Unsupervised Resource Creation for Unsupervised Part of Speech Tagging

Bill Bryce

Draft 1.0

November, 2015

I. Introduction

Unsupervised learning has become a popular vein of research in natural language processing. The ability to learn structure from nothing but raw text empowers the researcher to mine content and connections from large corpora, even if these corpora are of obscure languages collected in short period of time and with little if any prior linguistic knowledge. However, the art of unsupervised learning is far from perfected. Supervised approaches often yield better results in most tasks due to their being trained on hand-annotated gold standard corpora. Yet the cost of producing gold standard corpora is high, and typically provides knowledge of only one target language at a time. If the same amount of time can be spent researching a system which is language independent, which can derive linguistic knowledge from many languages without any training at all, the gain is greater. The allure of researching unsupervised learning methods becomes clear.

Here we look at the task of POS induction – the unsupervised task of learning the parts of speech of a language using no annotated training data, but only raw text. Specifically, we expand upon the work of Haghighi and Klein (2006) and Christodoulopoulos et al. (2010). The former presents a system that uses prototypes as features in order to cluster the types of words in a corpus. These prototypes are hand-coded gold standard prototypes, and so the approach is not entirely unsupervised – rather semi-supervised. Christodoulopoulos et al. (2010) then derive prototypes from the output of other unsupervised systems and pass them as features to Haghighi and Klein's (2010) system, thus rendering the prototype-driven approach to POS induction wholly unsupervised – absolutely no prior linguistic knowledge required. We expand upon the work of these authors by trying novel methods of prototype derivation from the unsupervised POS induction systems first presented in Brown et al. (1992) and Clark (2003).

In section II we lay out the task of POS induction and the kind of linguistic knowledge that can be obtained from raw text. In section III we review some of the previous work that has been done in POS induction. In section IV we describe our experiments and the different ways in which we derive prototypes. In section V we give the results of those experiments. In section VI we discuss the results. In section VII we conclude.

II. POS Induction

Learning parts of speech is often the first step in parsing the syntactic structure of a language. This syntactic structure in turn aids systems in applications of knowledge mining, question answering, and machine translation to name a few. A good knowledge of a language's POS inventory and the natural combinatory usage of that inventory should lead to more robust inferences about the syntactic structures of the language.

POS induction goes like this: We have a corpus of text in some language. From this text and only from this text, we want to be able to find patterns which help us to discover which tokens are related to others in terms of their orthography, their morphology, and their collocation with other tokens in the text. The input may look like this:

(1) The quick boy makes a high jump .

We want the output of the system to look something like this:

(2) The/1 quick/2 boy/3 makes/4 a/1 high/2 jump/3 ./5

But the numbers that tag the tokens in the text mean nothing unless we compare them to a gold standard, like this:

(3) The/DT quick/JJ boy/NN makes/VBZ a/DT high/JJ jump/NN ./.

Looking closely at (2) and (3), we see that every time the tag "DT" occurs in the gold standard, a "1" occurs in the system output, and so on for all other tags. The hypothetical output in (2), then, is perfect when compared to the gold standard in (3). So, a POS induction system will take raw text as input, and render some kind of output such that the raw text can then be tagged with the output. The value of any POS induction system must then be tested on gold standard annotation. Critically, to be completely unsupervised, the system must not train on this gold standard data in any way. If it does, it is either semi-supervised in nature or wholly supervised and so should be called a POS tagger or at least not a POS inducer.

We will here describe the outputs of those POS induction systems that cluster and those that label; we will outline the previous work using these strategies in the next section. Let's say that this is our corpus, and that we have normalized the text so that all alpha characters are lowercase:

(4) the live animals live in the forest .

This corpus has eight tokens (the period is a token) and six types (only unique tokens are considered types – the strings *the* and *live* appear twice). There are two ways to go about labeling this text. The first is to assign a tag to each type in the text, and then, when producing tagged output, to always tag that type with that text. This approach, clustering, can be thought of as placing every unique string into a box. Consider the unique strings in this text:

(5) { the live animals in forest . }

The system may cluster the types like so:

(6)

1 →	the	4 →	in
2 →	live	5 →	.
3 →	animals, forest		

How many clusters are produced depends on the system, but typically the number of clusters is passed as a parameter. Ideally, all types placed in the same cluster should have in common that they are the same POS. We see that this is indeed the case for *animals* and *forest*, which are both nouns (though we ignore plural morphology in this example). It should be clear what is undesirable about this approach to POS induction if we look at the tagged output of this system:

(7) the/1 live/2 animals/3 live/2 in/4 the/1 forest ./5

Notice that both instances of the string *live* have been tagged as members of cluster 2, implying that both instances are the same POS. Yet clearly, the first instance of *live* is an adjective and the second is a verb.

The second kind of POS inducer we will mention here, labeling systems, can solve the problem of *live*. Labeling systems tag each token in the text, and so there is not the same commitment made by clustering systems that all instances of *live* must be tagged exactly the same way. But labeling systems can be subject to their own drawbacks. Consider this possible output form a POS inducer that labels:

(8) the/1 live/2 animals/3 live/4 in/5 the/6 forest/3 ./7

The number of labels to be used, like the number of clusters above, is often passed as a parameter. Notice that in this output, the two instance of the string *the* have been given two different labels. This is not likely to happen in a labeling system, but it is allowed, and illustrates our point here. Since labeling systems tend to depend on Bayesian probabilistic models for their output, it makes decisions at each token as it progresses through the corpus. So it is possible for the observed probabilities to cause the system to decide that *the* is what we consider a determiner when found at the start of a sentence, but for the system to consider *the* something else when sandwiched between *in* and any other string. In the case of *the*, the desirable output is that of the system that clusters types, but in the case of *live*, the desirable output is that of the system which labels tokens.

III. Previous Work

To induce the POS tag of a string given only raw text, there are basically three sources of information to be exploited: the string itself as a whole, substrings shared by different strings in the text – i.e. morphology – and finally the context in which the string is placed – i.e. what other strings consistently appear adjacent to or near instances of the string. Brown et al. (1992) use two of these sources of information, leaving morphology to be considered by later systems. Using the contexts in which they appear, their system renders a hierarchical clustering of all words in the corpus. In theory, each cluster could be mapped to a single POS tag in gold standard annotation. Improvements upon this system are described in Meng (2005).

Clark (2003) improved upon the performance of Brown (1992) with a similar clustering system, but his also uses a set of prior probabilities that biases the system to cluster together those words having similar morphology. A Hidden Markov Model (HMM) models the first and last three characters of strings and places those with similar substrings into the same cluster.

After the clustering system of Biemann (2006), most researchers developed a penchant for Bayesian systems that label rather than cluster, an effort which really began with the work of Merialdo (1994) who tried using a trigram HMM to tag text.

Goldwater and Griffiths (2007) constructed an HMM-based system using Dirichlet priors for both transition and emission probabilities. Johnson (2007) tried using variational Bayesian Estimation Maximization. Graca et al. (2009) constrained posterior probabilities. Ravi and Knight (2009) use input a dictionary to their system to tag, and so their approach is more semi-supervised. Berg-Kirkpatrick et al. (2010) input morphological features to their HMM in a log linear fashion, and they obtain the best performance of a Bayesian system thus far according to Christodoulopoulos et al . (2010).

Christodoulopoulos et al. (2010) also include some surprising findings in their evaluation of POS induction systems, stating that the older clustering systems of Brown et al. (1992) and Clark (2003) actually perform better at POS induction than do the more recent, Bayesian systems. This is according to V-Measure (Rosenberg and Hirschberg, 2007), which they find to be the most stable metric over varying sizes of tagsets. They also evaluate the performance of Haghighi and Klein's (2006) prototype-driven POS inducer, which was originally designed to run with hand-coded prototypes of each cluster as features of the system. Christodoulopoulos et al. (2010) derived prototypes from other POS induction systems to feed to Haghighi and Klein's system, and found that the clustering systems also produced the best prototypes. Curiously, however, the very best prototypes came from Brown et al. (1992) instead of Clark (2003). This is a surprising finding since Clark (2003) uses morphological information to cluster, but Brown et al. (1992) uses no such knowledge.

IV. Experiments

Our experiments are a continuation of the work of Christodoulopoulos et al. (2010). We want to try novel methods of prototype derivation that may further improve performance in POS induction. We also want to vary several parameters of our experiments, in hopes that doing so may tease out why the morphologically-unaware clustering system of Brown et al. (1992) yielded better-performing clusters than did Clark (2003).

We derived prototypes from the systems of Brown et al. (1992) (brown92 henceforth) and Clark (2003) (clark03 henceforth) varying several parameters. We then pass these derived prototype sets as features to the system used in Haghighi and Klein (2006) (H&K henceforth) to observe contributions of the different prototype sets to performance in POS induction. We use as our data the version of the Brown corpus included with – and annotated in the style of – the third release of the Penn Treebank (PTB3) (Marcus et al., 1993).

IV.1 Systems used for derivation of prototypes – brown92 and clark03

We consider the outputs of two systems as sources for prototypes: brown92 and clark03. As mentioned above, these two systems cluster their outputs– that is, every type is assigned to a single POS tag (cluster), and so every instance of that type in a testing corpus will be labeled in exactly the same way. This is in contrast to labeling systems, which produce a tag for each token in the testing corpus, often as a result of probabilities modeled as an Hidden Markov Model (HMM) or some variation thereof. The motivation for considering the outputs of clustering systems versus those of labeling systems comes from Christodoulopoulos et al. (2010), in which study they find that the older clustering systems actually perform better at POS tagging on the WSJ portion of the Penn Treebank (Marcus et al., 1993) as well on the Multext East corpus (Erjavec, 2004). For clustering systems, they found that Clark's system performed better than Brown's. Clark's system is similar to Brown clustering, but uses additional prior probabilities about morphology, modeling strings of letters with an HMM. Intuitively, the additional morphological information adds to the specific information about the tokens, which the clustering systems have to work with, and so the greater performance of Clark's system over Brown's makes sense. Surprisingly, however, Christodoulopoulos (2010) found that when it comes to producing prototypes for Haghighi and Klein's (2006) system, the clusters produced by Brown's system yield better performing prototypes than do the clusters produced by Clark's. Why this may be the case is left as an open question. We hoped teasing out an answer to this question would be possible from the results of our study.

IV.2 The number of clusters a system outputs – 45 or 13

The systems of both Brown et al. (1992) and Clark (2003) require a finite number of clusters to be input as a parameter – each system must know how many clusters it is looking for before it begins to categorize the types within the text. In the case of the PTB3 Brown corpus, there are 44 atomic tags used to label POS. We do not explicitly consider any tags which have been hyphenated, since hyphenation is intended to capture annotator uncertainty or finer-grained complication of categorization; if a token is given a hyphenated tag in the corpus, we map it into a 45th category – unknown (UNK). In half of our experiments, therefore, we have the brown92 and clark03 systems divide the types within the Brown corpus into 45 clusters, which can then be evaluated against the original annotation of the PTB3 Brown corpus.

We also consider having the systems categorize the corpus types into 13 clusters. The 13 clusters are inspired by those used in Christodoulopoulos et al. (2010), but we have slightly changed the mapping from the original 45 PTB3 tags (see table 1); most notably, POS (a tag for possessive markers) which was mapped into its own category, has been instead mapped to ADJ in order to reflect the tendency of possessive markers to describe nouns (e.g. *Mom's car* describes a car in parallel to *blue car*). Once again, any hyphenated tags have been mapped into the 13th category, UNK.

NUM	←	CD
ADJ	←	JJ JJR JJS PRP$ $ # POS
CONJ	←	CC
PUNCT	←	. `` ('') , : LS SYM UH
PRT	←	RP
TO	←	TO
V	←	MD VBD VBP VB VBZ VBG VBN
ADV	←	RB RBR RBS
DET	←	DT PDT
N	←	EX FW NN NNP NNPS NNS PRP
PREP	←	IN
W	←	WDT WP$ WP WRB

Table 1: Mapping of the 45 PTB3 POS tags to the reduced 13-tag set.

The motivation for evaluating prototypes from different numbers of clusters stems first of all from the fact that POS tagging is a subjective exercise for the annotator. Francis and Kucera (1964) can specify a high degree of granularity at the release of the Brown corpus with a set of over 180 POS tags. Later, in PTB3, this tagset is reduced to 45 atomic tags. One could also re-label the corpus using the twelve "universal" tags described in Petrov et al. (2011). Indeed, with some metrics, performance seems to increase simply because the number of POS tags into which the text is classified increases. More linguistically relevant perhaps is the fact that, regardless of whether a universal tagset is feasible. The available annotated testing data in English and in other languages are tagged using tagsets that differ in number and specification. Some tags used in the Multext East corpus to capture the high inflectional and derivational morphology of Hungarian have no relevance to the relatively poor morphology of English. With tagsets varying so much among available annotated data, it is important to tease out the effects of number of tags from the overall performance of the systems.

IV.3. Methods of prototype derivation

In Haghighi and Klein (2006), the prototype features passed to their system are hand-coded from the gold standard annotation. For a given POS tag of the Penn Treebank, the prototypes for that tag are the three most frequent tokens in the corpus annotated with that tag and not annotated with any other tag. For example, the prototypes encoded for VBD – verb in the past tense – were *said*, *had*, and *was*. Christodoulopoulos et al. (2010) were the first to try to derive prototypes without prior linguistic knowledge. They derived their prototypes from the output of each system using a measurement of similarity. This similarity measurement was based on that of Haghighi and Klein (2006). It uses Singular Value Decomposition on word context vectors and cosine similarity. They consider drawing prototypes from those strings that have both high similarity to other strings within the same cluster as well as high total dissimilarity (= 1 – similarity) to strings in other clusters. The selected prototypes are then passed as features to Haghighi and Klein's system. Passing these prototypes to Haghighi and Klein's system yields better performance in clustering than does running either Brown's or Clark's system on its own (see below for metrics).

IV.3.1 Methods of prototype derivation – C

For this study, we derive prototypes from the outputs of these system using three simplified approaches. For the first derivation approach, we take as prototypes those words in the clusters which appear most frequently in the corpus. So, the prototypes from cluster 25, for example, would start with the word in the cluster which has the highest count in the corpus, *said*. The word with the next highest corpus count in this cluster is *made*, and so *made* is the second prototype. The third prototype is *came*, since it is the cluster member that appears third most frequently. This process continues until we have selected our target number of prototypes. In this paper, the notation used for this derivation approach is C.

IV.3.2 Methods of prototype derivation – CcosCBOW, CcosSG

For the other prototype derivation approaches, we find a seed word in each cluster, and then derive as prototypes those words which have the greatest cosine similarity to that seed. Using Word2Vec (Mikolov, 2013) we can build a word vector model that returns a cosine similarity measure for each word in the corpus. Cosine similarity is a value between 0 and 1 that measures the angle between two words in an N-dimensional vector space. More intuitively, two words will have a cosine similarity closer to 1 if they appear in similar contexts. For example, the cosine similarity of *cat* and *dog* will be closer to 1 than the cosine similarity of *cat* and *walk*. This is because *cat* and *dog* both have high probabilities of filling in the blank in the context *The brown* ____ *ran*, whereas *walk* has a low probability of appearing in that same context. Using Word2Vec, we can build two kinds of models from which to obtain cosine similarities: the continuous bag of words, or CBOW, and the skipgram model, SG. Put simply, CBOW models words in a corpus as contexts of adjacent words. So for each word w_i, and given a window size n, the context modeled for that word will be w_{i-n}… w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}… w_{i+n}. SG models on the other hand will model w_i as words appearing in the context of w_i but not necessarily adjacent to w_i. These two kinds of models will calculate different cosine similarities for the same words from the same corpus. For our purposes, we are interested in the kind of prototypes derived from the cosine similarities of one model versus those of the other model.

We have briefly described cosine similarity and CBOW and SG models. To repeat from above, our second approach to finding prototypes utilizes the cosine similarity between words in a cluster to derive prototypes from that cluster. We find a seed word from each cluster, and then take as prototypes those words in the cluster which have the highest cosine similarities to the seed. We take as a seed the word in the cluster with the highest corpus count. Notice that the word in the cluster with the highest cosine similarity to the seed is the seed itself. Therefore, the seed is the first prototype. Notice also that the first prototype derived by this approach is exactly the same as the first one derived by the C approach – it is the word with the highest corpus count. Having found our seed, we then take the word with the highest cosine similarity to that seed as the next prototype. The word with the next highest cosine similarity to the seed is the third prototype and so on until we have obtained the desired number of prototypes. So, cluster 25 will have the same first prototype as in the C approach, *said*, followed by *replied* and then *asked* as these are the next two words with highest cosine similarity to *said*. In this paper, we notate these approaches as CcosCBOW if the cosine similarity is taken from a CBOW model, and we notate as CcosSG if the cosine similarity is taken from a SG model.

IV.3.3 Methods of prototype derivation – cosCBOW, cosSG

Our third approach to deriving prototypes from system output is similar to our second approach. We still consider the cosine similarity of words in a cluster to that cluster's seed, but the criteria for a seed are now different. In this third approach, the seed is that word in a cluster that has the highest total cosine similarity to all other words within the cluster. So for example, say we have a cluster with four words as its members, *A*, *B*, *C*, and *D*. The cosine similarity of A with B is 0.2, of A with C is 0.8, and of A with D is 0.4. The total cosine similarity of *A* would be then be the sum of these, 1.4. Total cosine similarity is calculated for each word in the cluster, and the word with the highest cosine similarity is then that cluster's seed. The prototypes are then derived similarly via cosine similarity to the seed as was done for CcosCBOW and CcosSG. Once again, since the cosine similarity of the seed to itself is 1, the cluster seed is the first prototype selected. In this paper, the approaches using total cosine similarity to select a seed are notated as cosCBOW for cosine similarities given by a CBOW model, and as cosSG for cosine similarities given by a SG model.

IV.3.4 Number of prototypes – 3, 5, or 7

We have been silent about how many prototypes are selected using the approaches above. Since how many prototypes passed to Haghighi and Klein's system can have an effect on performance, we tried the top 3, 5, and 7 prototypes yielded from each approach described above. In this paper, we notate the use of 3, 5, or 7 prototypes by adding _3, _5, or 7 to the method notations given so far. For example, C_3 would mean the top three prototypes per cluster derived by the counts only approach. Also, CcosCBOW_7 would mean the top seven prototypes per cluster derived by the cosine similarity approach with most frequent word as seed.

IV.4 Evaluation

The best way to evaluate POS induction systems is by no means clear. The basic goal in evaluating such a system is something like this: we have a gold standard set of tagged data, in which each token has been annotated with a POS tag. We also have the output of a system, which assigns a label to each token in the corpus. We want to in some way compare the labeling of the system with the labeling of the gold standard annotation. If the system is supervised or semi-supervised, then the labeling schema (i.e. the gold standard tagset) could be passed to the system ahead of time. A reasonable question then is, *How many tokens labeled "NN" in the gold standard were also labeled with "NN" by the system – and how many tokens labeled NN by the system were labeled as something else in the gold standard?* An unsupervised system, on the other hand, will be wholly unaware of the gold standard, and will produce its own labels, generally in the form of integers. In this case, we might want to ask the question, *Do all tokens which the system labeled "1" correspond better with the tokens labeled "NN" by the gold standard or with those labeled "VBZ"?* Working with systems that cluster types rather than with those that label tokens, we are already at a disadvantage. Clustering systems must decide whether every instance of *jump* is an instance of NN, of VBZ, or of some other tag. Assuming the annotator has recognized that some instances of *jump* are nouns while others are verbs, the output of the clustering system is guaranteed to not be 100% accurate when compared with the gold standard – every instance of the same word will be labeled with exactly the same tag. With this in mind, it's worth reiterating the counterintuitive finding of Christodoulopoulos et al. (2010), that older POS induction systems that cluster types perform better than newer ones that label tokens. Again, this finding is one reason why we consider the clustering systems of brown92 and clark03 in this study.

As data, we use the version of the Brown Corpus included with PTB3. Using all of the unlabeled Brown corpus data as input, we derive prototypes using the various approaches described above. We then pass those prototypes as features to the H&K induction system. The H&K system requires two training sets of data: one for training a similarity model and another for training a sequence model. It also requires a test set to be tagged via the models and to be submitted for evaluation. We systematically select 2000 sentences of the Brown corpus for the test set and 5000 for the sequence model training set. The remaining 45,107 sentences are reserved for training the similarity model, which benefits from more training data than the sequence model.

We use three metrics[1] to assess the system's performance in producing tagged text: many-to-one mapping (M-1), one-to-one mapping (1-1), and V-Measure (VM). While many different metrics have been used to assess the performance of POS induction systems, these three metrics together provide a decent level of intuitive evaluation as well as stable evaluation across different parameters. Each of these metrics typically makes a reference to *clusters*, but it should be noted that in the context of these metrics, *cluster* refers to all those tokens that have been tagged in the same way. In this way, the metrics can refer even to the outputs of labeling POS induction systems in terms of clusters. In the case of clustering POS induction systems, the two senses of cluster would be analogous. To avoid confusion, when referring to the clusters considered by the metrics, we will use *category* where other literature may use *cluster* instead.

M-1 is an often-used metric that allows for multiple categories to be mapped to a gold standard tag. The greater the value of M-1, the better, and this value tends to increase with the number of categories a system outputs to the point that, if every token is assigned its own category, M-1 could be 100%. It is clearly undesirable,

[1] Source code for evaluation metrics can be found at http://christos-c.com/resources.html

however, to map every token to its own category. We include this metric for comparability with other studies that use it.

The 1-1 metric maps each category to only one gold standard tag. This is a more restrictive test than M-1, but is perhaps more intuitive to the goal of this study. We want each category (i.e. cluster) produced by brown92 or clark03 to match closely with one and only one POS tag from the gold standard annotation. This metric assesses how accurately this mapping takes place.

VM is analogous to F-measure. Where F-measure weights the performance metrics of precision (percentage of retrieved items that are relevant) and recall (percentage of relevant items retrieved), VM weights homogeneity and completeness. Homogeneity is satisfied if *only* tokens labeled with a single tag are assigned to a single category – thus it is a variation on precision; completeness is satisfied if *all* tokens labeled with a single tag are assigned to a single category – thus it is a variation on recall (Rosenberg and Hirschberg, 2007). The appeal of VM as a metric for this task is its relative stability among different values of the parameters that we are altering in our experiments (Christodoulopoulos et al. 2010). We weight homogeneity and completeness equally.

V. Results

In total, we ran 60 experiments, deriving prototypes 30 times from brown92 and 30 times from clark03. These experiments differed over the parameters of number of clusters input to the system {45, 13}, method of prototype derivation {C, CcosCBOW, CcosSG, cosCBOW, cosSG}, and number of prototypes derived {3, 5, 7}. The results from all experiments are given in table 2. Baseline metrics from brown92, clark03, and H&K are also given in table 2. These are the results obtained by labeling the 2000 test sentences according to the clusters output by brown92 and clark03 – the performance of these systems alone without any further derivation of prototypes to be passed to H&K. The H&K baseline uses the 45 gold standard prototypes used in Haghighi and Klein (2006). No set of 13 comparable prototypes is provided. Any experiment labeled *FAIL* indicates that H&K, for reasons unknown, did not complete its computation when given the set of prototypes. In almost all cases, the best performance is the baseline. In all cases, the best performance of any prototype-driven approach is one that utilizes the counts of words to derive prototypes without using the cosine similarity of those words in any way. Surprisingly, the VM scores from our best-performing prototype systems actually surpass the scores from H&K using the gold standard prototypes. It is possible that this is a result of the H&K prototypes having been hand produced from the WSJ portion of the Penn TreeBank, and if hand-coded prototypes had been coded from the PTB3 Brown corpus in the same way, perhaps results would have been similar to those of Haghighi and Klein (2006).

45 clusters (Brown '92)

	M-1	1-1	VM
BASELINE	**67.639**	**55.230**	**66.046**
C_3	63.146	50.719	57.973
C_5	**64.157**	**52.230**	**61.283**
C_7	63.787	50.761	59.324
C_cbow_3	56.509	42.546	50.933
C_cbow_5	60.098	46.051	53.926
C_cbow_7	63.354	50.982	58.213
C_sg_3	59.936	45.788	53.274
C_sg_5	61.128	47.604	55.092
C_sg_7	63.010	51.245	58.322
cbow_3	FAIL	FAIL	FAIL
cbow_5	FAIL	FAIL	FAIL
cbow_7	40.438	35.338	40.861
sg_3	20.151	19.124	16.995
sg_5	31.284	29.015	29.466
sg_7	31.505	30.323	29.945

13 clusters (Brown '92)

	M-1	1-1	VM
BASELINE	**69.213**	**54.872**	**53.695**
C_3	54.936	34.824	33.172
C_5	**63.559**	**53.929**	**46.542**
C_7	47.859	34.849	32.311
C_cbow_3	53.939	34.666	35.204
C_cbow_5	55.137	34.507	37.690
C_cbow_7	56.792	38.532	41.206
C_sg_3	54.696	35.138	34.217
C_sg_5	57.922	41.847	39.717
C_sg_7	58.883	45.462	42.740
cbow_3	33.357	33.252	9.008
cbow_5	42.825	42.633	23.143
cbow_7	40.087	39.863	20.783
sg_3	FAIL	FAIL	FAIL
sg_5	32.764	32.752	7.844
sg_7	FAIL	FAIL	FAIL

45 clusters (Clark '03)

	M-1	1-1	VM
BASELINE	**75.049**	**53.961**	**68.009**
C_3	59.029	46.578	52.900
C_5	**65.663**	**50.826**	58.366
C_7	65.127	50.524	**60.127**
C_cbow_3	55.805	46.005	50.408
C_cbow_5	60.063	47.739	53.344
C_cbow_7	60.958	48.138	54.238
C_sg_3	57.120	47.596	51.585
C_sg_5	60.836	49.371	55.165
C_sg_7	62.379	50.532	56.450
cbow_3	26.224	25.500	23.585
cbow_5	31.792	30.406	31.098
cbow_7	FAIL	FAIL	FAIL
sg_3	29.582	27.880	29.238
sg_5	26.052	25.174	26.794
sg_7	29.451	27.856	31.976

13 clusters (Clark '03)

	M-1	1-1	VM
BASELINE	**64.548**	**44.511**	**51.381**
C_3	55.127	**48.395**	38.255
C_5	57.800	41.400	40.520
C_7	**58.972**	41.597	**42.121**
C_cbow_3	47.911	39.307	33.354
C_cbow_5	48.865	40.619	33.055
C_cbow_7	48.960	39.447	32.731
C_sg_3	42.859	32.802	25.052
C_sg_5	51.036	46.515	35.477
C_sg_7	51.947	51.947	45.707
cbow_3	FAIL	FAIL	FAIL
cbow_5	34.274	34.136	13.815
cbow_7	33.102	31.940	9.751
sg_3	32.792	32.778	7.934
sg_5	32.999	25.518	6.536
sg_7	33.009	28.091	9.493

H&K GS prototypes (45)

M-1	1-1	VM
58.715	56.349	55.117

Table 2: Results of all 60 experiments as well as baseline scores for many-to-one, one-to-one, and V-measure metrics

VI. Discussion

The results presented in table XX do not encourage the discovery of prototypes via the methods used in this study. Some of the 1-1 results come tantalizingly close to the baseline, and in one case the baseline is surpassed (C_3 for clark03 with 13 clusters). In all cases, VM is greater for the baseline than with the use of any prototype set at all.

First a general note on the 1-1 metric versus VM. In our case, there are never more clusters produced by the system than there are tags in the gold standard. Therefore, a similar performance in 1-1 is likely to indicate that, token for token, a near equal number of tokens were given a "correct" tag. Put another way, when mapped to a tag, the cluster contained enough words that were labeled with that tag in the gold standard. With enough clusters and enough tags, a certain percentage of accuracy can be attained which is similar to another clustering and the same set of tags. Where VM surpasses 1-1 as a metric is in the fact that VM can give an idea of the quality of the cluster output itself given the original "clustering" of tags among tokens in the original annotation. The bottom line is that VM is likely the most stable metric and most indicative of good performance. Our analysis moves forward with the result of VM from each experiment.

We tried deriving prototypes from 45 and 13 clusters. In all cases, VM scores are greater for 45 clusters than for 13. This is consistent with the findings of Christodoulopoulos et al. (2010). It may simply be the case that more clusters means that more prototypes will be passed to H&K as features, lending to the overall accuracy of placing words into their target clusters. Deriving prototypes from more than 45 clusters may test this claim further. The original Brown corpus was tagged with over 180 tags. It may be informative to observe the performance of H&K when passed prototypes from as many clusters.

We tried using both brown92 and clark03 to derive prototypes. Performance did not improve over the baseline for either system through any use of prototypes. Christodoulopoulos et al. (2010) reported improvement over the baseline of these systems on WSJ data. This may be a result of the prototype derivation methods, which differs between our work and theirs, as well as the data. Consistent with this study, however, is the fact that while the clark03 VM baseline is greater than the brown92 VM baseline, the greatest VM score from clark03 with any prototype set is less than VM from brown92 with any prototype set. Morphological information is used by clark03 but not by brown92. So intuitively, we might expect clark03 to perform better on clustering words – which it does – and also to produce more informative prototypes – but it does not. Further inquiry should be made as to why this is the case.

We tried taking the most frequent cluster tokens as prototypes as well as taking those tokens which were greatest in cosine similarity to a given seed. The prototypes based on simple corpus counts performed better in every case. Still, the fact that there is no improvement over the baseline in any case suggests that the similarity/dissimilarity mean used by Christodoulopoulos et al. (2010) may still be a better approach to derivation of prototypes from clusters since they did report improvement over a similar baseline.

We tried passing sets of 3, 5, and 7 prototypes to H&K from each of our approaches. When prototypes were derived based on corpus counts alone, the best performance often capped at 5 prototypes per cluster. There may be reason to think that 5 is somehow just informative enough to the system, however, when we look at the results from other prototype derivation methods – i.e. those involving some measure of cosine similarity – we see that performance actually increases when larger numbers of prototypes are passed to H&K. The number of prototypes passed to the system, therefore, appears to depend on the type of approach with which they were derived. The ideal number of prototypes may be studies further by passing greater numbers of prototypes to H&K that were derived using the similarity/dissimilarity approach of Christodoulopoulos et al. (2010).

VII. Conclusion

We performed POS induction using the system of Haghighi and Klein (2006). As was performed in Christodoulopoulos et al. (2010), we did not use hand-coded prototypes, but tried to derive prototypes from other POS induction systems, namely those of Brown et al. (1992) and Clark (2003). We used novel approaches to prototype derivation considering the corpus counts of words in clustered outputs as well as cosine similarity to other words within the same cluster. Using the PTB3 version of the Brown corpus as data, we were unable to gain any performance over baseline runs of the Brown et al. (1992) or Clark (2003) systems. We were able to attain better performance using our prototypes than the hand-coded ones used in Haghighi and Klein (2006), suggesting that searching for novel ways to derive prototypes is worth the effort for the sake of performance as well as for rendering a completely unsupervised approach to prototype-driven POS induction.

Works Cited