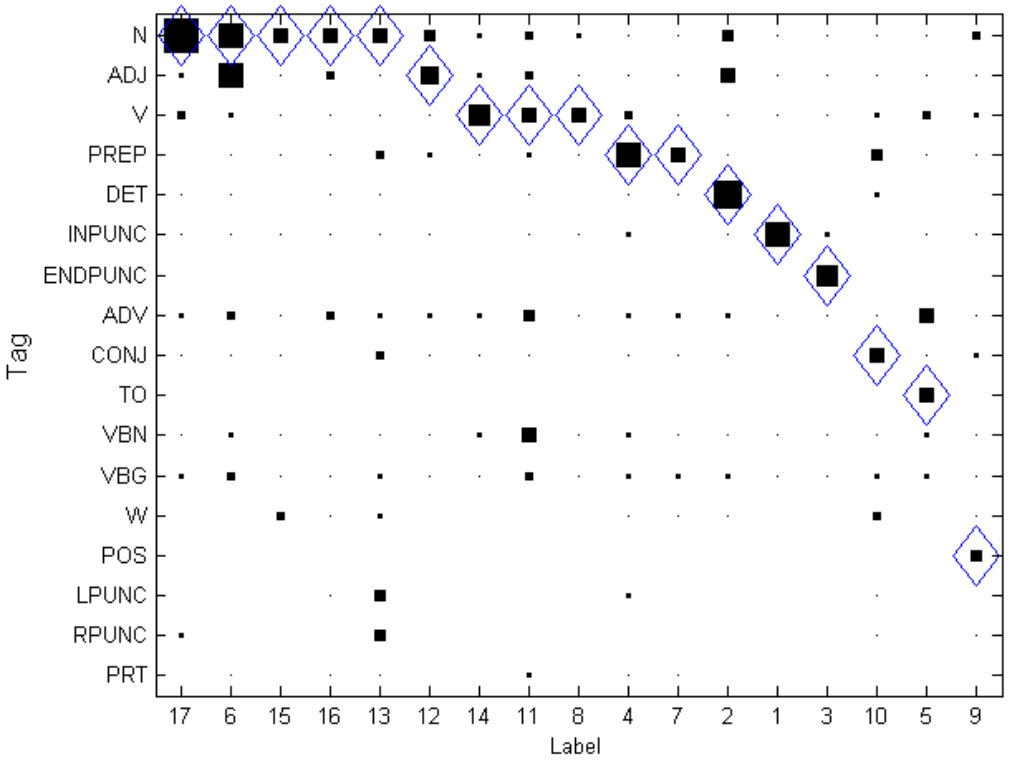

Figure 4: The confusion matrix for LDC's labeling under PTB17. The area of a black square indicates the number of tokens in each element of the confusion matrix. The diamonds indicate the induced tag under the MTO mapping. Several labels are mapped to N (Noun), and one of these labels causes appreciable confusion between nouns and adjectives. Because multiple labels are dedicated to a single tag (N, V and PREP), several tags (in this case 7) are left with no label.

currence under MTO, shared by seven of the seventeen tags.

To further illuminate the errors made by LDC, we construct the confusion matrix (figure 4). Element (*i,j*) of this matrix stores the fraction of all tokens of POS tag *i* that are given label *j* by the model. In a perfect labeling, exactly one element of each row and each column would be non-zero. As illustrated in figure 4, the confusion matrices produced by LDC are far from perfect. LDC consistently splits the Nouns into several labels and often confuses Nouns and Adjectives under a single label. These types of mistakes have been observed as well in models that use supervision (Haghighi and Klein, 2006).

5 Discussion

When devising a model for unsupervised POS induction, one challenge is to choose a model of adequate complexity, this choice being related to the bias-variance dilemma ubiquitous in statistical estimation problems. While large datasets are available, they are typically not large enough to allow efficient unsupervised learnability in models that are powerful enough to capture complex features of natural languages. Ambiguity is one of these features. Here we propose a new approach to this set of issues: start with a model that explicitly entertains ambiguity, and gradually constrain it so that it eventually converges to an unambiguous tagger.

Thus, although the algorithm uses probabilistic assignments, of Gaussian-mixture type, the goal is the construction of hard assignments. By

requiring the Gaussians to be isotropic with uniform width and by allowing that width to shrink to zero, the algorithm forces the soft assignments to converge to a set of hard assignments. Based on its performance, this simulated-annealing-like approach appears to provide a good compromise in the choice of model complexity.

LDC bears some similarities with the algorithm of Ney, Essen and Kneser (1994), further implemented, with extensions, by Clark (2003). Both models use an iterative approach to minimize an objective function, and both initialize with frequent words. However, the model of Ney et al. is, in essence, an HMM where each word type is constrained to belong to a single class (i.e., in HMM terminology, be emitted by a single hidden state). Accordingly, the objective function is the data likelihood under this constrained HMM. This takes into account only the rightward transition probabilities. Our approach is conceptually rather different from an HMM. It is more similar to the approach of Schütze (1995) and Lamar et al. (2010), where each word type is mapped into a descriptor vector derived from its left and right tag contexts. Accordingly, the objective function is that of the *K*-means clustering problem, namely a sum of intra-cluster squared distances. This objective function, unlike the likelihood under an HMM, takes into account both left and right contexts. It also makes use in a crucial way of cluster centroids (or Gaussian means), a notion that has no counterpart in the HMM approach. We note that LDC achieves much better results (by about 10%) than a recent implementation of the Ney et al. approach (Reichart et al. 2010).

The only parameters in LDC are the two parameters used to define the σ schedule, and r_1 used in the first iteration. Performance was generally found to degrade gracefully with changes in these parameters away from their optimal values. When σ was made too large in the first few iterations, it was found that the algorithm converges to the trivial minimum of the objective function $F(A)$, which maps all word types to a unique label (see section 2). An alternative would be to estimate the variance for each Gaussian separately, as is usually done in EM for Gaussian mixtures. This would not necessarily preclude the use of an iteration-dependent scaling factor, which would achieve the goal of gradually forcing the tagging to become deterministic. Investigating this and related options is left for future work.

Reduced-rank SVD is used in the initialization of the descriptor vectors, for the optimization to get off the ground. The details of this initialization step do not seem to be too critical, as witnessed by robustness against many parameter changes. For instance, using only the 400 most frequent words in the corpus—instead of all words—in the construction of the left-word and right-word context vectors in iteration 1 causes no appreciable change in performance.

The probabilistic-assignment algorithm was found to be much more robust against parameter changes than the hard-assignment version of LDC, which parallels the classical *K*-means clustering algorithm (see Section 1). We experimented with this hard-assignment latent-descriptor clustering algorithm (data not shown), and found that a number of additional devices were necessary in order to make it work properly. In particular, we found it necessary to use reduced-rank SVD on each iteration of the algorithm—as opposed to just the first iteration in the version presented here—and to gradually increase the rank *r*. Further, we found it necessary to include only the most frequent words at the beginning, and only gradually incorporate rare words in the algorithm. Both of these devices require fine tuning. Provided they are indeed appropriately tuned, the same level of performance as in the probabilistic-assignment version could be achieved. However, as mentioned, the behavior is much less robust with hard clustering.

Central to the success of LDC is the dynamic interplay between the progressively harder cluster assignments and the updated latent descriptor vectors. We operate under the assumption that if all word types were labeled optimally, words that share a label should have similar descriptor vectors arising from this optimal labeling. These similar vectors would continue to be clustered together, producing a stable equilibrium in

the dynamic process. The LDC algorithm demonstrates that, despite starting far from this optimal labeling, the alternation between vector updates and assignment updates is able to produce steadily improving clusters, as seen by the steady increase of tagging accuracy.

We envision the possibility of extending this approach in several ways. It is a relatively simple matter to extend the descriptor vectors to include context outside the nearest neighbors, which may well improve performance. In view of the computational efficiency of LDC, which runs in under one minute on a desktop PC, the added computational burden of working with the extended context is not likely to be prohibitive. LDC could also be extended to include morphological or other features, rather than relying exclusively on context. Again, we would anticipate a corresponding increase in accuracy from this additional linguistic information.

References

Omri Abend, Roi Reichart and Ari Rappoport. Improved Unsupervised POS Induction through Prototype Discovery. 2010. In *Proceedings of the 48th Annual Meeting of the ACL*.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, LLC.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless Unsupervised Learning with Features. In *proceedings of NAACL 2010*.

Alexander Clark. 2001. The unsupervised induction of stochastic context-free grammars using distributional clustering. In *CoNLL*.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 59–66.

Jianfeng Gao and Mark Johnson. 2008. A comparison of bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 344–352.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751.

João V. Graça, Kuzman Ganchev, Ben Taskar, and Fernando Pereira. 2009. Posterior vs. Parameter Sparsity in Latent Variable Models. *Neural Information Processing Systems Conference* (NIPS).

Michael Lamar, Yariv Maron, Mark Johnson, Elie Bienenstock. 2010. SVD and Clustering for Unsupervised POS Tagging. In *Proceedings of the 48th Annual Meeting of the ACL*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

William P. Headden, David McClosky, and Eugene Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of the International Conference on Computational Linguistics* (*COLING '08)*.

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (EMNLP-CoNLL), pages 296–305.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8, 1-38.

Roi Reichart, Raanan Fattal and Ari Rappoport. 2010. Improved Unsupervised POS Induction Using Intrinsic Clustering Quality and a Zipfian Constraint. *CoNLL*.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP,* pages 504–512.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual*

Meeting of the Association for Computational Linguistics (ACL'05), pages 354–362.

Kristina Toutanova, Dan Klein, Christopher D. Manning and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL* 2003, pages 252-259.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data. In *Proceedings of HLT/EMNLP*, pp. 467-474.