Improved Unsupervised POS Induction through Prototype Discovery

Omri Abend[1*] **Roi Reichart**[2] **Ari Rappoport**[1]

[1]Institute of Computer Science, [2]ICNC
Hebrew University of Jerusalem
{omria01|roiri|arir}@cs.huji.ac.il

Abstract

We present a novel fully unsupervised algorithm for POS induction from plain text, motivated by the cognitive notion of prototypes. The algorithm first identifies *landmark* clusters of words, serving as the cores of the induced POS categories. The rest of the words are subsequently mapped to these clusters. We utilize morphological and distributional representations computed in a fully unsupervised manner. We evaluate our algorithm on English and German, achieving the best reported results for this task.

1 Introduction

Part-of-speech (POS) tagging is a fundamental NLP task, used by a wide variety of applications. However, there is no single standard POS tagging scheme, even for English. Schemes vary significantly across corpora and even more so across languages, creating difficulties in using POS tags across domains and for multi-lingual systems (Jiang et al., 2009). Automatic induction of POS tags from plain text can greatly alleviate this problem, as well as eliminate the efforts incurred by manual annotations. It is also a problem of great theoretical interest. Consequently, POS induction is a vibrant research area (see Section 2).

In this paper we present an algorithm based on the theory of prototypes (Taylor, 2003), which posits that some members in cognitive categories are more central than others. These practically define the category, while the membership of other elements is based on their association with the central members. Our algorithm first clusters words based on a fine morphological representation. It then clusters the most frequent words, defining *landmark* clusters which constitute the cores of the categories. Finally, it maps the rest of the words to these categories. The last two stages utilize a distributional representation that has been shown to be effective for unsupervised parsing (Seginer, 2007).

We evaluated the algorithm in both English and German, using four different mapping-based and information theoretic clustering evaluation measures. The results obtained are generally better than all existing POS induction algorithms.

Section 2 reviews related work. Sections 3 and 4 detail the algorithm. Sections 5, 6 and 7 describe the evaluation, experimental setup and results.

2 Related Work

Unsupervised and semi-supervised POS tagging have been tackled using a variety of methods. Schütze (1995) applied latent semantic analysis. The best reported results (when taking into account all evaluation measures, see Section 5) are given by (Clark, 2003), which combines distributional and morphological information with the likelihood function of the Brown algorithm (Brown et al., 1992). Clark's tagger is very sensitive to its initialization. Reichart et al. (2010b) propose a method to identify the high quality runs of this algorithm. In this paper, we show that our algorithm outperforms not only Clark's mean performance, but often its best among 100 runs. Most research views the task as a sequential labeling problem, using HMMs (Merialdo, 1994; Banko and Moore, 2004; Wang and Schuurmans, 2005) and discriminative models (Smith and Eisner, 2005; Haghighi and Klein, 2006). Several

[*] Omri Abend is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

techniques were proposed to improve the HMM model. A Bayesian approach was employed by (Goldwater and Griffiths, 2007; Johnson, 2007; Gao and Johnson, 2008). Van Gael et al. (2009) used the infinite HMM with non-parametric priors. Graça et al. (2009) biased the model to induce a small number of possible tags for each word.

The idea of utilizing seeds and expanding them to less reliable data has been used in several papers. Haghighi and Klein (2006) use POS 'prototypes' that are manually provided and tailored to a particular POS tag set of a corpus. Freitag (2004) and Biemann (2006) induce an initial clustering and use it to train an HMM model. Dasgupta and Ng (2007) generate morphological clusters and use them to bootstrap a distributional model. Goldberg et al. (2008) use linguistic considerations for choosing a good starting point for the EM algorithm. Zhao and Marcus (2009) expand a partial dictionary and use it to learn disambiguation rules. Their evaluation is only at the type level and only for half of the words. Ravi and Knight (2009) use a dictionary and an MDL-inspired modification to the EM algorithm.

Many of these works use a dictionary providing allowable tags for each or some of the words. While this scenario might reduce human annotation efforts, it does not induce a tagging scheme but remains tied to an existing one. It is further criticized in (Goldwater and Griffiths, 2007).

Morphological representation. Many POS induction models utilize morphology to some extent. Some use simplistic representations of terminal letter sequences (e.g., (Smith and Eisner, 2005; Haghighi and Klein, 2006)). Clark (2003) models the entire letter sequence as an HMM and uses it to define a morphological prior. Dasgupta and Ng (2007) use the output of the *Morfessor* segmentation algorithm for their morphological representation. *Morfessor* (Creutz and Lagus, 2005), which we use here as well, is an unsupervised algorithm that segments words and classifies each segment as being a stem or an affix. It has been tested on several languages with strong results.

Our work has several unique aspects. First, our clustering method discovers prototypes in a fully unsupervised manner, mapping the rest of the words according to their association with the prototypes. Second, we use a distributional representation which has been shown to be effective for unsupervised parsing (Seginer, 2007). Third, we use a morphological representation based on signatures, which are sets of affixes that represent a family of words sharing an inflectional or derivational morphology (Goldsmith, 2001).

3 Distributional Algorithm

Our algorithm is given a plain text corpus and optionally a desired number of clusters k. Its output is a partitioning of words into clusters. The algorithm utilizes two representations, distributional and morphological. Although eventually the latter is used before the former, for clarity of presentation we begin by detailing the base distributional algorithm. In the next section we describe the morphological representation and its integration into the base algorithm.

Overview. The algorithm consists of two main stages: landmark clusters discovery, and word mapping. For the former, we first compute a distributional representation for each word. We then cluster the coordinates corresponding to high frequency words. Finally, we define *landmark clusters*. In the word mapping stage we map each word to the most similar landmark cluster.

The rationale behind using only the high frequency words in the first stage is twofold. First, prototypical members of a category are frequent (Taylor, 2003), and therefore we can expect the salient POS tags to be represented in this small subset. Second, higher frequency implies more reliable statistics. Since this stage determines the cores of all resulting clusters, it should be as accurate as possible.

Distributional representation. We use a simplified form of the elegant representation of lexical entries used by the Seginer unsupervised parser (Seginer, 2007). Since a POS tag reflects the grammatical role of the word and since this representation is effective to parsing, we were motivated to apply it to the present task.

Let W be the set of word types in the corpus. The right context entry of a word $x \in W$ is a pair of mappings $r\_int_x : W \to [0,1]$ and $r\_adj_x : W \to [0,1]$. For each $w \in W$, $r\_adj_x(w)$ is an adjacency score of w to x, reflecting w's tendency to appear on the right hand side of x.

For each $w \in W$, $r\_int_x(w)$ is an interchangeability score of x with w, reflecting the tendency of w to appear to the left of words that tend to appear to the right of x. This can be viewed as a similarity measure between words with respect to their right context. The higher the scores the more the words tend to be adjacent/interchangeable.

Left context parameters l_int_x and l_adj_x are defined analogously.

There are important subtleties in these definitions. First, for two words $x, w \in W$, $r\_adj_x(w)$ is generally different from $l\_adj_w(x)$. For example, if w is a high frequency word and x is a low frequency word, it is likely that w appears many times to the right of x, yielding a high $r\_adj_x(w)$, but that x appears only a few times to the left of w yielding a low $l\_adj_w(x)$. Second, from the definition of $r\_int_x(w)$ and $r\_int_w(x)$, it is clear that they need not be equal.

These functions are computed incrementally by a bootstrapping process. We initialize all mappings to be identically 0. We iterate over the words in the training corpus. For every word instance x, we take the word immediately to its right y and update x's right context using y's left context:

$$\forall w \in W: \quad r\_int_x(w) \mathrel{+}= \frac{l\_adj_y(w)}{N(y)}$$

$$\forall w \in W: \quad r\_adj_x(w) \mathrel{+}= \begin{cases} 1 & w = y \\ \frac{l\_int_y(w)}{N(y)} & w \neq y \end{cases}$$

The division by $N(y)$ (the number of times y appears in the corpus before the update) is done in order not to give a disproportional weight to high frequency words. Also, $r\_int_x(w)$ and $r\_adj_x(w)$ might become larger than 1. We therefore normalize them after all updates are performed by the number of occurrences of x in the corpus.

We update l_int_x and l_adj_x analogously using the word z immediately to the left of x. The updates of the left and right functions are done in parallel.

We define the distributional representation of a word type x to be a $4|W| + 2$ dimensional vector v_x. Each word w yields four coordinates, one for each direction (left/right) and one for each mapping type (int/adj). Two additional coordinates represent the frequency in which the word appears to the left and to the right of a stopping punctuation. Of the $4|W|$ coordinates corresponding to words, we allow only $2n$ to be non-zero: the n top scoring among the right side coordinates (those of r_int_x and r_adj_x), and the n top scoring among the left side coordinates (those of l_int_x and l_adj_x). We used $n = 50$.

The distance between two words is defined to be one minus the cosine of the angle between their representation vectors.

Coordinate clustering. Each of our landmark clusters will correspond to a set of high frequency words (HFWs). The number of HFWs is much larger than the number of expected POS tags. Hence we should cluster HFWs. Our algorithm does that by unifying some of the non-zero coordinates corresponding to HFWs in the distributional representation defined above.

We extract the words that appear more than N times per million[1] and apply the following procedure I times (5 in our experiments).

We run average link clustering with a threshold α (AVGLINK$_\alpha$, (Jain et al., 1999)) on these words, in each iteration initializing every HFW to have its own cluster. AVGLINK$_\alpha$ means running the average link algorithm until the two closest clusters have a distance larger than α. We then use the induced clustering to update the distributional representation, by collapsing all coordinates corresponding to words appearing in the same cluster into a single coordinate whose value is the sum of the collapsed coordinates' values. In order to produce a conservative (fine) clustering, we used a relatively low α value of 0.25.

Note that the AVGLINK$_\alpha$ initialization in each of the I iterations assigns each HFW to a separate cluster. The iterations differ in the distributional representation of the HFWs, resulting from the previous iterations.

In our English experiments, this process reduced the dimension of the HFWs set (the number of coordinates that are non-zero in at least one of the HFWs) from 14365 to 10722. The average number of non-zero coordinates per word decreased from 102 to 55.

Since all eventual POS categories correspond to clusters produced at this stage, to reduce noise we delete clusters of less than five elements.

Landmark detection. We define landmark clusters using the clustering obtained in the final iteration of the coordinate clustering stage. However, the number of clusters might be greater than the desired number k, which is an optional parameter of the algorithm. In this case we select a subset of k clusters that best covers the HFW space. We use the following heuristic. We start from the most frequent cluster, and greedily select the clus-

[1]We used $N = 100$, yielding 1242 words for English and 613 words for German.

ter farthest from the clusters already selected. The distance between two clusters is defined to be the average distance between their members. A cluster's distance from a set of clusters is defined to be its minimal distance from the clusters in the set. The final set of clusters $\{L_1, ..., L_k\}$ and their members are referred to as *landmark clusters* and *prototypes*, respectively.

Mapping all words. Each word $w \in W$ is assigned the cluster L_i that contains its nearest prototype:

$$d(w, L_i) = min_{x \in L_i}\{1 - cos(v_w, v_x)\}$$
$$Map(w) = argmin_{L_i}\{d(w, L_i)\}$$

Words that appear less than 5 times are considered as *unknown words*. We consider two schemes for handling unknown words. One randomly maps each such word to a cluster, using a probability proportional to the number of unique known words already assigned to that cluster. However, when the number k of landmark clusters is relatively large, it is beneficial to assign all unknown words to a separate new cluster (after running the algorithm with $k - 1$). In our experiments, we use the first option when k is below some threshold (we used 15), otherwise we use the second.

4 Morphological Model

The morphological model generates another word clustering, based on the notion of a signature. This clustering is integrated with the distributional model as described below.

4.1 Morphological Representation

We use the *Morfessor* (Creutz and Lagus, 2005) word segmentation algorithm. First, all words in the corpus are segmented. Then, for each stem, the set of all affixes with which it appears (its *signature*, (Goldsmith, 2001)) is collected. The morphological representation of a word type is then defined to be its stem's signature in conjunction with its specific affixes[2] (See Figure 1).

We now collect all words having the same representation. For instance, if the words *joined* and *painted* are found to have the same signature, they would share the same cluster since both have the affix '_ed'. The word *joins* does not share the same cluster with them since it has a different affix, '_s'. This results in coarse-grained clusters exclusively defined according to morphology.

[2]A word may contain more than a single affix.

Types	join	joins	joined	joining
Stem	join	join	join	join
Affixes	ϕ	_s	_ed	_ing
Signature	$\{\phi$,_ed,_s,_ing$\}$			

Figure 1: An example for a morphological representation, defined to be the conjunction of its affix(es) with the stem's signature.

In addition, we incorporate capitalization information into the model, by constraining all words that appear capitalized in more than half of their instances to belong to a separate cluster, regardless of their morphological representation. The motivation for doing so is practical: capitalization is used in many languages to mark grammatical categories. For instance, in English capitalization marks the category of proper names and in German it marks the noun category . We report English results both with and without this modification.

Words that contain non-alphanumeric characters are represented as the sequence of the non-alphanumeric characters they include, e.g., 'vis-à-vis' is represented as *("-", "-")*. We do not assign a morphological representation to words including more than one stem (like *weatherman*), to words that have a null affix (i.e., where the word is identical to its stem) and to words whose stem is not shared by any other word (signature of size 1). Words that were not assigned a morphological representation are included as singletons in the morphological clustering.

4.2 Distributional-Morphological Algorithm

We detail the modifications made to our base distributional algorithm given the morphological clustering defined above.

Coordinate clustering and landmarks. We constrain AVGLINK$_\alpha$ to begin by forming links between words appearing in the same morphological cluster. Only when the distance between the two closest clusters gets above α we remove this constraint and proceed as before. This is equivalent to performing AVGLINK$_\alpha$ separately within each morphological cluster and then using the result as an initial condition for an AVGLINK$_\alpha$ coordinate clustering. The modified algorithm in this stage is otherwise identical to the distributional algorithm.

Word mapping. In this stage words that are not prototypes are mapped to one of the landmark clusters. A reasonable strategy would be to map all words sharing a morphological cluster as a single unit. However, these clusters are too coarse-grained. We therefore begin by partitioning the morphological clusters into sub-clusters according to their distributional behavior. We do so by applying AVGLINK$_\beta$ (the same as AVGLINK$_\alpha$ but with a different parameter) to each morphological cluster. Since our goal is cluster *refinement*, we use a β that is considerably higher than α (0.9).

We then find the closest prototype to each such sub-cluster (averaging the distance across all of the latter's members) and map it as a single unit to the cluster containing that prototype.

5 Clustering Evaluation

We evaluate the clustering produced by our algorithm using an external quality measure: we take a corpus tagged by gold standard tags, tag it using the induced tags, and compare the two taggings. There is no single accepted measure quantifying the similarity between two taggings. In order to be as thorough as possible, we report results using four known measures, two mapping-based measures and two information theoretic ones.

Mapping-based measures. The induced clusters have arbitrary names. We define two mapping schemes between them and the gold clusters. After the induced clusters are mapped, we can compute a derived accuracy. The **Many-to-1** measure finds the mapping between the gold standard clusters and the induced clusters which maximizes accuracy, allowing several induced clusters to be mapped to the same gold standard cluster. The **1-to-1** measure finds the mapping between the induced and gold standard clusters which maximizes accuracy such that no two induced clusters are mapped to the same gold cluster. Computing this mapping is equivalent to finding the maximal weighted matching in a bipartite graph, whose weights are given by the intersection sizes between matched classes/clusters. As in (Reichart and Rappoport, 2008), we use the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957) to solve this problem.

Information theoretic measures. These are based on the observation that a good clustering reduces the uncertainty of the gold tag given the induced cluster, and vice-versa. Several such measures exist; we use **V** (Rosenberg and Hirschberg, 2007) and **NVI** (Reichart and Rappoport, 2009), **VI**'s (Meila, 2007) normalized version.

6 Experimental Setup

Since a goal of unsupervised POS tagging is inducing an annotation scheme, comparison to an existing scheme is problematic. To address this problem we compare to three different schemes in two languages. In addition, the two English schemes we compare with were designed to tag corpora contained in our training set, and have been widely and successfully used with these corpora by a large number of applications.

Our algorithm was run with the exact same parameters on both languages: $N = 100$ (high frequency threshold), $n = 50$ (the parameter that determines the effective number of coordinates), $\alpha = 0.25$ (cluster separation during landmark cluster generation), $\beta = 0.9$ (cluster separation during refinement of morphological clusters).

The algorithm we compare with in most detail is (Clark, 2003), which reports the best current results for this problem (see Section 7). Since Clark's algorithm is sensitive to its initialization, we ran it a 100 times and report its average and standard deviation in each of the four measures. In addition, we report the percentile in which our result falls with respect to these 100 runs.

Punctuation marks are very frequent in corpora and are easy to cluster. As a result, including them in the evaluation greatly inflates the scores. For this reason we do not assign a cluster to punctuation marks and we report results using this policy, which we recommend for future work. However, to be able to directly compare with previous work, we also report results for the full POS tag set. We do so by assigning a singleton cluster to each punctuation mark (in addition to the k required clusters). This simple heuristic yields very high performance on punctuation, scoring (when all other words are assumed perfect tagging) 99.6% (99.1%) 1-to-1 accuracy when evaluated against the English fine (coarse) POS tag sets, and 97.2% when evaluated against the German POS tag set.

For English, we trained our model on the 39832 sentences which constitute sections 2-21 of the PTB-WSJ and on the 500K sentences from the NYT section of the NANC newswire corpus (Graff, 1995). We report results on the WSJ part of our data, which includes 950028 words tokens in 44389 types. Of the tokens, 832629 (87.6%)

English	Fine k=13				Coarse k=13				Fine k=34			
	Prototype Tagger	Clark		%	Prototype Tagger	Clark		%	Prototype Tagger	Clark		%
		μ	σ			μ	σ			μ	σ	
Many–to–1	**61.0**	55.1	1.6	100	**70.0**	66.9	2.1	94	**71.6**	69.8	1.5	90
	55.5	48.8	1.8	100	**66.1**	62.6	2.3	94	**67.5**	65.5	1.7	90
1–to–1	**60.0**	52.2	1.9	100	**58.1**	49.4	2.9	100	**63.5**	54.5	1.6	100
	54.9	46.0	2.2	100	**53.7**	43.8	3.3	100	**58.8**	48.5	1.8	100
NVI	**0.652**	0.773	0.027	100	**0.841**	0.972	0.036	100	**0.663**	0.725	0.018	100
	0.795	0.943	0.033	100	**1.052**	1.221	0.046	100	**0.809**	0.885	0.022	100
V	**0.636**	0.581	0.015	100	**0.590**	0.543	0.018	100	**0.677**	0.659	0.008	100
	0.542	0.478	0.019	100	**0.484**	0.429	0.023	100	**0.608**	0.588	0.010	98

German	k=17				k=26			
	Prototype Tagger	Clark		%	Prototype Tagger	Clark		%
		μ	σ			μ	σ	
Many–to-1	64.6	**64.7**	1.2	41	**68.2**	67.8	1.0	60
	58.9	**59.1**	1.4	40	**63.2**	62.8	1.2	60
1–to–1	**53.7**	52.0	1.8	77	**56.0**	52.0	2.1	99
	48.0	46.0	2.3	78	**50.7**	45.9	2.6	99
NVI	**0.667**	0.675	0.019	66	**0.640**	0.682	0.019	100
	0.819	0.829	0.025	66	**0.785**	0.839	0.025	100
V	**0.646**	0.645	0.010	50	**0.675**	0.657	0.008	100
	0.552	**0.553**	0.013	48	**0.596**	0.574	0.010	100

Table 1: Top: English. Bottom: German. Results are reported for our model (Prototype Tagger), Clark's average score (μ), Clark's standard deviation (σ) and the fraction of Clark's results that scored worse than our model (%). For the mapping based measures, results are accuracy percentage. For $V \in [0, 1]$, higher is better. For high quality output, $NVI \in [0, 1]$ as well, and lower is better. In each entry, the top number indicates the score when including punctuation and the bottom number the score when excluding it. In English, our results are always better than Clark's. In German, they are almost always better.

are not punctuation. The percentage of unknown words (those appearing less than five times) is 1.6%. There are 45 clusters in this annotation scheme, 34 of which are not punctuation.

We ran each algorithm both with k=13 and k=34 (the number of desired clusters). We compare the output to two annotation schemes: the fine grained PTB WSJ scheme, and the coarse grained tags defined in (Smith and Eisner, 2005). The output of the k=13 run is evaluated both against the coarse POS tag annotation (the *'Coarse k=13'* scenario) and against the full PTB-WSJ annotation scheme (the *'Fine k=13'* scenario). The k=34 run is evaluated against the full PTB-WSJ annotation scheme (the *'Fine k=34'* scenario).

The POS cluster frequency distribution tends to be skewed: each of the 13 most frequent clusters in the PTB-WSJ cover more than 2.5% of the tokens (excluding punctuation) and together 86.3% of them. We therefore chose k=13, since it is both the number of coarse POS tags (excluding punctuation) as well as the number of frequent POS tags in the PTB-WSJ annotation scheme. We chose k=34 in order to evaluate against the full 34 tags PTB-WSJ annotation scheme (excluding punctuation) using the same number of clusters.

For German, we trained our model on the 20296 sentences of the NEGRA corpus (Brants, 1997) and on the first 450K sentences of the DeWAC corpus (Baroni et al., 2009). DeWAC is a corpus extracted by web crawling and is therefore out of domain. We report results on the NEGRA part, which includes 346320 word tokens of 49402 types. Of the tokens, 289268 (83.5%) are not punctuation. The percentage of unknown words (those appearing less than five times) is 8.1%. There are 62 clusters in this annotation scheme, 51 of which are not punctuation.

We ran the algorithms with k=17 and k=26. k=26 was chosen since it is the number of clusters that cover each more than 0.5% of the NEGRA tokens, and in total cover 96% of the (non-punctuation) tokens. In order to test our algorithm in another scenario, we conducted experiments with k=17 as well, which covers 89.9% of the tokens. All outputs are compared against NEGRA's gold standard scheme.

We do not report results for k=51 (where the number of gold clusters is the same as the number of induced clusters), since our algorithm produced only 42 clusters in the landmark detection stage. We could of course have modified the parameters to allow our algorithm to produce 51 clusters. However, we wanted to use the exact same parameters as those used for the English experiments to minimize the issue of parameter tuning.

In addition to the comparisons described above, we present results of experiments (in the 'Fine

	B	B+M	B+C	F(I=1)	F
M-to-1	53.3	54.8	58.2	57.3	**61.0**
1-to-1	50.2	51.7	55.1	54.8	**60.0**
NVI	0.782	0.720	0.710	0.742	**0.652**
V	0.569	0.598	0.615	0.597	**0.636**

Table 2: A comparison of partial versions of the model in the 'Fine k=13' WSJ scenario. M-to-1 and 1-to-1 results are reported in accuracy percentage. Lower NVI is better. *B* is the strictly distributional algorithm, *B+M* adds the morphological model, *B+C* adds capitalization to *B*, *F(I=1)* consists of all components, where only one iteration of coordinate clustering is performed, and *F* is the full model.

	M-to-1	1-to-1	V	VI
Prototype	**71.6**	**63.5**	**0.677**	**2.00**
Clark	69.8	54.5	0.659	2.18
HK	–	41.3	–	–
J	43–62	37–47	–	4.23–5.74
GG	–	–	–	2.8
GJ	–	40–49.9	–	4.03–4.47
VG	–	–	0.54-0.59	2.5–2.9
GGTP-45	65.4	44.5	–	–
GGTP-17	70.2	49.5	–	–

Table 4: Comparison of our algorithms with the recent fully unsupervised POS taggers for which results are reported. The models differ in the annotation scheme, the corpus size and the number of induced clusters (k) that they used. HK: (Haghighi and Klein, 2006), 193K tokens, fine tags, k=45. GG: (Goldwater and Griffiths, 2007), 24K tokens, coarse tags, k=17. J : (Johnson, 2007), 1.17M tokens, fine tags, k=25–50. GJ: (Gao and Johnson, 2008), 1.17M tokens, fine tags, k=50. VG: (Van Gael et al., 2009), 1.17M tokens, fine tags, k=47–192. GGTP-45: (Graça et al., 2009), 1.17M tokens, fine tags, k=45. GGTP-17: (Graça et al., 2009), 1.17M tokens, coarse tags, k=17. Lower VI values indicate better clustering. VI is computed using e as the base of the logarithm. Our algorithm gives the best results.

k=13' scenario) that quantify the contribution of each component of the algorithm. We ran the base distributional algorithm, a variant which uses only capitalization information (i.e., has only one non-singleton morphological class, that of words appearing capitalized in most of their instances) and a variant which uses no capitalization information, defining the morphological clusters according to the morphological representation alone.

7 Results

Table 1 presents results for the English and German experiments. For English, our algorithm obtains better results than Clark's in all measures and scenarios. It is without exception better than the average score of Clark's and in most cases better than the maximal Clark score obtained in 100 runs.

A significant difference between our algorithm and Clark's is that the latter, like most algorithms which addressed the task, induces the clustering

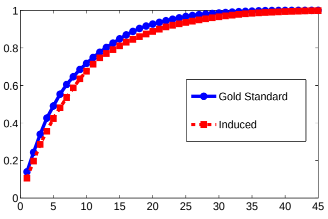

Figure 2: POS class frequency distribution for our model and the gold standard, in the 'Fine k=34' scenario. The distributions are similar.

by maximizing a non-convex function. These functions have many local maxima and the specific solution to which algorithms that maximize them converge strongly depends on their (random) initialization. Therefore, their output's quality often significantly diverges from the average. This issue is discussed in depth in (Reichart et al., 2010b). Our algorithm is deterministic[3].

For German, in the k=26 scenario our algorithm outperforms Clark's, often outperforming even its maximum in 100 runs. In the k=17 scenario, our algorithm obtains a higher score than Clark with probability 0.4 to 0.78, depending on the measure and scenario. Clark's average score is slightly better in the Many-to-1 measure, while our algorithm performs somewhat better than Clark's average in the 1-to-1 and NVI measures.

The DeWAC corpus from which we extracted statistics for the German experiments is out of domain with respect to NEGRA. The corresponding corpus in English, NANC, is a newswire corpus and therefore clearly in-domain with respect to WSJ. This is reflected by the percentage of unknown words, which was much higher in German than in English (8.1% and 1.6%), lowering results.

Table 2 shows the effect of each of our algorithm's components. Each component provides an improvement over the base distributional algorithm. The full coordinate clustering stage (several iterations, F) considerably improves the score over a single iteration (F(I=1)). Capitalization information increases the score more than the morphological information, which might stem from the granularity of the POS tag set with respect to names. This analysis is supported by similar experiments we made in the 'Coarse k=13' scenario (not shown in tables here). There, the decrease in performance was only of 1%–2% in the mapping

[3]The fluctuations inflicted on our algorithm by the random mapping of unknown words are of less than 0.1% .

	Excluding Punctuation				Including Punctuation				Perfect Punctuation			
	M-to-1	1-to-1	NVI	V	M-to-1	1-to-1	NVI	V	M-to-1	1-to-1	NVI	V
Van Gael	59.1	48.4	0.999	0.530	62.3	51.3	0.861	0.591	64.0	54.6	0.820	0.610
Prototype	**67.5**	**58.8**	**0.809**	**0.608**	**71.6**	**63.5**	**0.663**	**0.677**	**71.6**	**63.9**	**0.659**	**0.679**

Table 3: Comparison between the *iHMM: PY-fixed* model (Van Gael et al., 2009) and ours with various punctuation assignment schemes. Left section: punctuation tokens are excluded. Middle section: punctuation tokens are included. Right section: perfect assignment of punctuation is assumed.

based measures and 3.5% in the V measure.

Finally, Table 4 presents reported results for all recent algorithms we are aware of that tackled the task of unsupervised POS induction from plain text. Results for our algorithm's and Clark's are reported for the 'Fine, k=34' scenario. The settings of the various experiments vary in terms of the exact annotation scheme used (coarse or fine grained) and the size of the test set. However, the score differences are sufficiently large to justify the claim that our algorithm is currently the best performing algorithm on the PTB-WSJ corpus for POS induction from plain text[4].

Since previous works provided results only for the scenario in which punctuation is included, the reported results are not directly comparable. In order to quantify the effect various punctuation schemes have on the results, we evaluated the *'iHMM: PY-fixed'* model (Van Gael et al., 2009) and ours when punctuation is excluded, included or perfectly tagged[5]. The results (Table 3) indicate that most probably even after an appropriate correction for punctuation, our model remains the best performing one.

8 Discussion

In this work we presented a novel unsupervised algorithm for POS induction from plain text. The algorithm first generates relatively accurate clusters of high frequency words, which are subsequently used to bootstrap the entire clustering. The distributional and morphological representations that we use are novel for this task.

We experimented on two languages with mapping and information theoretic clustering evaluation measures. Our algorithm obtains the best reported results on the English PTB-WSJ corpus. In addition, our results are almost always better than Clark's on the German NEGRA corpus.

[4]Graça et al. (2009) report very good results for 17 tags in the M-1 measure. However, their 1-1 results are quite poor, and results for the common IT measures were not reported. Their results for 45 tags are considerably lower.

[5]We thank the authors for sending us their data.

We have also performed a manual error analysis, which showed that our algorithm performs much better on closed classes than on open classes. In order to asses this quantitatively, let us define a random variable for each of the gold clusters, which receives a value corresponding to each induced cluster with probability proportional to their intersection size. For each gold cluster, we compute the entropy of this variable. In addition, we greedily map each induced cluster to a gold cluster and compute the ratio between their intersection size and the size of the gold cluster (mapping accuracy).

We experimented in the 'Fine k=34' scenario. The clusters that obtained the best scores were (brackets indicate mapping accuracy and entropy for each of these clusters) coordinating conjunctions (95%, 0.32), prepositions (94%, 0.32), determiners (94%, 0.44) and modals (93%, 0.45). These are all closed classes.

The classes on which our algorithm performed worst consist of open classes, mostly verb types: past tense verbs (47%, 2.2), past participle verbs (44%, 2.32) and the morphologically unmarked non-3rd person singular present verbs (32%, 2.86). Another class with low performance is the proper nouns (37%, 2.9). The errors there are mostly of three types: confusions between common and proper nouns (sometimes due to ambiguity), unknown words which were put in the unknown words cluster, and abbreviations which were given a separate class by our algorithm. Finally, the algorithm's performance on the heterogeneous adverbs class (19%, 3.73) is the lowest.

Clark's algorithm exhibits[6] a similar pattern with respect to open and closed classes. While his algorithm performs considerably better on adverbs (15% mapping accuracy difference and 0.71 entropy difference), our algorithm scores considerably better on prepositions (17%, 0.77), superlative adjectives (38%, 1.37) and plural proper names (45%, 1.26).

[6]Using average mapping accuracy and entropy over the 100 runs.

Naturally, this analysis might reflect the arbitrary nature of a manually design POS tag set rather than deficiencies in automatic POS induction algorithms. In future work we intend to analyze the output of such algorithms in order to improve POS tag sets.

Our algorithm and Clark's are monosemous (i.e., they assign each word exactly one tag), while most other algorithms are polysemous. In order to assess the performance loss caused by the monosemous nature of our algorithm, we took the M-1 greedy mapping computed for the entire dataset and used it to compute accuracy over the monosemous and polysemous words separately. Results are reported for the English 'Fine k=34' scenario (without punctuation). We define a word to be monosemous if more than 95% of its tokens are assigned the same gold standard tag. For English, there are approximately 255K polysemous tokens and 578K monosemous ones. As expected, our algorithm is much more accurate on the monosemous tokens, achieving 76.6% accuracy, compared to 47.1% on the polysemous tokens.

The evaluation in this paper is done at the token level. Type level evaluation, reflecting the algorithm's ability to detect the set of possible POS tags for each word type, is important as well. It could be expected that a monosemous algorithm such as ours would perform poorly in a type level evaluation. In (Reichart et al., 2010a) we discuss type level evaluation at depth and propose type level evaluation measures applicable to the POS induction problem. In that paper we compare the performance of our Prototype Tagger with leading unsupervised POS tagging algorithms (Clark, 2003; Goldwater and Griffiths, 2007; Gao and Johnson, 2008; Van Gael et al., 2009). Our algorithm obtained the best results in 4 of the 6 measures in a margin of 4–6%, and was second best in the other two measures. Our results were better than Clark's (the only other monosemous algorithm evaluated there) on all measures in a margin of 5–21%. The fact that our monosemous algorithm was better than good polysemous algorithms in a type level evaluation can be explained by the prototypical nature of the POS phenomenon (a longer discussion is given in (Reichart et al., 2010a)). However, the quality upper bound for monosemous algorithms is obviously much lower than that for polysemous algorithms, and we expect polysemous algorithms to outperform monosemous algorithms in the future in both type level and token level evaluations.

The skewed (Zipfian) distribution of POS class frequencies in corpora is a problem for many POS induction algorithms, which by default tend to induce a clustering having a balanced distribution. Explicit modifications to these algorithms were introduced in order to bias their model to produce such a distribution (see (Clark, 2003; Johnson, 2007; Reichart et al., 2010b)). An appealing property of our model is its ability to induce a skewed distribution without being explicitly tuned to do so, as seen in Figure 2.

Acknowledgements. We would like to thank Yoav Seginer for his help with his parser.

References

Michele Banko and Robert C. Moore, 2004. *Part of Speech Tagging in Context.* COLING '04.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi and Eros Zanchetta, 2009. *The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora.* Language Resources and Evaluation.

Chris Biemann, 2006. *Unsupervised Part-of-Speech Tagging Employing Efficient Graph Clustering.* COLING-ACL '06 Student Research Workshop.

Thorsten Brants, 1997. *The NEGRA Export Format.* CLAUS Report, Saarland University.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souze, Jenifer C. Lai and Robert Mercer, 1992. *Class-Based N-Gram Models of Natural Language.* Computational Linguistics, 18(4):467–479.

Alexander Clark, 2003. *Combining Distributional and Morphological Information for Part of Speech Induction.* EACL '03.

Mathias Creutz and Krista Lagus, 2005. *Inducing the Morphological Lexicon of a Natural Language from Unannotated Text.* AKRR '05.

Sajib Dasgupta and Vincent Ng, 2007. *Unsupervised Part-of-Speech Acquisition for Resource-Scarce Languages.* EMNLP-CoNLL '07.

Dayne Freitag, 2004. *Toward Unsupervised Whole-Corpus Tagging.* COLING '04.

Jianfeng Gao and Mark Johnson, 2008. *A Comparison of Bayesian Estimators for Unsupervised Hidden Markov Model POS Taggers.* EMNLP '08.

Yoav Goldberg, Meni Adler and Michael Elhadad, 2008. *EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start).* ACL '08.

John Goldsmith, 2001. *Unsupervised Learning of the Morphology of a Natural Language.* Computational Linguistics, 27(2):153–198.

Sharon Goldwater and Tom Griffiths, 2007. *Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging.* ACL '07.

João Graça, Kuzman Ganchev, Ben Taskar and Frenando Pereira, 2009. Posterior vs. Parameter Sparsity in Latent Variable Models. *NIPS '09*.

David Graff, 1995. *North American News Text Corpus.* Linguistic Data Consortium. LDC95T21.

Aria Haghighi and Dan Klein, 2006. *Prototype-driven Learning for Sequence Labeling.* HLT–NAACL '06.

Anil K. Jain, Narasimha M. Murty and Patrick J. Flynn, 1999. *Data Clustering: A Review.* ACM Computing Surveys 31(3):264–323.

Wenbin Jiang, Liang Huang and Qun Liu, 2009. *Automatic Adaptation of Annotation Standards: Chinese Word Segmentation and POS Tagging – A Case Study.* ACL '09.

Mark Johnson, 2007. *Why Doesnt EM Find Good HMM POS-Taggers?* EMNLP-CoNLL '07.

Harold W. Kuhn, 1955. *The Hungarian method for the Assignment Problem.* Naval Research Logistics Quarterly, 2:83-97.

Marina Meila, 2007. *Comparing Clustering – an Information Based Distance.* Journal of Multivariate Analysis, 98:873–895.

Bernard Merialdo, 1994. *Tagging English Text with a Probabilistic Model.* Computational Linguistics, 20(2):155–172.

James Munkres, 1957. *Algorithms for the Assignment and Transportation Problems.* Journal of the SIAM, 5(1):32–38.

Sujith Ravi and Kevin Knight, 2009. *Minimized Models for Unsupervised Part-of-Speech Tagging.* ACL '09.

Roi Reichart and Ari Rappoport, 2008. *Unsupervised Induction of Labeled Parse Trees by Clustering with Syntactic Features.* COLING '08.

Roi Reichart and Ari Rappoport, 2009. *The NVI Clustering Evaluation Measure.* CoNLL '09.

Roi Reichart, Omri Abend and Ari Rappoport, 2010a. *Type Level Clustering Evaluation: New Measures and a POS Induction Case Study.* CoNLL '10.

Roi Reichart, Raanan Fattal and Ari Rappoport, 2010b. *Improved Unsupervised POS Induction Using Intrinsic Clustering Quality and a Zipfian Constraint.* CoNLL '10.

Andrew Rosenberg and Julia Hirschberg, 2007. *V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure.* EMNLP '07.

Hinrich Schütze, 1995. *Distributional part-of-speech tagging.* EACL '95.

Yoav Seginer, 2007. *Fast Unsupervised Incremental Parsing.* ACL '07.

Noah A. Smith and Jason Eisner, 2005. *Contrastive Estimation: Training Log-Linear Models on Unlabeled Data.* ACL '05.

John R. Taylor, 2003. *Linguistic Categorization: Prototypes in Linguistic Theory, Third Edition.* Oxford University Press.

Jurgen Van Gael, Andreas Vlachos and Zoubin Ghahramani, 2009. *The Infinite HMM for Unsupervised POS Tagging.* EMNLP '09.

Qin Iris Wang and Dale Schuurmans, 2005. *Improved Estimation for Unsupervised Part-of-Speech Tagging.* IEEE NLP–KE '05.

Qiuye Zhao and Mitch Marcus, 2009. *A Simple Unsupervised Learner for POS Disambiguation Rules Given Only a Minimal Lexicon.* EMNLP '09.