POS tags. For instance, since we have ten POS tags for English, we will train 45 binary classifiers.[8] To determine the POS tag of a given English word *w*, we will use these 45 pairwise classifiers to independently assign a label to *w*. For instance, the NN-JJ classifier will assign either NN or JJ to *w*. We then count how many times *w* is tagged with each of the ten POS tags. If there is a POS tag *t* whose count is nine, it means that all the nine classifiers associated with *t* have classified *w* as *t*, and so our method will label *w* as *t*. Otherwise, we remove *w* from our seed set, since we cannot confidently label it using our classifier ensemble.

To create the training set for the NN-JJ classifier, for instance, we can possibly use all of the words labeled with NN and JJ as positive and negative instances, respectively. However, to ensure that we do not have a skewed class distribution, we use the same number of instances from each class to train the classifier. More formally, let I_{NN} be the set of instances labeled with NN, and I_{JJ} be the set of instances labeled with JJ. Without loss of generality, assume that $|I_{NN}| < |I_{JJ}|$, where $|X|$ denotes the size of the set X. To avoid class skewness, we have to sample from I_{JJ}, since it is the larger set. Our sampling method is motivated by bagging (Breiman, 1996). More specifically, we create 10 training sets from I_{JJ}, each of which has size $|I_{NN}|$ and is formed by sampling with replacement from I_{JJ}. We then combine each of these 10 training sets separately with I_{NN}, and train 10 SVM classifiers from the 10 resulting training sets. Given a test instance *i*, we first apply the 10 classifiers independently to *i* and obtain the signed confidence values[9] of the predictions provided by the classifiers. We then take the average of the 10 confidence values, assigning *i* the positive class if the average is at least 0, and negative otherwise.

As mentioned above, we use distributional features to represent an instance created from a word *w*. The distributional features are created based on Schütze's (1995) method. Specifically, the left context and the right context of *w* are each encoded using the most frequent 500 words from the vocabulary. A feature in the left (right) context has the value 1 if the corresponding word appears to the left (right) of *w* in our corpus, and 0 otherwise. However, we found that using distributional features alone would erroneously classify words like "car" and "cars" as having the same POS because the two words are distributionally similar. In general, it is difficult to distinguish words in NN from those in NNS by distributional means. The same problem occurs for words in VB and VBD. To address this problem, we augment the feature set with suffixal features. Specifically, we create one binary feature for each of the 30 most frequent suffixes that we employed in the previous section. The feature corresponding to suffix *x* has the value 1 if *x* is the suffix of *w*. Moreover, we create an additional suffixal feature whose value is 1 if none of the 30 most frequent suffixes is the suffix of *w*.

6 Augmenting the Seed Set

After purification, we have a set of clusters filled with distributionally and morphologically reliable seed words that receive the same POS tag when predicted independently by morphological features and distributional features. Our goal in this section is to augment this seed set. Since we have a small seed set (5K words for English and 8K words for Bengali) and a large number of unlabeled words, we believe that it is most natural to apply a weakly supervised learning algorithm to bootstrap the clusters. Specifically, we employ a version of self-training together with SVM as the underlying learning algorithm.[10] Below we first present the high-level idea of our self-training algorithm and then discuss the implementation details.

Conceptually, our self-training algorithm works as follows. We first train a multi-class SVM classifier on the seed set for determining the POS tag of a word using the morphological and distributional features described in the previous section, and then apply it to label the unlabeled (i.e., unclustered) words. Words that are labeled with a confidence value that exceeds the current threshold (which is initially set to 1 and -1 for positively and negatively labeled instances, respectively) will be

[8] We could have trained just one 10-class classifier, but the fairly large number of classes leads us to speculate that this multi-class classifier will not achieve a high accuracy.

[9] Here, a large positive number indicates that the classifier confidently labels the instance as NN, and a large negative number represents confident prediction for JJ.

[10] As a related note, Clark's (2001) bootstrapping algorithm uses KL-divergence to measure the distributional similarity between an unlabeled word and a labeled word, adding to a cluster the words that are most similar to its current member. For us, SVM is a more appealing option because it automatically combines the morphological and distributional features.