

TOWARD LANGUAGE-INDEPENDENT MORPHOLOGICAL SEGMENTATION  
AND PART-OF-SPEECH INDUCTION

by

Sajib Dasgupta

APPROVED BY SUPERVISORY COMMITTEE:

---

Dr. Vincent Ng, Chair

---

Dr. Latifur Khan

---

Dr. Yang Liu

Copyright 2007

Sajib Dasgupta

All Rights Reserved

To My Mom and Dad

TOWARD LANGUAGE-INDEPENDENT MORPHOLOGICAL SEGMENTATION  
AND PART-OF-SPEECH INDUCTION

by

SAJIB DASGUPTA, B.S.

THESIS

Presented to the Faculty of  
The University of Texas at Dallas  
in Partial Fulfillment  
of the Requirements  
for the Degree of

MASTER OF SCIENCE IN  
COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December, 2007

## ACKNOWLEDGEMENTS

There are so many people who helped me complete this thesis, it becomes nearly impossible for me to enlist everyone's name and contributions in a single paragraph. Special thanks to my family, specially my mom and dad and my elder brother Rajib, for being part of every bit of my life: their endless motivations, constant mental support and unconditional love were instrumental in whatever I have achieved so far. I still remember, every time I gave them a phone call out of frustrations and bruised state-of-the-mind, their only reply to me was: "There is always a next time, son, your good days are just ahead". They truly are the part and parcel of my good and bad days.

There is no doubt my advisor, Dr. Vincent Ng, comes right next. I would always be grateful for his insightful advices, encouragements and guidance needed to cross the final hurdles. He gave me the independence to pursue my own research, and at the same time provided ideas and support, whenever needed. I still remember the night, when I was writing the paper on the submission date of the NAACL and Dr Ng. was with me all night helping with the writing and inserting all the missing "the"-s in between, and still found some time after the submission (at 5 am in the morning) to discuss what I could have done better and how. I was privileged to work with someone of such knowledge and expertise.

I would also like to thank the Dallas Bangladeshi community, who gave me the necessary support such as transportation to the Walmart and Dollar (50 cents for students, though) movie theatre on Tuesday. Special thanks to all my roommates including Mehedy

Masud, Vinod Prakash, and specially Numair “vhai”, who has been my best friend in Dallas. My life in Dallas would have been very different without this special person.

I would also like to thank Dr. Mumit Khan, Kamrul Haider and Naushad Uzzaman and Zahurul Islam of Centre for Research on Bangla Language Processing (CRBLP), BRAC University, Bangladesh for allowing us to use “Prothom Alo” Corpus and other linguistic resources. In addition, Zeeshan, Rumana, Tahid and Mahbubur of the University of Texas at Dallas helped me several times in creating the annotated Bengali datasets.

Finally, all my friends back home and here in USA deserve special thanks. They are the ones who are always there to spend the time with, and share my joys and sorrows. In the end, I would like to thank almighty God for giving me the strength to achieve whatever I have achieved so far.

August, 2007

TOWARD LANGUAGE-INDEPENDENT MORPHOLOGICAL SEGMENTATION  
AND PART-OF-SPEECH INDUCTION

Publication No. \_\_\_\_\_

Sajib Dasgupta, M.S.  
The University of Texas at Dallas, 2007

Supervising Professor: Dr. Vincent Ng

This thesis addresses two fundamental tasks in natural language processing, namely morphological segmentation and part-of-speech induction. In contrast to existing algorithms developed for these problems, we have proposed a learning system where a morphological analyzer and a part-of-speech lexicon can be built automatically from just a text corpus without using any additional language specific grammatical knowledge. We also give empirical support that our system is totally language independent, i.e., it can be extended to many different languages. In fact, our morphological segmentation algorithm outperforms Goldsmith's *Linguistica* and Creutz and Lagus's *Morfessor* for English and Bengali, and achieves performance that is comparable to the best results for all three PASCAL evaluation datasets on English, Finnish and Turkish. Our unsupervised part-of-speech acquisition system differs from existing bootstrapping algorithms developed for this problem in that it more tightly integrates morphological information with the distributional POS induction framework and adjusts well to languages where distributional features are not reliable enough. Experimental results demonstrate that our approach works well for English and

Bengali, thus providing suggestive evidence that it is applicable to both morphologically impoverished languages and highly inflectional languages.

## TABLE OF CONTENTS

Acknowledgements.....	v
Abstract.....	vii
List of Tables .....	xi
List of Figures.....	xiii
CHAPTER 1 INTRODUCTION .....	1
CHAPTER 2 UNSUPERVISED MORPHOLOGICAL SEGMENTATION.....	5
2.1 Problem Definition and Goals .....	5
2.2 Related Work .....	8
2.3 The Basic Morpheme Induction Algorithm.....	10
2.3.1 Extracting a List of Candidate Affixes .....	10
2.3.2 Detecting Composite Suffixes .....	11
2.3.3 Extracting a List of Candidate Roots .....	16
2.4 Detecting Incorrect Attachments Using Relative Frequency.....	16
2.5 Detecting Incorrect Attachments Using Suffix Level Similarity.....	19
2.6 Inducing Orthographic Rules and Allomorphs.....	22
2.7 Word Segmentation .....	27
2.8 Evaluation .....	29
2.8.1 Experimental Setup.....	29
2.8.2 Results for English and Bengali.....	32
2.8.3 Discussion and Error Analysis.....	33
2.8.4 PASCAL Challenge Results .....	35
CHAPTER 3 UNSUPERVISED PART-OF-SPEECH ACQUISITION.....	38
3.1 Problem Definition and Goals .....	38
3.2 Related Work .....	42

3.3	The English and Bengali Tag Sets .....	44
3.4	Clustering the Morphologically Similar Words.....	45
3.4.1	Creating the Initial Clusters .....	48
3.4.2	Improving the Initial Clusters .....	48
3.4.3	Populating the Clusters with Words .....	51
3.4.4	Labeling and Merging the Initial Clusters .....	52
3.4.5	Generating and Labeling Sub-clusters .....	52
3.5	Purifying the Seed Set.....	55
3.6	Bootstrapping Distributionally Reliable Words.....	60
3.7	Classifying Morphologically and Distributionally Poor Words .....	63
3.8	Evaluation .....	64
3.8.1	Results and Discussion .....	65
3.8.2	Additional Experiments .....	73
CHAPTER 4 CONCLUSIONS.....		79
Bibliography .....		81
VITA		

## LIST OF TABLES

Number	Page
Table 1. Top Scoring Prefixes and Suffixes for English and Bengali	11
Table 2. Examples of Bengali suffixes checked for compositeness. The strength of each suffix is parenthesized. The erroneous suffixes are boldfaced.	15
Table 3. Word-root frequency ratios for English	19
Table 4. Word-root frequency ratios for Bengali	19
Table 5. Hypothesis validation for English and Bengali	19
Table 6. Suffix level similarity checking for segmentation associated with suffix “ion”	22
Table 7. Examples of orthographic rules learned for English	27
Table 8. Results (reported in terms of exact accuracy (A), precision (P), recall (R) and F-score (F))	33
Table 9. F-scores for the PASCAL gold standards	36
Table 10. English Open Class POS Tagset	44
Table 11. Bengali Open Class POS Tagset	44
Table 12. Initial clusters derived from English corpus, $t$ set to 0.4	50
Table 13. Initial clusters derived from Bengali Corpus, $t$ set to 0.4	51
Table 14. POS distribution of morphologically formed clusters	53
Table 15. Morphological clusters after setting the threshold differently	55
Table 16. Distribution of final training and test set	63
Table 17a. POS induction results for English based on word type	67
Table 17b. POS induction results for English based on word type	67

Table 18a. POS induction results for Bengali based on word type	68
Table 18b. POS induction results for Bengali based on word type	68
Table 19. Most common POS confusions after morphological clustering	69
Table 20. Morphological clustering according to secondary key constraints	69
Table 21. Most common POS confusions after the final augmentation	71
Table 22. Performance vs. SVM confidence	75
Table 23a. Our algorithm's performance vs. corpus frequency	75
Table 23b. Clark's algorithm's performance vs. corpus frequency	76
Table 24. Typed vs. tokenized results	76
Table 25. Most frequent tagging results for English	77
Table 26. Results for English after incorporating ambiguous words	78

## LIST OF FIGURES

Number	Page
Figure 1. Suffix Graph comprising suffixes “s”, “er”, “est”, “ed”, “ing” and “less”	48
Figure 2. Morphology and context distribution	60

## CHAPTER 1

### INTRODUCTION

Knowledge-based approaches were the primary way to construct natural language processing (NLP) systems prior to the 1980s. Knowledge-based systems accomplish an NLP task by applying a set of manually designed heuristics. For instance, a knowledge-based noun phrase identification system would be composed of a set of hand-crafted rules for extracting noun phrases from a document. Constructing a knowledge-based NLP system is therefore difficult, time-consuming, and requires a lot of linguistic expertise. In addition, the manually designed heuristics are often language-dependent, which means that a new set of heuristics needs to be designed for every new language encountered.

Statistical and machine learning approaches, which have become popular in the NLP community in the late 1980s, have revolutionized the way NLP research is conducted. The most successful learning approaches for NLP problems are arguably supervised machine learning approaches, in which systems are trained on data manually annotated for a particular task. The often cited advantages of supervised machine learning approaches to NLP are that they require less time and linguistic expertise to construct, and that they are more robust in the face of noisy input in comparison to their knowledge-based counterparts.

While it is true that learning-based systems are robust, it is questionable whether they require less time and linguistic expertise. As mentioned before, supervised learning approaches assume as input the existence of a (typically large) corpus manually annotated for the target task. Hence, as we shift from knowledge-based to machine learning approaches,

the time and linguistic expertise that were required to design heuristics are now being used to annotate the data needed by our supervised learning models. Worse still, learning-based NLP systems that are developed for one language can rarely be applied in a straightforward manner to other languages. This implies that a new corpus needs to be annotated for a given NLP task for each new language encountered.

More recently, techniques have been developed to produce linguistic annotations for resource-scarce languages in an inexpensive manner. One idea is to *project* linguistic annotations from a resource-rich language to a resource-scarce language in a parallel corpus (Yarowsky and Ngai, 2001). This approach is very promising, but its success depends critically on the existence of a (large) parallel corpus - an assumption that is not satisfied for the majority of the world's resource-scarce languages.

The goal of this thesis is to advance the state of the art in natural language technology by developing approaches that require less time and linguistic expertise than existing supervised learning approaches. More specifically, we investigate *unsupervised, language-independent* approaches to NLP in the context of two fundamental text-processing tasks, morphological segmentation and part-of-speech (POS) induction. Briefly, the goal of unsupervised morphological segmentation is to segment a word into morphemes (i.e., the smallest meaning-bearing units of a natural language) that are derived automatically from an unannotated corpus. On the other hand, the goal of unsupervised POS induction is to learn the set of possible POS tags for each word in a language from an unannotated corpus.

Perhaps not surprisingly, developing high-performance unsupervised NLP systems in a language-independent way (i.e., without using any language-specific grammatical knowledge) is a very challenging task. Indeed, in MorphoChallenge 2005 (the 2005

PASCAL Challenge on Unsupervised Segmentation of Words into Morphemes),<sup>1</sup> the participating systems were evaluated on three different languages (namely English, Finnish, and Turkish), but no segmentation algorithm achieves good performance for all three languages, according to the MorphoChallenge organizers. Part of this thesis will be devoted to address this challenge i.e. to develop an unsupervised, language-independent morphological segmentation and POS induction system that works well across languages.

The contributions of this thesis are three-fold:

**High-performance, language-independent morphological segmentation.** We introduce an unsupervised morphological segmentation algorithm that shows robust performance for four languages with different levels of morphological complexity. In particular, our algorithm outperforms Goldsmith's (2001) *Linguistica* and Creutz and Lagus's (2005) *Morfessor* for English and Bengali, and achieves performance that is comparable to the best results for all three PASCAL evaluation datasets. Improvements arise from (1) the use of relative corpus frequency and suffix level similarity for detecting incorrect morpheme attachments and (2) the induction of orthographic rules and allomorphs for segmenting words where roots exhibit spelling changes during morpheme attachments.

**Toward language-independent part-of-speech induction by exploiting morphological and distributional information.** We propose a new bootstrapping approach to unsupervised POS induction. In contrast to previous bootstrapping algorithms developed for this problem, our approach aims to improve the quality of the seed clusters by employing seed words that are both distributionally and morphologically reliable. In particular, we present a novel method for combining morphological and distributional information for seed selection.

---

<sup>1</sup> See <http://www.cis.hut.fi/morphochallenge2005/>.

Experimental results demonstrate that our approach works well for English and Bengali, thus providing suggestive evidence that it is applicable to both morphologically impoverished languages and highly inflectional languages.

**Linguistic resources for Bengali language processing.** As mentioned above, we evaluate our approaches to morphological segmentation and POS induction on Bengali. Owing to the lack of publicly available resources for Bengali, we manually created annotated datasets for evaluation purposes. Hence, one contribution of our work lies in the creation of linguistic resources for Bengali. By making these datasets publicly available,<sup>2</sup> we hope to facilitate the comparison of different unsupervised morphological segmentation and POS induction algorithms and to stimulate interest in Bengali language processing.

The rest of this thesis is structured as follows. Chapter 2 presents our language-independent approach to unsupervised morphological segmentation. In Chapter 3, we present a novel way of combining morphological and distributional information for POS induction. Finally, we conclude with future work in Chapter 4.

---

<sup>2</sup> See <http://www.hlt.utdallas.edu/~sajib>.

## CHAPTER 2

### UNSUPERVISED MORPHOLOGICAL SEGMENTATION

#### 2.1 Problem Definition and Goals

Morphological analysis is the task of segmenting a word into *morphemes*, the smallest meaning-bearing elements of natural languages. For instance, the English word “unforgettable” can be divided into three morphemes: “un”, “forget”, and “able”. Though very successful, *knowledge-based* morphological analyzers operate by relying on manually designed segmentation heuristics (e.g., Koskenniemi (1983)), which require a lot of linguistic expertise and are time-consuming to construct. As a result, research in morphological analysis has exhibited a shift to *unsupervised* approaches, in which a word is typically segmented based on morphemes that are automatically induced from an unannotated corpus.

Unsupervised approaches have achieved considerable success for English and many European languages (e.g., Goldsmith (2001), Schone and Jurafsky (2001), Freitag (2005)). The recent PASCAL Challenge on *Unsupervised Segmentation of Words into Morphemes*<sup>3</sup> has further intensified interest in this problem, selecting as target languages English as well as two highly agglutinative languages, Turkish and Finnish. However, the evaluation of the Challenge reveals that (1) the success of existing unsupervised morphological parsers does not carry over to the two agglutinative languages, and (2) no segmentation algorithm achieves good performance for all three languages.

---

<sup>3</sup> <http://www.cis.hut.fi/morphochallenge2005/>

Motivated by these state-of-the-art results, our goal in this thesis is to develop an *unsupervised* morphological segmentation algorithm that can *work well across different languages*. With this goal in mind, we evaluate our algorithm on four languages with different levels of morphological complexity, namely English, Turkish, Finnish and Bengali. It is worth noting that Bengali is an under-investigated Indo-Aryan language that is highly inflectional and lies between English and Turkish/Finnish in terms of morphological complexity. Experimental results demonstrate the robustness of our algorithm across languages: it not only outperforms Goldsmith’s (2001) *Linguistica* and Creutz and Lagus’s (2005) *Morfessor* for English and Bengali, but also compares favorably to the best-performing PASCAL morphological parsers when evaluated on all three datasets in the Challenge.

The performance improvements of our segmentation algorithm over existing morphological analyzers can be attributed to our extending Keshava and Pitler’s (2006) segmentation method, the best performer for English in the aforementioned PASCAL Challenge, with the capability of handling two under-investigated problems:

**Detecting incorrect attachments.** Many existing morphological parsers incorrectly segment “candidate” as “candid”+“ate”, since they fail to identify that the morpheme “ate” should *not* attach to the word “candid”. Schone and Jurafsky’s (2001) work represents one of the few attempts to address this inappropriate morpheme attachment problem, introducing a method that exploits the semantic relatedness between word pairs. In contrast, we propose two arguably simpler, yet effective techniques that rely on *relative corpus frequency* and *suffix level similarity* to solve the problem.

**Inducing orthographic rules and allomorphs.** One problem with Keshava and Pitler’s algorithm is that it fails to segment words where the roots exhibit spelling changes during attachment to morphemes (e.g., “denial” = “deny”+“al”). To address this problem, we automatically acquire allomorphs and orthographic change rules from an unannotated corpus. These rules also allow us to output the actual segmentation of the words that exhibit spelling changes during morpheme attachment, thus avoiding the segmentation of “denial” as “deni”+“al”, as is typically done in existing morphological parsers.

In addition to addressing the aforementioned problems, our segmentation algorithm has two appealing features. First, it can segment words with any number of morphemes, whereas many analyzers can only be applied to words with one root and one suffix (e.g., DéJean (1998), Snover and Brent (2001)). Second, it exhibits robust performance even when inducing morphemes from a very large vocabulary, whereas Goldsmith’s (2001) and Freitag’s (2005) morphological analyzers perform well only when a small vocabulary is employed, showing deteriorating performance as the vocabulary size increases.

The rest of this Chapter is organized as follows. Section 2.2 presents related work on unsupervised morphological analysis. In Section 2.3, we describe our basic morpheme induction algorithm. We then show how to exploit the induced morphemes to (1) detect incorrect attachments by using relative corpus frequency (Section 2.4) and suffix level similarity (Section 2.5) and (2) induce orthographic rules and allomorphs (Section 2.6). Section 2.7 describes our algorithm for segmenting a word using the induced morphemes. Finally, we present evaluation results in Section 2.8.

## 2.2 Related Work

As mentioned before, the problem of unsupervised and minimally supervised morphological learning has been extensively studied for English and many other European languages. In this section, we will give an overview of the three major approaches to this problem.

One common approach to unsupervised morphological learning is to first identify morpheme boundaries and then use the segmented words to extract a list of morphemes. For instance, Harris (1955) develops a strategy for identifying morpheme boundaries that checks whether the number of different letters following a sequence of letters exceeds some given threshold. Hafer and Weiss (1974) improve Harris's algorithm by proposing 15 different heuristics that depend on successor and predecessor frequencies to identify morpheme boundaries. Their best heuristic achieves a precision of 0.91 and recall of 0.61 on an English corpus of approximately 6200 word types, which is very small compared to the number of word types typically seen in existing literature on unsupervised morphological induction. DéJean (1998) improves Harris's segmentation algorithm by first inducing a list of the 100 most frequent morphemes and then using those morphemes for word segmentation. The aforementioned PASCAL Challenge on Unsupervised Word Segmentation undoubtedly intensified interest in this problem. Among the participating groups, Keshava and Pitler's (2006) segmentation algorithm combines the ideas of DéJean and Harris and achieves the best result on the English dataset.

Another approach to unsupervised morphological learning is based on an application of the Minimum Description Length (MDL) principle. The goal is to find a set of morphemes such that when each word in a given corpus is segmented according to these morphemes, the total length of an encoding of the corpus is minimized. Specifically, the Expectation

Maximization (EM) algorithm is used to iteratively segment a list of words taken from a given corpus using some predefined heuristics until the length of the morphological grammar converges to a minimum. Brent *et al.* (1995) introduce an information-theoretic notion of compression to represent the MDL framework, although the overall aim of their work is to find an appropriate set of suffixes from a corpus rather than the correct morphological analysis of each word. They use the  $n$  most common words in the Wall Street Journal corpus of the Penn Treebank to induce the suffix list, where  $n$  ranges from 500 to 8000. Snover and Brent (2001) later propose a Bayesian Model for MDL that yields very few false suffixes over a wide range of input sizes in English and French. Goldsmith (1997) tries to find the segmentation point of a word based on the probability and length of the hypothesized stems and affixes. In a subsequent paper, Goldsmith (2001) adopts the MDL approach and provides a new information-theoretic compression system that gives a measure of the length of the morphological grammar. He applies his algorithm to English and French and reports accuracies of 82.9% and 83.3% respectively. He also groups together the possible suffixes for each stem, and introduces the signature paradigm that is helpful for determining syntactic word classes (*i.e.*, part-of-speech classes). Motivated by Goldsmith, Creutz (2003) and Creutz and Lagus (2005) propose a probabilistic maximum *a posteriori* formulation that uses prior distributions of morpheme length and frequency to measure the goodness of an induced morpheme. They work on English and Finnish (a highly agglutinative language) and report better accuracy than Goldsmith's Linguistica morphological parser.

The last approach, introduced by Freitag (2005), first automatically clusters the words using local co-occurrence information and then induces the suffixes according to the orthographic dissimilarity between the words in different clusters. His segmentation

algorithm achieves a high precision (0.95) when morphemes are induced from an English vocabulary that consists of the 10K most frequent terms in the Wall Street Journal corpus of the Penn Treebank. He also makes the interesting observation that employing a larger vocabulary size (say 20K) for morpheme induction considerably degrades system precision and recall (0.8 and 0.82, respectively).

### 2.3 The Basic Morpheme Induction Algorithm

Our unsupervised segmentation algorithm is composed of two steps: (1) inducing *prefixes*, *suffixes* and *roots* from a vocabulary that consists of words taken from a large corpus, and (2) segmenting a word using these induced morphemes. This section describes our *basic* morpheme induction method.

#### 2.3.1 Extracting a List of Candidate Affixes

The first step of our morpheme induction method involves extracting a list of candidate prefixes and suffixes. We rely on a fairly simple idea originally proposed by Keshava and Pitler (2006) for extracting candidate affixes. Assume that  $\alpha$  and  $\beta$  are two character sequences and  $\alpha\beta$  is the concatenation of  $\alpha$  and  $\beta$ . If  $\alpha\beta$  and  $\alpha$  are both found in the vocabulary, then we extract  $\beta$  as a candidate suffix. Similarly, if  $\alpha\beta$  and  $\beta$  are both found in the vocabulary, then we extract  $\alpha$  as a candidate prefix.

The above affix induction method is arguably overly simplistic and therefore can generate many spurious affixes. To filter spurious affixes, we (1) score each affix by multiplying its *frequency* (i.e., the number of distinct words to which each affix attaches) and

its *length*<sup>4</sup>, and then (2) retain only the  $K$  top-scoring affixes, where  $K$  is set differently for prefixes and suffixes. The value of  $K$  is somewhat dependent on the vocabulary size, as the affixes in a larger vocabulary system are generated from a larger number of words. For example, we set the thresholds to 70 for prefixes and 50 for suffixes for English; on the other hand, since the Finnish vocabulary is almost six times larger than that of English, we set the corresponding thresholds to be approximately six times larger (400 and 300 for prefixes and suffixes respectively).<sup>5</sup> Table 1 shows the 10 top scoring prefixes and suffixes generated for English and Bengali.

**Table 1.** Top Scoring Prefixes and Suffixes for English and Bengali

English		Bengali	
Prefix	Suffix	Prefix	Suffix
re	s	bI (বি)	Er (ের)
un	ing	a (অ)	kE (কে)
over	ed	p~rTI (প্রতি)	r (র)
dis	ly	mhA (মহা)	o (ও)
under	d	p~r (প্র)	I (ি)
non	ers	SU (সু)	Sh (সহ)
in	ness	@ (আ)	E (ে)
de	er	bIs~b (বিশ্ব)	dEr (দের)
mis	es	bA (বা)	TE (তে)
inter	ment	sIk~FA (শিক্ষা)	gUIO (গুলো)

### 2.3.2 Detecting Composite Suffixes

Our second extension to the basic morpheme induction algorithm involves the detection of **composite suffixes**. A composite suffix is a suffix formed by combining multiple suffixes.

---

<sup>4</sup> The dependence on frequency and length is motivated by the observation that less-frequent and shorter affixes (especially those of length 1) are more likely to be erroneous (see Goldsmith (2001)).

<sup>5</sup> Since this method for setting our vocabulary-dependent thresholds is fairly simple, the use of these thresholds should not be viewed as rendering our segmentation algorithm language-dependent.

For instance, “তাকে”<sup>6</sup> (TAkE) is a composite suffix in Bengali that comprises “ত” (TA) and “কে” (kE) (like “ers” in English which is formed by “er” and “s”). However, not all suffixes formed by combining multiple suffixes are composite. For instance, “ের” (Er) is a **non-composite** suffix in Bengali, even though it comprises the two simple suffixes “এ” (E) and “র” (r) (like “ent” in English, which cannot be segmented into “en”+“t”).

Our goal is to detect and remove composite suffixes from the list of morphemes induced using our basic algorithm, because their presence can produce incorrect segmentation of words. For example, if the composite suffix “ers” is present in the induced morpheme list, then “walkers” will be erroneously segmented as “walk”+“ers” (note: the correct segmentation is “walk”+“er”+“s”). The reason is that the presence of the composite suffix causes the segmentation algorithm to believe that “ers” is a non-divisible unit, leading to under-segmentation. Composite suffix detection is particularly important for Bengali, as it helps segment the Bengali verbs correctly.

Now the question is: how to detect a composite suffix? Not all strings that can be segmented into two suffixes are actually composite suffixes. As we have seen before, “Er”, “E” and “r” all are valid suffixes in Bengali but “Er” is not a composite suffix. Hence, we need a more sophisticated method for detecting composite suffixes. Specifically, our method posits a suffix as a composite suffix if both of the following criteria are satisfied.

**Suffix strength.** This criterion is motivated by the observation that, given a composite suffix  $a$  formed by combining two suffixes  $a_1$  and  $a_2$ , the strength of  $a$  (*i.e.*, the number of different

---

<sup>6</sup> Throughout this thesis, we use the Romanized transliteration for Bengali, which is almost phonetic. For example, ‘অ’ is ‘a’, ‘আ’ is ‘@’, ‘া’ is ‘A’, ‘ক’ is ‘k’, ‘ট’ is ‘t’, ‘ত’ is ‘T’, ‘থ’ is ‘th’, etc. We have used ‘~’ for Halant in Bengali. Our transliteration mapping table is shown in our data distribution site at <http://www.hlt.utdallas.edu/~sajib/dataset.html>.

words to which  $a$  attaches) should be smaller than the minimum of the strength of  $a_1$  and the strength of  $a_2$ . As an example, consider the composite suffix “fullness” (“full”+“ness”) in English. The number of words to which “full” or “ness” attaches is far greater than the number of words to which “fullness” attaches in a naturally-occurring corpus. Consider the *non-composite* Bengali suffix “Er”. It attaches to 9817 word types in our corpus, but its component suffix “E” only attaches to 6218 words. Hence, this suffix violates the suffix strength criterion and is correctly predicted to be non-composite. However, there are suffixes like “AT” and “Ar” (see the right column of Table 2) that satisfy the suffix strength criterion and yet are not composite. This illustrates why using suffix strength alone is not sufficient for determining the compositeness of a suffix.

**Word-level similarity.** This criterion is motivated by the observation that, if a composite suffix (AB) attaches to a word  $w$ , then it is highly likely that the first component suffix A will also attach to  $w$ . In other words, AB and A should be similar in terms of the words to which they attach. For example, if the composite suffix “ers” attaches to an English word (*e.g.*, “sing”), then its first component suffix “er” should attach to the same word. This property does not hold for non-composite suffixes, however. For instance, while the non-composite suffix “ent” attaches to words such as “absorb”, its first component suffix “en” does not. Given this observation, we can detect composite suffixes by first computing the similarity between a suffix (AB) and its first component suffix (A) as follows:

$$\text{Similarity} (AB, A) = P(A | AB) = \frac{|W'|}{|W|}$$

where  $|W'|$  is the number of words to which both AB and A attach, and  $|W|$  is the number of words to which AB attaches.

In other words, the similarity between the two suffixes, AB and A, is the probability of seeing A conditioned on seeing AB. If this probability is greater than some threshold (we set it to 0.6) and the first criterion (*i.e.*, suffix strength) is satisfied, then we posit AB as a composite suffix. One advantage of the above probabilistic metric is that it can potentially be used to select the best segmentation of a word among multiple candidates. For example, “েরই” (Eri) is a composite suffix in Bengali that can be segmented as either “E”+“ri” (the incorrect segmentation) or “Er”+“i” (the correct segmentation). Since the similarity between “Eri” and “Er” (0.979) is greater than that between “Eri” and “E” (0.739), “Er”+“i” is more likely to be the correct segmentation of “Eri”.

Most importantly, composite suffix detection has enabled us to segment many Bengali verbs with complex morphology correctly. For example, the actual segmentation of the verb “হাটছিলাম” (hAtCIIAm) is “hAt”+“CI”+“l”+“Am”, where “hAt” is the root, “CI” is the tense (Continuous) marker, “l” is the time (Past) marker, and “Am” is the person (first person) marker. Below we show how our algorithm segments “hAtCIIAm” step by step:

$$\begin{aligned}
 \text{hAtCIIAm} &= \text{hAt} + \text{CIIAm} \\
 &= \text{hAt} + \text{CI} + \text{lAm} && \text{[detection of composite suffix CIIAm]} \\
 &= \text{hAt} + \text{CI} + \text{l} + \text{Am} && \text{[detection of composite suffix lAm]}
 \end{aligned}$$

To investigate how reliable suffix strength and word-level similarity are with respect to detecting composite suffixes, we (1) apply these two criteria to all the suffixes that are concatenations of multiple suffixes, and (2) determine which are composite suffixes and which are not. Results for a randomly selected set of suffixes are shown in Table 2, where the left column lists the suffixes identified by our criteria as composite, and the right column lists the suffixes that are identified as non-composite.

**Table 2.** Examples of Bengali suffixes checked for compositeness. The strength of each suffix is parenthesized. The erroneous suffixes are boldfaced.

Suffixes determined to be composite			Suffixes determined to be non-composite		
Suffix	Division	Similarity	Suffix	Division	Similarity
AkE (220)	A (1764) + kE (6728)	0.954	AT (83)	A (1764) + T (340)	0.45
AnO (98)	A (1764) + nO (160)	0.70	Ar (854)	A (1764) + r (12747)	0.57
Ei (1274)	E (6218) + i (7872)	0.96	IyE (116)	I (1246) + yE (325)	0.53
Eri (445)	Er (9817) + i (7872)	0.979	TA (463)	T (340) + A (1764)	0.038
Tao (82)	TA (463) + o (8213)	0.94	TE (2148)	T (340) + E (6218)	0.057
T~bEr (45)	T~b (62) + Er (9817)	0.91	Tm (85)	T (1246) + m (236)	0.023
dEri (107)	dEr (1958) + i (7872)	0.95	Tr (54)	T (346) + r (12747)	0.07
krNE (27)	krN (84) + E (6218)	0.77	kE (6728)	k (332) + E (6218)	0.015
CEn (259)	CE (335) + n (1478)	0.83	nA (188)	n (1478) + A (1764)	0.4
ECI (34)	E (6218) + CI (144)	0.97	Er (9817)	E (6218) + r (12747)	0.43
bEn (94)	bE (147) + n (1478)	0.82	<b>bE (55)</b>	<b>b (156) + E (6218)</b>	<b>0.47</b>
lAm (120)	l (616) + Am (235)	0.85	<b>bI (81)</b>	<b>b (156) + I (1246)</b>	<b>0.45</b>
lEn (233)	l (616) + En (597)	0.86	<b>c~CII (22)</b>	<b>c~CI (20) + l (616)</b>	<b>0.45</b>

Note that all the entries in the left column are indeed valid composite suffixes in Bengali. In addition, all but the last three entries (“bE”, “bI” and “c~CII”, which are different tense markers in Bengali) in the right column are valid non-composite suffixes. Failure to detect these three and similar tense markers has resulted in incorrect segmentations of present or past continuous and future indefinite forms of Bengali verbs. For example, the word “হাটবে” (“hAtbE”, future tense, third person form of verb “hAt”) is under-segmented as “hAt”+“bE” (note: the correct segmentation is “hAt”+“b”+“E”). The reason why the algorithm fails to detect “bE” as a composite suffix is that there are not enough words in the vocabulary to which the suffix “b” (first person, future indefinite tense form of a verb) attaches, and so the similarity value between “bE” and “b” is low (0.47).

The question, then, is: why are there not enough words in the vocabulary to which the suffix “b” attaches? The reason can be attributed to the fact that “b” is a first-person marker, but the Bengali corpus from which we extracted our vocabulary is composed of news

articles, which are normally written in “Third Person” form. Unless we have a text collection with different verb forms (first, second and third person variations), it would be very difficult to segment Bengali verbs correctly.

### 2.3.3 Extracting a List of Candidate Roots

Finally, we extract a list of candidate roots using the induced list of affixes as follows. For each word,  $w$ , in the vocabulary, we check whether  $w$  is *divisible*, i.e., whether  $w$  can be segmented as  $r+x$  or  $p+r$ , where  $p$  is an induced prefix,  $x$  is an induced suffix, and  $r$  is a word in the vocabulary. We then add  $w$  to the root list if it is not divisible. Note, however, that the resulting root list may contain *compound* words (i.e., words with multiple roots). Hence, we make another pass over our root list to remove any word that is a concatenation of multiple words in the vocabulary.

## 2.4 Detecting Incorrect Attachments Using Relative Frequency

Our induced root list is not perfect: many correct roots are missing due to over-segmentation. For example, since “candidate” and “candid” are in the vocabulary and “ate” is an induced suffix, our root induction method will incorrectly segment “candidate” as “candid”+“ate”; as a result, it does not consider “candidate” as a root. Hence, to improve the root induction method, we need to determine that the attachment of the morpheme “ate” to the root word “candid” is incorrect. In this section, we propose a simple yet novel idea of using relative corpus frequency to determine whether the attachment of a morpheme to a root word is plausible or not.

Consider again the two words “candidate” and “candid”. While “candidate” occurs 6380 times in our corpus, “candid” occurs only 119 times. This frequency disparity can be an

important clue to determining that there is no morphological relation between “candidate” and “candid”. Similar observation is also made by Yarowsky and Wicentowski (2000), who successfully employ relative frequency similarity or disparity to rank candidate VBD/VB pairs (e.g., “sang”/“sing”) that are irregular in nature. Unlike Yarowsky and Wicentowski, however, our goal is to detect incorrect affix attachments and improve morphological analysis.

Our incorrect attachment detection algorithm, which exploits frequency disparity, is based on the following hypothesis: if a word  $w$  is formed by attaching an affix  $m$  to a root word  $r$ , then the corpus frequency of  $w$  is likely to be less than that of  $r$  (i.e., the frequency ratio of  $w$  to  $r$  is less than one). In other words, we hypothesize that the inflectional or derivational forms of a root word occur less frequently in a corpus than the root itself.

To illustrate this hypothesis, Tables 3 and 4 show some randomly chosen English and Bengali words together with their *word-root frequency ratios* (WRFs). The <word, root> pairs in the left side of the table are examples of correct attachments, whereas those in the right side are not. Note that only those words that represent correct attachments have a WRF less than 1.

The question, then, is: to what extent does our hypothesis hold? To investigate this question, we generated examples of correct attachments by randomly selecting 400 words from our English vocabulary and then removing those that are root words, proper nouns, or compound words. We then manually segmented each of the remaining 378 words as Prefix+Root or Root+Suffix with the aid of the CELEX lexical database (Baayen et al., 1996). Somewhat surprisingly, we found that the WRF is less than 1 in only 71.7% of these

attachments. When the same experiment was repeated on 287 hand-segmented Bengali words, the hypothesis achieves a higher accuracy of 83.6%.

To better understand why our hypothesis does not work well for English, we measured its accuracy separately for the Root+Suffix words and the Prefix+Root words, and found that the hypothesis fails mostly on the suffixal attachments (see Table 5). Though surprising at first glance, the relatively poor accuracy on suffixal attachments can be attributed to the fact that many words in English (e.g., “amusement”, “winner”) appear more frequently in our corpus than their corresponding root forms (e.g., “amuse”, “win”). For Bengali, our hypothesis fails mainly on verbs, whose inflected forms occur more often in our corpus than their roots. This violation of the hypothesis can be attributed to the grammatical rule that the main verb of a Bengali sentence has to be inflected according to the subject in order to maintain sentence order.

To improve the accuracy of our hypothesis on detecting correct attachments, we relax our initial hypothesis as follows: if an attachment is correct, then the corresponding WRFR is less than some predefined threshold  $t$ , where  $t > 1$ . However, we do not want  $t$  to be too large, since our algorithm may then determine many incorrect attachments as correct. In fact, we found out that if we set the threshold as 10 and 2 for suffixal and prefixal attachments respectively, our hypothesis holds true for all the word segmentations in the validation set for both English and Bengali. Taking into account these considerations, we use a threshold of 10 for suffixes and 2 for prefixes for all the languages we consider in this thesis.

Now we can employ our hypothesis to detect incorrect attachments and improve root induction as follows. For each word,  $w$ , in the vocabulary, we check whether (1)  $w$  can be segmented as  $r+x$  or  $p+r$ , where  $p$  and  $x$  are valid prefixes and suffixes respectively and  $r$  is

another word in the vocabulary, and (2) the WRFR for  $w$  and  $r$  is less than our predefined thresholds (10 for suffixes and 2 for prefixes). If both conditions are satisfied, it means that  $w$  is divisible. Hence, we add  $w$  into the list of roots if at least one of the conditions is violated.

**Table 3.** Word-root frequency ratios for English

Correct Parses			Incorrect Parses		
Word	Root	WRFR	Word	Root	WRFR
bear-able	bear	0.01	candid-ate	candid	53.6
attend-ance	attend	0.24	medic-al	medic	483.9
arrest-ing	arrest	0.06	prim-ary	prim	327.4
sub-group	group	0.0002	ac-cord	cord	24.0
re-cycle	cycle	0.028	ad-diction	diction	52.7
un-settle	settle	0.018	de-crease	crease	20.7

**Table 4.** Word-root frequency ratios for Bengali

Correct Parses			Incorrect Parses		
Word	Root	WRFR	Word	Root	WRFR
@SrEr (আসরের)	@Sr	0.17	nArII (নারী)	nAr	556
@bEgE (আবেগে)	@bEg	0.39	JAbTly (যাবতীয়)	JAbT	66
jIbnKE (জীবনকে)	jIbn	0.07	KOIA (খোলা)	KOI	146.7
Apb~jy (অপব্যয়)	b~jy	0.01	jAmAyAT (জামায়াত)	jAmAy	199.2
upjATi (উপজাতি)	jATi	0.03	bAjAr (বাজার)	bAj	364.3
p~rTIdIn (প্রতিদিন)	dIn	0.10	jbAb (জবাব)	jbA	271

**Table 5.** Hypothesis validation for English and Bengali

		Root+Suffix	Prefix+Root	Overall
English	# of words	344	34	378
	WRFR < 1	70.1%	88.2%	71.7%
Bengali	# of words	245	41	286
	WRFR < 1	81.1%	79%	80.7%

## 2.5 Detecting Incorrect Attachments Using Suffix Level Similarity

Many of the incorrect suffixal attachments have a WRFR between 1 and 10, but due to the relaxation of our hypothesis, the detection algorithm described in the previous section will determine all of them as correct attachments. Hence, in this section, we propose another

technique, which we call suffix level similarity, to identify some of these incorrect attachments.

Suffix level similarity is motivated by the following observation: if a word  $w$  combines with a suffix  $x$ , then  $w$  should also combine with the suffixes that are “morphologically similar” to  $x$ . To exemplify, consider the suffix “ate” and the root word “candid”. The words that combine with the suffix “ate” (e.g., “alien”, “fabric”, “origin”) also combine with suffixes like “ated”, “ation” and “s”. Given this observation, the question of whether “candid” combines with the suffix “ate” then lies in whether or not “candid” combines with “ated”, “s” and “ation”. The fact that “candid” does not combine with many of the above suffixes provides suggestive evidence that “candidate” cannot be derived from “candid”.

More specifically, to check whether a word  $w$  combines with a suffix  $x$  using suffix level similarity, we (1) find the set of words  $W_x$  that can combine with  $x$ ; (2) find the set of suffixes  $S_x$  that attach to all of the words in  $W_x$  under the constraint that  $S_x$  does not contain  $x$ ; and (3) find the 10 suffixes in  $S_x$  that are most “similar” to  $x$ . The question, then, is how to define similarity. Intuitively, a good similarity metric should reflect, for instance, the fact that “ated” is a better suffix to consider in the attachment decision for “ate” than “s” (i.e., “ated” is more similar to “ate” than “s”), since “s” attaches to most nouns and verbs in English and hence should not be a distinguishing feature for incorrect attachment detection.

We employ a probabilistic measure that computes the similarity between suffixes  $x$  and  $y$  as the product of (1) the probability of a word combining with  $y$  given that it combines with  $x$  and (2) the probability of a word combining with  $x$  given that it combines with  $y$ . More specifically,

$$PM(x, y) = P(y|x) * P(x|y) = \frac{n}{n_1} * \frac{n}{n_2},$$

where  $n_1$  is the number of distinct words that combine with  $x$ ,  $n_2$  is the number of distinct words that combine with  $y$ , and  $n$  is the number of distinct words that combine with both  $x$  and  $y$ . Note that this metric has the desirable property of returning a low similarity value for “s”: while  $n$  is likely to be large, it will be offset by a large  $n_2$ .

After getting the 10 suffixes that are most similar to  $x$ , we employ them as features and use the associated similarity values (we scale them linearly between 1 and 10) as the weights of these 10 features. The decision of whether a suffix  $x$  can attach to a word  $w$  depends on whether the following inequality is satisfied:

$$\sum_1^{10} f_i w_i > t,$$

where  $f_i$  is a boolean variable that has the value 1 if  $w$  combines with  $x_i$ , where  $x_i$  is one of the 10 suffixes that are most similar to  $x$ ;  $w_i$  is the scaled similarity between  $x$  and  $x_i$ ; and  $t$  is a predefined threshold that is greater than 0.

Table 6 shows the top ranked similar suffixes of the suffix “ion”. Given a segmentation of “reaction” into “react”+“ion”, whose WRFR is in the range of 1 to 10, suffix level similarity checks which suffixes “react” attaches to. As you can see from Table 6, “react” attaches to suffixes like “ive”, “or”, “ors”, “ivity”, which are the most similar suffixes of “ion”. Thus, according to suffix level similarity the segmentation of “reaction” gets a high score. On the other hand, the segmentation of “billion” into “bill”+“ion”, for example, is wrong (according to suffix level similarity), as “bill” attaches to “ed” and “ing”, which are not among the most similar suffixes of “ion”.

**Table 6.** Suffix level similarity checking for segmentation associated with suffix “ion”

Top rank similar Suffixes of “ion”	Word	WRFR	Suffixes that root attaches to	Correct/Not
ive	reaction = react+ion	4.1	ive or ors ivity ed	<b>Yes</b>
or	billion = bill+ion	1.7	ed, ing	<b>Not</b>
ors	bullion = bull+ion	1.3	ed	<b>Not</b>
ivity	section = sect+ion	5.9	ors	<b>Not</b>
ory	mission = miss+ion	4.4	ed ing ive	<b>Not</b>

One potential problem with suffix level similarity is that it is an overly strict condition for those words that combine with only one or two suffixes in the vocabulary. For example, if the word “character” has just one variant in the vocabulary (e.g., “characters”), suffix level similarity will determine the attachment of “s” to “character” as incorrect, since the weighted sum in the above inequality will be 0. To address this sparseness problem, we rely on both relative corpus frequency and suffix level similarity to identify incorrect attachments. Specifically, if the WRFR of a <word, root> pair is between 1 and 10, we determine that an attachment to the root is incorrect if

$$-WRFR + \gamma * (\text{suffix level similarity}) < 0,$$

where  $\gamma$  is set to 0.15.

Finally, since long words have a higher chance of getting segmented, we do not apply suffix level similarity to words whose length is greater than 10.

## 2.6 Inducing Orthographic Rules and Allomorphs

The biggest drawback of the system, described thus far, is its failure to segment words where the roots exhibit spelling changes during attachment to morphemes (e.g., “denial” = “deny”+“al”). The reasons are (1) the system does not have any knowledge of language-specific orthographic rules (e.g., in English, the character ‘y’ at the morpheme boundary is

changed to ‘i’ when the root combines with the suffix “al”), and (2) the vocabulary we employ for morpheme induction does not normally contain the *allomorphic variations* of the roots (e.g., “deni” is allomorphic variation of “deny”). To segment these words correctly, we need to generate the allomorphs and orthographic rules automatically given a set of induced roots and affixes.

Before giving the details of the generation method, we note that the induction of orthographic rules is a challenging problem, since different languages exhibit orthographic changes in different ways. For some languages (e.g., English) rules are mostly predictable, whereas for others (e.g., Finnish) rules are highly irregular. It is hard to obtain a generalized mapping function that aligns every <root, allomorph> pair, considering the fact that our system is unsupervised. An additional challenge is to ensure that the incorporation of these orthographic rules will not adversely affect system performance (i.e., they will not be applied to regular words and thus segment them incorrectly). Yarowsky and Wicentowski (2000) propose an interesting algorithm that employs four similarity measures to successfully identify the most probable root of a highly irregular word. Unlike them, however, our goal is to (1) check whether the learned rules can actually improve an unsupervised morphological system, not just to align <root, allomorph> pair, and (2) examine whether our system is extendable to different languages.

Taking into consideration the aforementioned challenges, our induction algorithm will (1) handle orthographic character changes that occur only at morpheme boundaries; (2) generate allomorphs for suffixal attachments only<sup>7</sup>, assuming that roots exhibit the character

---

<sup>7</sup> We only learn rules for suffixes of length greater than 1, since most suffixes of length 1 do not participate in orthographic changes.

changes during attachment, not suffixes; and (3) learn rules that aligns <root, allomorph> pairs of edit distance 1 (which may involve 1-character replacement, deletion or insertion). Despite these limitations, we will see that the incorporation of the induced rules improves segmentation accuracy significantly.

Let us first discuss how we learn a *replacement* rule, which identifies <allomorph, root> pairs where the last character of the root is replaced by another character. The steps are as follows:

### (1) Inducing candidate allomorphs

If  $\alpha A \beta$  is a word in the vocabulary (e.g., “denial”, where  $\alpha$ =“den”,  $A$ =“i”, and  $\beta$ =“al”),  $\beta$  is an induced suffix,  $\alpha B$  is an induced root (e.g., “deny”, where  $B$ =“y”), and the attachment of  $\beta$  to  $\alpha B$  is correct according to relative corpus frequency and suffix level similarity, then we hypothesize that  $\alpha A$  is an allomorph of  $\alpha B$ . For each induced suffix, we use this hypothesis to generate the allomorphs and identify those that are generated from at least two suffixes as *candidate allomorphs*. We denote the list of <candidate allomorph, root, suffix> tuples by  $L$ .

### (2) Learning orthographic rules

Every <candidate allomorph, root, suffix> tuple as learned above is associated with an orthographic rule. For example, from the words “denial”, “deny” and the suffix “al”, we learn the rule “y:i / \_ + al”.<sup>8</sup> On the other hand, from “social”, “sock” and “al”, we learn the rule “k:i / \_ + al”, which, however, is erroneous. As we said before, the rules learned should not be “harmful” i.e., they should not segment a regular word incorrectly. So, we check whether

---

<sup>8</sup> This is the Chomsky and Halle notation for representing orthographic rules.  $a:b / c \_ d$  means  $a$  changes to  $b$  when the left context is  $c$  and the right context is  $d$ .

each of the learned rules occurs frequently enough for all the <allomorph, root> pairs associated with a suffix, with the goal of filtering the low-frequency orthographic rules.

Specifically, for each suffix  $\beta$ , we repeat the following steps:

**(a) Counting the frequency of rules.** Let  $L_\beta$  be the list of <candidate allomorph, root> pairs in  $L$  that are associated with the suffix  $\beta$ . For each pair  $p$  in  $L_\beta$ , we first check whether its candidate allomorph appears in any other <candidate allomorph, root> pairs in  $L_\beta$ . If not, we increment the frequency of the orthographic rule associated with  $p$  by 1. For example, the pair <“deni”, “deny”> increases the frequency of the rule “y:i” by 1 on condition that “deni” does not appear in any other pairs.

**(b) Filtering the rules.** We first remove the infrequent rules, specifically those that are induced by less than 15% of the tuples in  $L_\beta$ . Then we check whether there exists two rules of the form  $A:B$  and  $A:C$  in the induced rule list. If so, then we have a morphologically undesirable situation where the character  $A$  changes to  $B$  and  $C$  under the same environment (i.e.,  $\beta$ ). To address this problem, we first calculate the strength of a rule as follows:

$$strength(A : B) = \frac{frequency(A : B)}{\sum_{@} frequency(A : @)}$$

We then retain only those rules whose frequency\*strength is greater than some predefined threshold. We denote the list of rules that satisfy the above constraints by  $R_\beta$ .

**(c) Identifying valid allomorphs.** For each rule in  $R_\beta$ , we identify the associated <candidate allomorph, root> pairs in  $L_\beta$ . We refer to the candidate allomorphs in each of those pairs as *valid allomorphs* and add them to the list of roots. We also remove from the original root list the words that can be segmented by the induced allomorphs and the associated rules (e.g., “denial”).

**(d) Identifying composite suffixes.** For each rule in  $R_\beta$ , we also check whether it can identify composite suffixes where the first component suffix’s last character is replaced during attachment to the second component suffix (e.g., “liness” = “ly”+“ness”). Specifically, if (1)  $A:B / \_ \beta$  is a rule in  $R_\beta$ , (2)  $\alpha A\beta$  (say “liness”),  $\beta$  (say “ness”) and  $\alpha B$  (say “ly”) are induced suffixes, and (3)  $\alpha A\beta$  satisfies the requirements of a composite suffix (see Section 2.3.2), then we determine that  $\alpha A\beta$  is a composite suffix composed of  $\alpha B$  and  $\beta$ .

We employ the same procedure for learning *insertion* and *deletion* rules, except that strength is always set to 1 for these two types of rules.<sup>9</sup> The threshold we set at step (b) is somewhat dependent on the vocabulary size, since the frequency count of each rule will naturally be larger when a larger vocabulary is used. Following our method for setting vocabulary-dependent thresholds (see Section 2.3.1), we set the threshold to 4 for English and 25 for Finnish, for instance. In Table 7, we show a list of induced rules for English.

Finally, we adapt our candidate allomorph detection method described above to induce allomorphs that are generated through orthographic changes of edit distance greater than 1. Specifically, if  $\alpha\beta$  is a word in the induced root list (e.g., “stability”<sup>10</sup>, where  $\alpha$ =“stabil” and  $\beta$ =“ity”),  $\beta$  is an induced suffix, and the attachment of  $\beta$  to  $\alpha$  is correct according to suffix level similarity, then we hypothesize that  $\alpha$  (“stabil”) is a candidate allomorph. For each induced suffix, we use this hypothesis to generate candidate allomorphs and consider as valid allomorphs only those that are generated from at least three different suffixes. This technique can also be used to induce out-of-vocabulary (OOV) roots. For example, the presence of “perplexity”, “perplexed” and “perplexing” in a vocabulary allows

---

<sup>9</sup> Since in insertion and deletion we do not have to encounter morphologically undesirable rules, we simply set the strength of an insertion or deletion rule to 1.

us to induce the root “perplex”. OOV root induction is particularly important for languages like Bengali, where verb roots mostly take the imperative form and hence are absent in a vocabulary created from a newspaper corpus, which normally comprises only the first and third person verb forms.

**Table 7.** Examples of orthographic rules learned for English

Replacement Rules	Example Pair	Deletion Rules	Example Pair	Insertion Rules	Example Pair
y:i / _ +:0 al	<denial deny>	e:0 / _ +:0 al	<urinal urine>	0:t / _ +:0 al	<rebuttall rebut>
y:i / _ +:0 er	<drier dry>	a:0 / _ +:0 ate	<replicate replica>	0:p / _ +:0 ed	<clapped clap>
y:i / _ +:0 ly	<happily happy>	e:0 / _ +:0 ed	<abused abuse>	0:t / _ +:0 ed	<emitted emit>
e:i / _ +:0 st	<activist active>	e:0 / _ +:0 ent	<solvent solve>	0:r / _ +:0 ent	<deterrent deter>
e:i / _ +:0 ty	<security secure>	y:0 / _ +:0 ic	<botanic botany>	0:t / _ +:0 er	<chatter chat>
e:a / _ +:0 nt	<mutant mute>	y:0 / _ +:0 ize	<fertilize fertile>	0:g / _ +:0 le	<smuggle smug>

## 2.7 Word Segmentation

After inducing the morphemes, we can use them to segment a word  $w$  in the test set. Specifically, we (1) *generate* all possible segmentations of  $w$  using only the induced affixes and roots, and (2) apply a sequence of tests to *remove* candidate segmentations until we are left with only one candidate, which we take to be the final segmentation of  $w$ . Our first test involves removing any candidate segmentation  $m_1 m_2 \dots m_n$  that violates any of the linguistic constraints below:

- At least one of  $m_1, m_2, \dots, m_n$  is a root.
- For  $1 \leq i < n$ , if  $m_i$  is a prefix, then  $m_{i+1}$  must be a root or a prefix.
- For  $1 < i \leq n$ , if  $m_i$  is a suffix, then  $m_{i-1}$  must be a root or a suffix.

---

<sup>10</sup> The correct segmentation of “stability” is “stable”+“ity”. The “stabil”-“stable” allomorph-root pair is an example of an orthographic change of edit distance 2.

- $m_1$  can't be a suffix and  $m_n$  can't be a prefix.

Next, we apply our second test, in which we retain only those candidate segmentations that have the smallest number of morphemes. For example, if “friendly” has two candidate segmentations “friend”+“ly” and “fri”+“end”+“ly”, we will select the first one to be the segmentation of  $w$ .

If more than one candidate segmentation still remains, we apply our third test to remove any candidate  $c$  that *satisfies* one of the three cases below.

**Case 1:** There exists a root  $r$  in  $c$  such that  $r$  is immediately preceded by a prefix  $p$  and immediately followed by a suffix  $s$ , but neither the substring  $pr$  nor the substring  $rs$  is in our vocabulary.

**Case 2:** There exists a root  $r$  in  $c$  such that  $r$  is immediately preceded by a prefix  $p$  but *not* immediately followed by a suffix, and the substring  $pr$  is *not* in our vocabulary.

**Case 3:** There exists a root  $r$  in  $c$  such that  $r$  is immediately followed by a suffix  $s$  but not immediately preceded by a prefix, and the substring  $rs$  is *not* in our vocabulary.

As an example of applying the third test described above, consider segmenting the Bengali word “আরবিতে” (@rbITE). This word has two candidate segmentations (“@rb”+“I”+“TE” and “@rb”+“IT”+“E”), both of which follow the Root+Suffix+Suffix pattern. Since “@rbI” is in our vocabulary whereas “@rbIT” is not, we remove “@rb”+“IT”+“E” from our list of candidate segmentations (because the second case is satisfied) but retain “@rb”+“I”+“TE” (because none of the three cases is satisfied).

If more than one candidate segmentation still exists, we score each remaining candidate using the heuristic below, selecting the highest-scoring candidate to be the final segmentation of  $w$ . Basically, we score each candidate segmentation by adding the *strength*

of each morpheme in the segmentation, where (1) the strength of an affix is the number of distinct words in the vocabulary to which the affix attaches, multiplied by the length of the affix, and (2) the strength of a root is the number of distinct morphemes with which the root combines, again multiplied by the length of the root.

## 2.8 Evaluation

In this section, we will first evaluate our segmentation algorithm for English and Bengali, and then examine its performance on the PASCAL datasets.

### 2.8.1 Experimental Setup

**Vocabulary creation.** We extracted our English vocabulary from the Wall Street Journal corpus of the Penn Treebank and the BLLIP corpus, preprocessing the documents by first tokenizing them and then removing capitalized words, punctuations and numbers. In addition, we removed words of frequency 1 from BLLIP, because many of them are proper nouns and misspelled words. The final English vocabulary consists of approximately 60K word types. We applied the same pre-processing steps to five years of articles taken from the Bengali newspaper *Prothom Alo* to generate our Bengali vocabulary. Specifically, we only use articles that are sports news or editorials, as well as those that appear in the first page and the last page of the newspaper.<sup>11</sup> The final Bengali vocabulary consists of 142K word types. Unlike morphological analysis for many European languages, however, we do not take the conventional step of removing proper nouns from our vocabulary, because we do not have a name entity identifier for Bengali.

---

<sup>11</sup> These are the major sections of *Prothom Alo*. The remaining sections are relatively small and are simply ignored.

**Test set preparation.** To create our English test set, we randomly chose 5000 words from our vocabulary that are at least 4-character long<sup>12</sup> and also appear in CELEX. Although 95% of the time we adopted the segmentation proposed by CELEX, in some cases the CELEX segmentations are erroneous (e.g., “rolling” and “lodging” remain unsegmented in CELEX). As a result, we cross-check with the online version of Merriam-Webster to make the necessary changes.

To create our Bengali test set, we randomly choose 5000 words from our vocabulary that are at least 3-character long. We then manually remove the proper nouns and words with spelling mistakes from the test set before giving it to two of our linguists for hand-segmentation. In the absence of a complete knowledge-based morphological parsing tool and a hand-tagged morphological database for Bengali, our linguists had to depend on the Bengali dictionary<sup>13</sup> for annotating our test cases. There is one caveat in our manual annotation procedure, however. Many Bengali words are morphologically derived from Sanskrit roots. These words are very difficult, if not impossible, for any morphological analyzer to segment correctly, because the orthographic changes that take place during the segmentation process are highly non-linear and complex in nature. One example of such word is “বিরুদ্ধ” (bIrUd~D), whose actual segmentation is “বি”+“রুদ্ধ”+“ত (ভ)” (bI+rUd+k~T (T)) – which is tough to obtain. As a result, we instruct our linguists to simplify the segmentation of these words so that the orthographic changes are within tractable edit distance. Given this restriction, the Bengali word shown above (i.e. “বিরুদ্ধ”) will simply be

---

<sup>12</sup> Words of length less than 4 do not have any morphological segmentation in English. Hence, by imposing this length restriction on the words in our test set, we effectively make the evaluation more challenging. This is also the reason for our using words that are at least 3-character long in the Bengali test set.

segmented as “বি”+“বুদ্ধ” (bI+rUd~D). However, if the meaning of a segmented word differs from that of the original word, then we simply treat the original word as a root (i.e. the word should not be segmented at all). Words that fall within this category include “প্রধান”, “আবেদন”, and “প্রতিবেদন”, for instance. After all the words have been manually segmented, we remove those for which the two linguists produce inconsistent segmentations. The resulting test set contains 4191 words.

**Evaluation metrics.** We use two standard metrics --- *exact accuracy* and *F-score* --- to evaluate the performance of our morphological parser on the test set. Exact accuracy is the percentage of the words whose proposed segmentation ( $S_P$ ) is identical to the correct segmentation ( $S_C$ ). F-score is simply the harmonic mean of recall and precision, as computed using the formulas below.

$$\text{Precision} = (H) / (H+I)$$

$$\text{Recall} = (H) / (H+D)$$

$$\text{F-score} = (2H) / (2H+I+D)$$

where H is the number of Hits (*i.e.*, correctly placed boundaries), and I, D represent the number of morpheme boundaries needed to be inserted into and deleted from  $S_C$ , respectively, to make it identical to  $S_P$ . For instance, comparing the incorrect segmentation “un”+“fri”+“endly” against the correct segmentation “un”+“friend”+“ly”, we obtain 1 Hit, 1 Insertion and 1 Deletion, thus yielding a F-score of 0.5 and an exact accuracy of 0. Note that most previous work simply reports results in terms of F-score, which is a less stringent evaluation metric than exact accuracy. However, we believe that reporting results in terms of

---

<sup>13</sup> The dictionaries we used are “বঙ্গীয় শব্দকোষ” (Bangiya Sabdakosh) by হরিচরণ বন্দ্যোপাধ্যায় (Haricharan Bandopadaya) and “বাংলা একাডেমী ব্যবহারিক বাংলা অভিধান” (Bangla Academy Bebhariic Bangla Avidan).

both metrics will give us a better picture of the strengths and weaknesses of a morphological parser.

### 2.8.2 Results for English and Bengali

**The baseline systems.** We use two publicly available and widely used unsupervised morphological learning systems -- Goldsmith's (2001) *Linguistica*<sup>14</sup> and Creutz and Lagus's (2005) *Morfessor 1.0*<sup>15</sup> -- as our baseline systems. The first two rows of Table 8 show the results of these systems for our test sets (with all the training parameters set to their default values). As we can see, *Linguistica* performs substantially better for English in terms of both exact accuracy and F-score, whereas *Morfessor* outperforms *Linguistica* for Bengali. Note that, *Morfessor* tends to split the words into small segments<sup>16</sup>, whereas *Linguistica* is more reluctant to segmenting words due to its use of signatures. As we said before, English is a morphologically impoverished language that hardly exhibits more than one segmentation per word. As a result, *Linguistica* performs very well for English, whereas *Morfessor* performs poorly as it mostly generates over-segmentations, thus failing to get a good precision (69% compared to *Linguistica*'s 84%). In fact, Creutz (2003) made a sterling observation that, for a large vocabulary system, the no-segmentation output outperforms both *Linguistica* and *Morfessor* for English. On the other hand, Bengali morphology is much more productive than that of English. So, whereas *Morfessor* seems to extract all the morphemes of a word in Bengali, *Linguistica* performs poorly, mostly generating under-segmentations and fails to get a good recall (58% compared to 89% as achieved by *Morfessor*).

---

<sup>14</sup> <http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/>

<sup>15</sup> <http://www.cis.hut.fi/projects/morpho/>

<sup>16</sup> Creutz (2003) later proposed to use prior length and frequency information to stem over-segmentation.

**Our segmentation algorithm.** Results of our segmentation algorithm are shown in rows 3-6 of Table 8. Specifically, row 3 shows the results of our basic segmentation system as described in Section 3. Rows 4-6 show the results where our three techniques (i.e., relative frequency, suffix level similarity and allomorph detection) are incorporated into the basic system one after the other. It is worth mentioning that (1) our basic algorithm already outperforms the baseline systems in terms of both exact accuracy and F-score; and (2) while each of our additions to the basic algorithm boosts system performance, relative corpus frequency and allomorph detection contribute to performance improvements particularly significantly. As we can see, the best segmentation performance is achieved when all of our three additions are applied to the basic algorithm.

**Table 8.** Results (reported in terms of exact accuracy (A), precision (P), recall (R) and F-score (F))

	English				Bengali			
	A	P	R	F	A	P	R	F
Linguistica	68.9	84.8	75.7	80.0	36.3	58.2	63.3	60.6
Morfessor	64.9	69.6	85.3	76.6	56.5	89.7	67.4	76.9
Basic induction	68.1	79.4	82.8	81.1	57.7	79.6	81.2	80.4
Relative frequency	74.0	86.4	82.5	84.4	63.2	85.6	79.9	82.7
Suffix level similarity	74.9	88.6	82.3	85.3	66.1	89.7	78.8	83.9
Allomorph	<b>78.3</b>	88.3	86.4	<b>87.4</b>	<b>68.3</b>	89.3	81.3	<b>85.1</b>

### 2.8.3 Discussion and Error Analysis

As part of the analysis of our algorithm, we examine whether our morphological analyzer can handle complicated test cases. For English, we successfully segment words with complex morphotactics like “indefinitely” into “in”+“define”+“ite”+“ly”. For Bengali, our system

successfully segments verbal inflections like “দুলিয়েছিল” (dUIIyECII) as “dUI”+“IyE”+“CI”+“I”, and multi-root words like “বিনোদনকেন্দ্রগুলোও” (bInOdnkEndRgUIOo) as “bInOd”+“n”+“kEndR”+“gUIO”+“o”. Even more interestingly, it correctly parses English words, which are widely used in the Sports section of the Bengali newspaper. For example, words like “বলিং” (bIIng) and “ফাইনালিস্ট” (FAInAIIS~t) are correctly segmented as “bl+Ing” and “FAInAI+IS~t”, respectively. It is worth mentioning that the compounding nature of Bengali and the influence of foreign languages have introduced into our repository a lot of new words, whose presence increases the difficulty of the segmentation task. Nevertheless, our morphological parser manages to stem those words correctly.

We also examined the words that were incorrectly segmented by our system. The errors can be broadly divided into following categories:

**(1) Over-segmentation.** This constitutes the biggest percentage of errors in both languages. For example, for English, we incorrectly segment “secretary” into “secret”+“ary” and “massage” into “mass”+“age”. For Bengali, “শিকল” (sIkI) is a root word but it is incorrectly parsed as “sIk”+“I”. Although we use relative frequency and suffix level similarity to detect incorrect morpheme attachments, many over-segmentations still remain undetected. The reason can be attributed to the fact that relative frequency and suffix level similarity can detect an incorrect attachment if and only if the WRFR is greater than 1. However, there are still some incorrect attachments which have a WRFR less than 1, which remain undetected. This suggests that we need an even more sophisticated algorithm for incorrect morpheme attachment detection.

**(2) Under-segmentation.** In some cases, our system fails to segment the words (e.g., “excellence”=“excel”+“ence”, “vigorous”=“vigor”+“ous”, “usage”=“use”+“age” in English),

which can be attributed to many reasons: first, relative frequency and suffix level similarity sometimes identify an correct attachment as an incorrect one; second, our system fails to learn all the spelling change rules thus failing to segment the words that show spelling change rules; and third, in some cases, the corpus does not contain the necessary root of the word.

**(3) Irregular words.** Though we automatically induced orthographic rules to segment the words which exhibit spelling changes during attachment, it was not enough to segment irregular words (e.g., “felt”, “drew”, “stability” etc.) that show complex spelling changes of edit distance greater than 1.

For Bengali, *verbal inflections* constitute a large portion of the words incorrectly segmented by our algorithm. There are two reasons for such errors. First, the root of an incorrectly segmented verb is missing from the corpus. For instance, “উঠা” (uthA) is incorrectly segmented because its root “উঠ” (uth) is not found in the corpus. Second, the first and second person forms of verbs are often missing in the corpus, as the newspaper articles from which our vocabulary is induced contain mostly third person forms of verbs.

#### **2.8.4 PASCAL Challenge Results**

To get an idea of how our algorithm performs in comparison to the PASCAL participants, we conducted evaluations on the PASCAL datasets for English, Finnish and Turkish. Table 9 shows the F-scores of four segmentation algorithms for these three datasets: the best-performing PASCAL system (Winner), Morfessor, our system that uses the basic morpheme induction algorithm (Basic), and our system with all three extensions incorporated (Complete). Below we discuss these results.

**English.** There are 533 test cases in this dataset. Using the vocabulary created as described in Section 8.1, our Complete algorithm achieves an F-score of 79.4%, which outperforms the winner (Keshava and Pitler, 2006) by 2.6%. Although our basic morpheme induction algorithm is similar to that of Keshava and Pitler, a closer examination of the results reveals that F-score increases significantly with the incorporation of relative frequency and allomorph detection.

**Finnish and Turkish.** The real challenge in the PASCAL Challenge is the evaluation on Finnish and Turkish due to their morphological richness. We use the 400K and 300K most frequent words from the Finnish and Turkish datasets provided by the organizers as our vocabulary. When tested on the gold standard of 661 Finnish and 775 Turkish words, our Complete system achieves F-scores of 65.2% and 66.2%, which are better than the winner’s scores (Bernhard (2006)). In addition, Complete outperforms Basic by 3-6% in F-score; these results suggest that the new techniques proposed in this thesis (especially allomorph detection) are also very effective for Finnish and Turkish.

**Table 9.** F-scores for the PASCAL gold standards

	<b>English</b>	<b>Finnish</b>	<b>Turkish</b>
<b>Winner</b>	76.8	64.7	65.3
<b>Morfessor</b>	66.2	66.4	70.1
<b>Basic</b>	75.8	59.2	63.4
<b>Complete</b>	79.4	65.2	66.2

As mentioned in the introduction, none of the participating PASCAL systems offers robust performance across different languages. For instance, Keshava and Pitler’s algorithm, the winner for English, has F-scores of only 47% and 54% for Finnish and Turkish respectively, whereas Bernhard’s algorithm, the winner for Finnish and Turkish, achieves an

F-score of only 66% for English. On the other hand, our algorithm outperforms the winners for *all* the languages in the competition, demonstrating its robustness across languages.

Finally, although Morfessor achieves better results for Turkish and Finnish than our Complete system, it performs poorly for English, having an F-score of only 66.2%. On the other hand, our results for Finnish and Turkish are not significantly poorer than those of Morfessor.

## CHAPTER 3

### UNSUPERVISED PART-OF-SPEECH ACQUISITION

#### 3.1 Problem Definition and Goals

As mentioned before, the goal of an unsupervised Part-Of-Speech (POS) acquisition system is to automatically build a POS lexicon (i.e., to learn the set of possible POS tags for each lexical item) for a particular language. In a broader sense, given an unannotated text corpus, we have to (soft) cluster each word into its corresponding POS class (es). A part-of-speech acquisition system, therefore, is different from a part-of-speech tagging system, whose goal is mainly to disambiguate, i.e., to select the most probable tag of a word in a particular context out of the list of allowable tags as supplied by the POS lexicon. For example, given the word “received” in English, a POS lexicon lists all possible POS tags of “received” (i.e., Past Tense of Verb and Adjective), whereas a POS tagging system determines that the POS tag of “received” in the sentence “The received message was distorted” is Adjective.

Although there has been considerable work on unsupervised POS tagging system (e.g., Kupiec (1992), Merialdo (1992), Banko and Moore (2004), Smith and Eisner (2004), Wang and Schuurmans (2005)), the same is not true for unsupervised POS lexicon acquisition system. Typically, an unsupervised POS tagging system learns the parameters of a generative model (e.g., HMM) using Baum-Welch re-estimation technique, assuming that there is a built in high coverage POS lexicon that gives all possible POS tags of each word. However, this assumption is too strong for resource-scarce languages, as we can hardly expect a complete POS lexicon for a resource-scarce language. Our goal in this thesis is to

address this problem by building a high-performance, high-coverage POS lexicon for a resource-scarce language, which can then be exploited to build a completely unsupervised POS tagging system.

The most common approach to unsupervised POS acquisition to date has been motivated by Harris's (1954) distributional hypothesis: words with similar co-occurrence patterns should have similar syntactic behavior. More specifically, unsupervised POS induction algorithms typically operate by (1) representing each target word (i.e., a word to be tagged with its POS) as a context vector that encodes its left and right context, (2) clustering distributionally similar words, and (3) manually labeling each cluster with a POS tag by inspecting the members of the cluster. This distributional approach works under the assumption that the context vector of each word encodes sufficient information for enabling accurate word clustering. However, many words are distributionally unreliable: due to data sparseness, they occur infrequently and hence their context vectors do not capture reliable statistical information. To overcome this problem, Clark (2000) proposes a bootstrapping approach, in which he (1) clusters the most distributionally reliable words, and then (2) incrementally augments each cluster with words that are distributionally similar to those already in the cluster.

The goal of this thesis is to propose a new bootstrapping algorithm that uses morphological information on top of distributional similarity to cluster the words according to the POS tags. Most notably, our approach aims to improve the quality of the seed clusters by employing seed words that are both distributionally and morphologically reliable. Morphology gives a strong clue to the part-of-speech of a word. For example, words ending with suffix "est" in English are always tagged as superlative adjectives, whereas words that

attach to suffixes like “ing”, “ed”, “s” are usually tagged as verbs. Although morphological information has been used in existing supervised or unsupervised POS tagging/induction systems, it was mainly employed to guess the POS tag of unknown or low frequency words (e.g., Mikheev (1997), Cucerzan and Yarowsky (2000), Clark (2003)). On the other hand, our approach more tightly integrates morphology into clustering framework, assuming that morphological features, when used on top of distributional features can improve the overall clustering accuracy.

It is perhaps not immediately clear why morphological information would play a crucial role in the induction process, especially since the distributional approach has achieved considerable success for English POS induction (see Lamb (1961), Schütze (1995) and Clark (2000)). To understand the role and significance of morphology, it is important to first understand why the distributional approach works well for English. Recall from the above that the distributional approach assumes that the information encoded in the context vector of each word, which typically consists of the 250 most frequent words of a given language, is sufficient for accurately clustering the words. This approach works well for English because the most frequent English words are composed primarily of closed-class words such as “to”, “is”, “the”, which provide strong clues to the POS of the target word. However, this assumption is not necessarily valid for fairly free word order and highly inflectional languages such as Bengali. The reason is that (1) co-occurrence statistics collected from free word order languages are not as reliable as those from fixed word order languages; and (2) many of the closed-class words that appear in the context vector for English words are realized as inflectional suffixes in Bengali. The absence of these highly informative words implies that the context vectors may no longer capture sufficient information for accurately

clustering Bengali words, and hence the use of morphological information becomes particularly important for unsupervised POS induction for these inflectional languages.

Now the important question is how to get the morphological information of a particular language. We cannot expect the availability of a complete knowledge-based morphological system for resource-scarce languages. In addition, our goal is to induce the POS lexicon from just an unannotated text corpus without using any additional language specific knowledge sources. Our proposed solution to this problem is to use an unsupervised morphological analyzer that can be built just from an unannotated corpus. In particular, we will use the unsupervised morphological analyzer described in Chapter 2. Although the unsupervised morphological system is not perfect (F-score in the range of 85-87%), experimental results show that we can still get state-of-the-art performance, suggesting that our clustering algorithm is noise tolerant to incorrect segmentations.

Due to the fact that closed-class words generally comprise a small percentage of the lexical items of a language, our focus in this thesis is to automatically label only open-class words with their POS tags. In fact, for English, the most frequent 500 words in the corpus form the 90% of the closed class words. The percentage of closed-class words in Bengali is smaller than that in English: as mentioned before, many closed-class words in English are realized as suffixes in Bengali.

Although our attempt to incorporate morphological information into the distributional POS induction framework was originally motivated by inflectional languages, experimental results show that our approach works well for both English and Bengali, suggesting its applicability to both morphologically impoverished languages and highly inflectional languages. Owing to the lack of publicly available resources for Bengali, we manually

created a 5000-word Bengali lexicon for evaluation purposes. Hence, one contribution of our work lies in the creation of an annotated dataset for Bengali. By making this dataset publicly available<sup>17</sup>, we hope to facilitate the comparison of different unsupervised POS induction algorithms and to stimulate interest in Bengali language processing.

The rest of this chapter is organized as follows. Section 3.2 discusses related work on unsupervised POS induction. Section 3.3 describes our tagsets for English and Bengali. The next four sections describe the four steps of our bootstrapping approach: cluster the words using morphological information (Section 3.4), remove potentially mislabeled words from each cluster (Section 3.5), bootstrap each cluster using a weakly supervised learner (Section 3.6) and classify the morphologically and distributionally poor words (Section 3.7). Finally, we present evaluation results in Section 3.8.

## 3.2 Related Work

Several unsupervised POS acquisition and tagging algorithms have also attempted to incorporate morphological information into the distributional framework, but our work differs from these in two respects.

**Computing morphological information.** Previous POS induction algorithms have attempted to derive morphological information from dictionaries (Hajič, 2000) and knowledge-based morphological analyzers (Duh and Kirchhoff, 2006). However, these resources are generally not available for resource-scarce languages. Consequently, researchers have attempted to derive morphological information heuristically (e.g., Cucerzan and Yarowsky (2000), Clark (2003), Freitag (2004)). For instance, Cucerzan and Yarowsky

---

<sup>17</sup> See <http://www.hlt.utdallas.edu/~sajib/posDatasets.html>.

(2000) posit a character sequence  $x$  as a suffix if there exists a sufficient number of distinct words  $w$  in the vocabulary such that the concatenations  $wx$  are also in the vocabulary. It is conceivable that such heuristically computed morphological information can be inaccurate, thus rendering the usefulness of a more accurate morphological analyzer. To address this problem, we exploit morphological information provided by an unsupervised word segmentation algorithm.

**Using morphological information.** Perhaps due to the overly simplistic methods employed to compute morphological information, morphology has only been used as what Biemann (2006) called *add-on's* in existing POS induction algorithms, which remain primarily distributional in nature. In contrast, our approach more tightly integrates morphology into the distributional framework. As we will see, we train SVM classifiers using both morphological and distributional features to select seed words for our bootstrapping algorithm, effectively letting SVM combine these two sources of information and perform automatic feature weighting. Another appealing feature of our approach is that when labeling each unlabeled word with its POS tag, an SVM classifier also returns a numeric value that indicates how confident the word is labeled. This opens up the possibility of having a human improve our automatically constructed lexicon by manually checking those entries that are tagged with low confidence by an SVM classifier.

Recently, there have been attempts to perform (mostly) unsupervised POS tagging without relying much on a POS lexicon. Smith and Eisner (2005) train a log-linear model for POS tagging in an unsupervised manner using a diluted dictionary, where infrequent words may have any tag. Haghghi and Klein's (2006) *prototype-driven* approach requires just a few (3 in fact) prototype examples for each POS tag, exploiting these labeled words to

constrain the labels of their distributionally similar words when training a generative log-linear model for POS tagging.

**Table 10. English Open Class POS Tagset**

Tag	Description	Treebank tags
JJ	Adjective	JJ
JJR	Adjective, comparative	JJR
JJS	Adjective, superlative	JJS
NN	Singular noun	NN, NNP
NNS	Plural noun	NNS, NNPS
RB	Adverb	RB
VB	Verb, non-3 <sup>rd</sup> ps. sing. present	VB, VBP
VBD	Verb, past tense or past participle	VBD, VBN
VBG	Verb, gerund/present participle	VBG
VBZ	Verb, 3 <sup>rd</sup> ps. sing. Present	VBZ

**Table 11. Bengali Open Class POS Tagset**

Tag	Description	Examples
JJ	Adjective	vhalo, garam, kharap
NN	Singular common noun	kanna, ridoy, shoshon
NN2	2 <sup>nd</sup> order inflectional noun	dhopake, kalamtike
NN6	6 <sup>th</sup> order inflectional noun	gharer, manusher
NN7	7 <sup>th</sup> order inflectional noun	dhakai, barite, graame
NNP	Singular proper noun	arjun, ahmmad
NNS	Plural noun	manushgulo, pakhider
NNSH	Noun ending with “sh”	barish, jatrish
VB	Finite verb	Kheyechi, krlam, krI
VBN	Non-finite verb	kre, giye, jete, kadte

### 3.3 The English and Bengali Tag Sets

Given our focus on automatically labeling open class words, our English and Bengali tagsets are designed to essentially cover all of the open-class words. Our English tagset, which is composed of ten tags, is shown in Table 10. As we can see, a tag in our tagset can be mapped to more than one Penn Treebank tags. For instance, we use the tag “NN” for both proper and common nouns. Our decision of which Penn Treebank tags to group together is based on that of Schütze (1995).

Our Bengali tagset, which also consists of ten tags, is adapted from the one proposed by Saha et al. (2004) (see Table 11). Note that NN2, NN6 and NN7 represent three of the seven linguistically defined Bengali noun inflections or *bivacti* (বিভক্তি). For example, nouns ending with a suffix like kE (কে), rE (রে) are labeled as NN2, nouns ending with a suffix like r (র), Er (এর) are labeled as NN6 and nouns ending with a suffix like E (এ), TE (তে), y (য়) are labeled as NN7. Note that Bengali verb phrases have a complex architecture: two word sequences sometimes occur together to form a *Compound Verb* (CV) (e.g., mErE phEllo (মেরে ফেললো), chOyA gEllo (ছোয়া গেলো), @k~rmN krl (আক্রমণ করল)). The second component of a CV is always a finite verb (i.e., VBF), but first component can take different forms according to the suffixes they are attached to. We pos-tag the first CV component like this: words ending with suffixes like “E”, “y”, “TE” (e.g., mErE (মেরে)) are tagged as verb non finite (VBN), whereas words ending with suffix “A” (e.g., chOyA (ছোয়া)) or no suffix (@k~rmN (আক্রমণ)) are tagged as NN (traditional linguists, though, tag those as verbal noun).

It is worth noting that unlike English, we assign different tags to Bengali proper nouns and common nouns. The reason is that for English, it is not particularly crucial to distinguish the two types of nouns during POS induction, since they can be distinguished fairly easily using heuristics such as initial capitalization. For Bengali, such simple heuristics do not exist, as the Bengali alphabet does not have any upper and lower case letters. Hence, it is important to distinguish Bengali proper nouns and common nouns during POS induction.

### 3.4 Clustering the Morphologically Similar Words

As mentioned before, our approach aims to more tightly integrate morphological information into the distributional POS induction framework. In fact, our POS induction algorithm begins

by clustering the *morphologically similar* words (i.e., words that combine with the same set of suffixes). The motivation for clustering morphologically similar words can be attributed to our hypothesis that words having similar POS should combine with a similar set of suffixes. For instance, verbs in English combine with suffixes like “ing”, “ed” and “s”, whereas adjectives combine with suffixes like “er” and “est”. Note, however, that the suffix “s” can attach to both verbs and nouns in English, and so it is not likely to be a useful feature for identifying the POS of a word. The question, then, is how to determine which suffixes are useful for the POS identification task in an *unsupervised* setting where we do not have any prior knowledge of language-specific grammatical constraints. This section proposes a method for identifying the “useful” suffixes and employing them to cluster the morphologically similar words. As we will see, our clustering algorithm not only produces soft clusters, but it also automatically determines the number of clusters for a particular language.

Before we describe how to identify the useful suffixes, we need to (1) induce all of the suffixes and (2) morphologically segment the words in our vocabulary.<sup>18</sup> However, neither of these tasks is simple for a truly resource-scarce language for which we do not have a dictionary or a knowledge-based morphological analyzer. As mentioned before, our proposed solution to both tasks is to use an unsupervised morphological analyzer that can be built just from an unannotated corpus.

Given the segmentation of each word and the most frequent 30 suffixes provided by our morphological analyzer, our clustering algorithm operates by (1) clustering the similar

---

<sup>18</sup> A vocabulary is simply a set of (distinct) words extracted from an unannotated corpus. We extracted our English and Bengali vocabulary from WSJ and Prothom Alo, respectively.

suffixes and then (2) assigning words to each cluster based on the suffixes a word combines with. One might wonder why we cluster the suffixes according to the words each suffix attaches to (i.e., using the words as features), instead of employing the more traditional approach of clustering the words according to the suffixes each word combines with (i.e., using the suffixes as features). There are two reasons for our decision. First, there are 35K word types in English. Hence, clustering 30 suffixes is arguably an easier task as we have to deal with a much smaller sample space. Second, our unsupervised morphological analyzer is not perfect (recall that F-score is in the range of 85-87%) and so clustering has to be robust to incorrect segmentations produced by the unsupervised analyzer. As we will see, suffix clustering is more robust to imperfect segmentation than the word clustering. Intuitively, out of all the words that combine with a particular suffix, if at least some percentage is correct, we will still be able to achieve a good clustering accuracy.

Before we describe the clustering process, we need to define the similarity between two suffixes. Informally, we say that two suffixes  $x$  and  $y$  are similar if a word that combines with  $x$  also combines with  $y$  and vice versa.

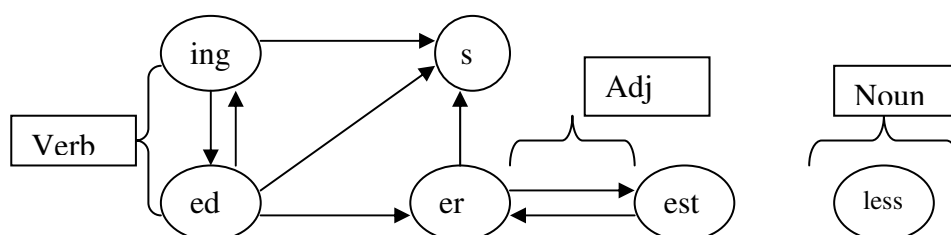
$$\text{sim}(x, y) \Leftrightarrow (\forall w : w + x \Rightarrow w + y) \ \& \ (\forall w : w + y \Rightarrow w + x)$$

In practice, we will rarely posit two suffixes as similar under this definition, for the following two reasons. First, not all suffixal combinations of a word are found in a naturally occurring corpus, unless we assume access to a complete vocabulary – an assumption that is especially unrealistic for resource-scarce languages. Second, the low recall of the unsupervised morphological analyzer suggests that many words might have remained unsegmented. As a result, we relax this definition and consider two suffixes  $x$  and  $y$  similar if  $P(x | y) > t$  and  $P(y | x) > t$ , where  $P(x | y)$  is the probability of a word combining with suffix  $x$

given that it combines with suffix  $y$ , and  $t$  is a threshold that we set to 0.4 in all of our experiments (we will discuss the effect of this threshold later in this section). Note that the probability  $P(x | y)$  and  $P(y | x)$  can be estimated from an unannotated corpus.<sup>19</sup> Given this definition of similarity, we can cluster the similar suffixes using the following steps:

### 3.4.1 Creating the Initial Clusters

First, we create a *suffix graph*, in which we have (1) one node for each of the 30 suffixes, and (2) a directed edge from suffix  $x$  to suffix  $y$  if  $P(y | x) > t$ . We then identify the strongly connected components of this graph using depth-first search. These strongly connected components define our initial partitioning of the 30 suffixes. We denote the suffixes assigned to a cluster the *primary keys* of the cluster. For example, from the suffix graph of Figure 1, we get the following four clusters: {ing, ed}, {er, est}, {s} and {less}.



**Figure 1.** Suffix Graph comprising suffixes “s”, “er”, “est”, “ed”, “ing” and “less”

### 3.4.2 Improving the Initial Clusters

Recall that we ultimately want to cluster the words by assigning each word  $w$  to the cluster in which  $w$  combines with all of its primary keys. Given this goal, it is conceivable that ambiguous clusters are not desirable. For instance, a cluster that has “s” as its only primary

<sup>19</sup> For instance, we compute  $P(x | y)$  as the ratio of the number of distinct words that combines with both  $x$  and  $y$  to the number of distinct words that combine with  $y$  only.

key is ambiguous, as “s” attaches to both nouns and verbs in English. In addition, many initial clusters are singleton (i.e., only one suffix) – which makes them prone to incorrect assignment of words. Consider the singleton cluster “ation” in English. Although “amalgam” (POS noun) can attach to “ation”, it should not be in the same cluster as “consult”, “explore”, “imagine” etc. (POS verb), which also attach to “ation”. As a result, we improve each initial cluster by adding more suffixes to the cluster, in hopes of improving the resulting clustering of the words by placing additional constraints on each cluster. More specifically, for each pair of clusters  $c$  and  $c'$ , we check whether the following condition is satisfied:

$$\forall x \in \text{primarykey}(c), \forall y \in \text{primarykey}(c') : P(y | x) > t$$

If this condition is satisfied, then all primary keys of cluster  $c'$  becomes the *secondary keys* of cluster  $c$ . For each initial cluster  $c$ , we perform this check to see whether any other cluster  $c'$  can be added to  $c$ . If, after this expansion step, we still have a cluster  $c^*$  defined only by primary key (i.e., no secondary key), but serving as a secondary key in other clusters, then cluster  $c^*$  is *probably ambiguous* (i.e., the suffixes in  $c^*$  can probably attach to words belonging to different POSs); and consequently, we remove  $c^*$ . We denote the resulting set of clusters by  $C$ . Table 12 and 13 show the initial clusters derived from the English and Bengali corpora by setting the threshold  $t$  to 0.4. As we can see, clusters #17 and #20 in English (with primary key “s” and “ly”<sup>20</sup> respectively) are ambiguous and are removed subsequently. For Bengali, our algorithm correctly identifies suffixes like i (ই), o (ও), kE (কে), Er (এর) as ambiguous. It is worth noting that not all singleton clusters are ambiguous: some of the Bengali clusters (like “TA”, cluster #9 in Table 13) do not get removed even though they are singleton. As defined before, we use a simple heuristic to determine an ambiguous

cluster. If a suffix cluster (say “s”) gets distributed (as a secondary key) to at least one other cluster, then it is probably ambiguous, as it is more likely to attach with different POSs. Cluster “TA” in Bengali is a completely disconnected component in the suffix graph, hence it does not get distributed to other clusters.

**Table 12.** Initial clusters derived from English corpus,  $t$  set to 0.4

Cluster #	Primary key	Secondary key	Sample words	POS class
0	ability	able ed er ing s	accept, depend	VB
1	able	ed er ing s	accept, adapt, agree	VB
2	al	s	accident, addition	NN
3	ally	al s	accident, historic	NN
4	ation	ed ing s	adapt, attest	VB
5	ed ing	s	abort, convey, sing	VB
6	er	ed ing s	adjust, defend	VB
7	est	er ly ness	bleak, loud, slow	JJ
8	ful	ed ing s	hope, waste	NN
9	ic	s	acid, symbol	NN
10	ion	ed ing s	addict, disconnect	VB
11	ism	ist s	brutal, plural	JJ
12	ist	ism s	national, plural	JJ
13	ity	ly	absurd, plural	JJ
14	ive	ed ing s ion	construct, product	VB
15	ize ized	s	author, visual	JJ
16	less	ed ing s	head, count	NN
17	ly	{deleted}		
18	ment	ed ing s	abandon, govern	VB
19	ness	ly	aware, conscious	JJ
20	s	{deleted}		
21	y	s ed ing	length, risk, word	NN

---

<sup>20</sup> “ly” in English attaches to both nouns (e.g., friend) and adjective (e.g., amazing) and hence form a ambiguous cluster.

**Table 13.** Initial clusters derived from Bengali Corpus,  $t$  set to 0.4

Cluster #	Primary key	Secondary key	Sample words	POS class
0	*	E Er	bIdEs, mAnb, bIs~bAS	NN
1	A	{deleted}		
2	Ar	A E Er	mAp, p~rOg~rAm, k~jAm~p	NN
3	CE CII	i o	nAmIyE, KEIE	VBN
4	E Er	{deleted}		
5	I	E Er	hISAb, upS~fIT	NN
6	Il	E Er	p~l~jAn, tAim	NN
7	S		SIS~tEm, c~jAm~pIyn, sUnECI	NN
8	Sh	kE	nAm, s~bsUr, tUtUI	NN
9	TA		@DunIk, bAS~Tb	JJ
10	TE	r	K'AcA, aTibRF~tI	NN
11	UI	kE	ersAd, hAb*b	NNP
12	Ur	A UI kE	mahAbUb, nASIm	NNP
13	dEr rA	kE	dOF*, nb*n	NN
14	gUIO tI	E Er i kE o	hISAb, mAmlA	NN
15	h*n	E Er i kE o tA	Sn~TAn, mVl~j	NN
16	i o	{deleted}		
17	kE	{deleted}		
18	mVlk	E Er i kE o r	JIGAlSA, prIcITI	NN
19	n		GUrCE, JITE, krECE	VB
20	nI		@nEn, JAnEn	VB
21	pUr	kE	gUrUdAS, sAJAhAn	NNP
22	r	{deleted}		
23	tA	E Er i kE o	mEJAJ, sUrU	NN
24	y	r	Pr~mUIA, S~bl~pTA,	NN
25	yEr		rsmAlAi, wAlAi	NN

### 3.4.3 Populating the Clusters with Words

Next, for each word  $w$  in our vocabulary, we check whether  $w$  can be assigned to any of the clusters in  $C$ . Specifically, we assign  $w$  to a cluster  $c$  if  $w$  can combine with each of its primary keys and at least *half* of its secondary keys. Note that, the more secondary keys we use, the better is the precision of the clusters. Hence, if all primary keys and all secondary keys are used instead, then the clusters are morphologically the strongest (with a very high precision), although the coverage of each cluster would be poor, as only a few words

combine with all plausible suffixes in a naturally occurring corpus. The number of secondary keys we use also determines the noise-tolerance of our clustering algorithm against the imperfect unsupervised morphological analyzer. Presumably, employing more secondary keys makes the clustering more noise-tolerant. For example, even though our unsupervised morphological analyzer incorrectly segments “ornament” as “orna”+“ment”, “orna” is not labeled as a verb like “abandon” or “govern” (which also attach to “ment”), since “orna” fails to attach to any of secondary keys of cluster #18 (e.g., “ed”, “ing” and “s”). We will investigate the effect of the number of secondary keys on performance in the evaluation section.

#### **3.4.4 Labeling and Merging the Initial Clusters**

After populating each cluster with words, we manually label each of them with a POS tag from the tagset. As shown in the last column of Tables 12 and 13, all of the clusters are labeled as one of the base POS classes i.e., NN, VB, or JJ. We then merge all the clusters labeled with the same POS tag, yielding only three “big” clusters. Note that these “big” clusters are *soft* clusters, since a word can belong to more than one of them. For instance, “cool” can combine with “s” or “ing” to form a VB, and it can also combine with “er” or “est” to form a JJ.

#### **3.4.5 Generating and Labeling Sub-clusters**

Now, for each cluster  $c$  in  $C$ , we create one sub-cluster  $c_x$  for each suffix  $x$  that appears as a primary or secondary key in  $c$ . Then, for each word  $w$  in  $c$ , we use our unsupervised morphological analyzer to generate  $w+x$  and add the surface form to the corresponding sub-cluster. We manually label each sub-cluster with a POS tag from our tagset. For example, all

the words ending in “ing” will be labeled as VBG. As before, we merge two clusters if they are labeled with the same POS tag. The resulting clusters are our morphologically formed clusters.

**Table 14.** POS distribution of morphologically formed clusters

English		Bengali	
POS	# of word types	POS	# of word types
JJ	953	JJ	1394
JJR	68	NN	11519
JJS	57	NN2	2796
NN	1616	NN6	5548
NNS	527	NN7	5098
RB	299	NNP	1043
VB	2347	NNS	1841
VBD	1911	NNSH	1807
VBG	1629	VB	2398
VBZ	1566	VCN	378
Total	10.2K	Total	32.7K

At the end of morphological clustering, we have 10.1K and 32.7K English and Bengali word types labeled with their corresponding POS tags. The huge number of Bengali words labeled at this point shows that Bengali is morphologically a lot richer than English. In Table 14, we show the distribution of POS classes for both English and Bengali. As we can see, the verb (VB) cluster is the largest for English, suggesting that verbs in English have the strongest morphological property. Although only a few comparative adjectives (JJR) and superlative adjectives (JJS) are labeled, we will see later that they are sufficient as far as bootstrapping is concerned, as JJR and JJS can be easily identified by their word ending suffixes. For Bengali, noun (NN) is the largest and adjective (JJ) is the smallest of the base POS classes. Even though there is just one adjective (JJ) cluster in Bengali (see cluster #9 in Table 13), it gets expanded at the sub-cluster level as words ending with suffixes like “h\*N” (হীণ), “mVlk” (মূলক) are all labeled as JJ. Bengali proper nouns (NNP) are also scarce, which

suggests that many of them are added to the noun clusters incorrectly. A big problem, though, is the failure to identify any adverbs in Bengali. Note that, basic adverbs (i.e., adverb not containing any suffix e.g., D~rUT (ক্ষত), jldI (জলদি) etc.) are rare in Bengali. Adverbs that end with a suffix like jOrE (জোরে), DIrE (ধীরে), nIr~mmBAbE (নির্মমভাবে) are realized as NN7 in our clustering algorithm, which is reasonable if not linguistically justified.

Labeling the clusters is the only human intervention needed by our unsupervised POS acquisition system. For English, we had to manually label 19 initial clusters and  $19 \times 4 = 76$  (assuming that, on average, there are 4 keys per cluster) sub-clusters. Similarly for Bengali, we had to label approximately 100 different clusters only. In contrast, Clark’s (2003) and Schütze’s (1995) POS induction systems require more human intervention than ours: Clark’s best performing clustering algorithm labels 128 clusters, whereas Schütze uses Buckshot to cluster the word tokens into 200 classes.

We also checked the effect of the threshold  $t$  on our clustering performance. Intuitively, if we set it too low, many suffixes are clustered together even though they are not similar. On the other hand, setting it too high will make the suffix clusters totally disjoint. As a sanity check, we reran our clustering algorithm with  $t$  set to a value close to 0.4 (see Table 15). When  $t=0.3$ , the largest verb cluster in English i.e., {ed, es, ing, s} gets removed, as our algorithm finds it ambiguous (see Section 3.4.2). In addition, many clusters end up with one primary key and no secondary keys, thus making it more prone to morphological segmentation errors. When we set  $t$  to 0.6, the primary key clusters look exactly the same as the clusters derived using  $t=0.4$ . However, many of the clusters now have none or just one secondary key, as the threshold of 0.6 is too strong to induce secondary keys.

**Table 15.** Morphological clusters after setting the threshold differently

$t = 0.3$		$t = 0.6$	
Primary key	Secondary key	Primary key	Secondary key
ability	able er	ability	able ed ing s
able	ed er es ing s	able	ed ing s
al ally		al	s
		ally	
ation		ation	ed ing s
ed es ing s	<b>{deleted}</b>	ed ing	s
er	<b>{deleted}</b>	er	ed ing s
est	er ly ness	est	Er
ful		ful	s
ic		ic	s
ion	ed er es ing s	ion	ed ing s
ism ist		ism	s
		ist	
ity	ly	ity	
ive	ed er es ing s ion	ive	ed ing s ion
ize ized	ly	ize ized	
less		less	s
ly	<b>{deleted}</b>	ly	<b>{deleted}</b>
ment	ed er es ing s	ment	ed ing s
ness	er ly	ness	ly
		s	<b>{deleted}</b>
y		y	s

### 3.5 Purifying the Seed Set

The clusters formed thus far cannot be expected to be perfectly accurate, since (1) our unsupervised morphological analyzer is not perfect, and (2) morphology alone is not always sufficient for determining the POS of a word. In fact, we found that many adjectives are mislabeled as nouns for both languages. For instance, “historic” is labeled as a noun, since it combines with suffixes like “al” and “ally” that “accident” combines with. In addition, many words are labeled with the POS that does not correspond to their most common word sense. For instance, while words like “chair”, “crowd” and “cycle” are more commonly used as nouns than verbs, they are labeled as verbs by our clustering algorithm. The reason is that

suffixes that typically attach to verbs (e.g., “s”, “ed”, “ing”) also attach to these words. Such labelings, though not incorrect, are undesirable, considering the fact that these words are to be used as seeds to bootstrap our morphologically formed clusters in a distributional manner. For instance, since “chair” and “crowd” are distributionally similar to nouns, their presence in the verb clusters can potentially contaminate the clusters with nouns during the bootstrapping process. Hence, for the purpose of effective bootstrapping, we also consider these words “mislabeled”.

To identify the words that are potentially mislabeled, we rely on the following assumption: words that are morphologically similar should also be distributionally similar and vice versa. Based on this assumption, we propose a *purification* method that posits a word  $w$  as potentially mislabeled (and therefore should be removed or relabeled) if the POS of  $w$  as predicted using distributional information differs from that as determined by morphology.

The question, then, is how to predict the POS tag of a word using distributional information? Our idea is to use “supervised” learning, where we train and test on the seed set. Conceptually, we (1) train a *multi-class* classifier on the morphologically labeled words, each of which is represented by its context vector, and (2) apply the classifier to relabel the *same* set of words. If the new label of a word  $w$  differs from its original label, then morphology and context disagree upon the POS of  $w$ ; and as mentioned above, our method then determines that the word is potentially misclassified. Note, however, that (1) the training instances are not perfectly labeled and (2) it does not make sense to train a classifier on data that is seriously mislabeled. Hence, we make the assumption that a large percentage ( $> 70\%$ ) of the

training instances is correctly labeled<sup>21</sup>, and that our method would work with a training set labeled at this level of accuracy. In addition, since we are training a classifier based on distributional features, we train and test on only *distributionally reliable* words, which we define to be words that appear at least five times in our corpus. Distributionally unreliable words will all be removed from the morphologically formed clusters, since we cannot predict their POS using distributional information only.

In our implementation of this method, rather than training a multi-class classifier, we train a set of binary classifiers using SVM<sup>light</sup> (Joachims, 1999) together with the distributional features for determining the POS tag of a given word.<sup>22</sup> More specifically, we train one classifier for each pair of POS tags. For instance, since we have ten POS tags for English, we will train 45 binary classifiers.<sup>23</sup> 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  $w$ .<sup>24</sup> 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.

---

<sup>21</sup> An inspection of the morphologically formed clusters reveals that this assumption is satisfied for both languages.

<sup>22</sup> In this and all subsequent uses of SVM<sup>light</sup>, we set all the training parameters to their default values.

<sup>23</sup> 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.

<sup>24</sup> If a word is ambiguous i.e., labeled both NN and JJ by our clustering algorithm, then we do not consider it for training and testing of the NN-JJ classifier.

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<sup>25</sup> 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 the words in the NN and NN2 clusters in

---

<sup>25</sup> 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.

Bengali. 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 Section 3.4. The feature corresponding to suffix  $x$  has the value 1 if and only if  $x$  is the suffix of  $w$ . Moreover, we create an additional suffixal feature (i.e., the “null” suffix) whose value is 1 if and only if none of the 30 most frequent suffixes is the suffix of  $w$ .

It is worth noting that suffixal features are different from the morphological features (we call them *signature features*) as defined in Section 3.4. For example, “worker” has the suffixal feature “er” and the signature feature in “s” (since “worker” can combine with “s” to form “workers”). On the other hand, “walk” has the “null” suffixal feature, but four signature features (i.e., “ed”, “ing”, “s” and “er”). For clarification, below we define the suffixal and signature features, both of which are binary valued:

**Suffixal Features ( $f_x$ ):** If a word  $w$  can be segmented as  $w'+\alpha$  and  $\alpha$  is one of the 30 suffixes we consider, then  $f_x(\alpha) = 1$  and  $\forall \beta \neq \alpha : f_x(\beta) = 0$ . If  $w$  can't be segmented at all,  $f_x(\text{"null"}) = 1$  and  $\forall \beta \neq \text{"null"} : f_x(\beta) = 0$ .

**Signature Features ( $f_s$ ):** If a word  $w$  can combine with a suffix  $\alpha$  (i.e., the surface form of  $w+\alpha$  is found in the vocabulary) and  $\alpha$  is one of the 30 suffixes we consider, then  $f_s(\alpha) = 1$ . More specifically, given a vocabulary  $V$ ,

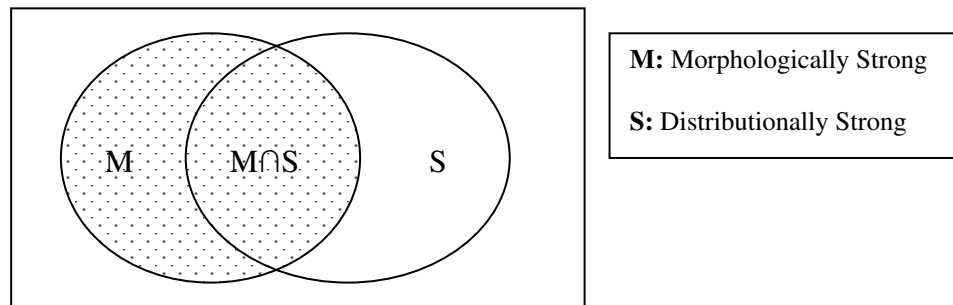
$$\forall \beta : f_x(\beta) = 1, \text{ where } w + \beta \in V.$$

$$\text{and } \forall \beta : f_x(\beta) = 0, \text{ where } w + \beta \notin V.$$

Our initial English morphological clusters have 10.1K word types, out of which 4.8K word types are of frequency at least 5. We run our SVM purification classifier on these 4.8K word types and identify 700 word types (14%) as incorrectly labeled. Hence, after removing those, we have 4.1K word types, which we use as a seed set for bootstrapping. For Bengali,

out of the 32.7K word types in the morphological clusters, 18K words are of frequency at least 5. Our purification classifier determines that 1.7K (9.4%) of those are incorrect; hence we use the remaining 16.3K words as our seeds.

### 3.6 Bootstrapping Distributionally Reliable Words



**Figure 2.** Morphology and context distribution

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 with words that are distributionally reliable. Figure 2 depicts the morphological and contextual distribution of a particular language. So far, we have labeled the words that are in the dotted portion of the diagram (i.e., the words that are morphologically reliable only (M-S) and the words that are both morphologically and distributionally reliable ( $M \cap S$ )). In this section, we describe how we induce additional words that are distributionally strong (S-M), using the words in  $M \cap S$  as seeds.

There are in total 10K English and 41K Bengali word types in our dataset with frequency at least 5. Out of those, 4.1K and 16.3K word types have already appeared in our seed clusters (i.e., in  $M \cap S$ ). Our goal here is to see how many of the remaining words can be labeled confidently using just distributional features. Since we have a small seed set 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.<sup>26</sup> 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 added to the seed set. In the next iteration, we retrain the classifier on the augmented labeled data, apply it to the unlabeled data, and add to the labeled data those instances whose predicted confidence is above the current threshold. If none of the instances has a predicted confidence above the current threshold, we reduce the threshold by 0.1. (For instance, if the original thresholds are 1 and -1, they will be changed to 0.9 and -0.9.) We then repeat the above procedure until the thresholds reach 0.5 and -0.5.

We decided to stop the bootstrapping procedure at thresholds of 0.5 and -0.5 for two reasons. First, our goal here is to induce distributionally reliable words only; a threshold of 0.5 ensures that all the bootstrapped words are classified by SVM with confidence greater than 0.5. Second, the more bootstrapping iterations we use, the lower are the quality of the

---

<sup>26</sup> 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.

bootstrapped data as well as the accuracy of the bootstrapped classifier. In fact, we found that the accuracy of bootstrapped classifier drops even before we reach the thresholds of 0.5 and - 0.5. For example, for Bengali, we started with the seed set of 16.3K words and added 28K additional words (close to double the number of seed words) when we stopped the bootstrapping process at the threshold of 0.5. Since the seed words are supposedly the ones that are most reliably labeled, augmenting the seed set with too many unlabeled words (i.e., 28K) essentially reduces the effect of the more reliable seed words. One way to alleviate this problem is to stop the bootstrapping process when the augmented dataset gets bigger than the seed set. Having taken this into account, we decided to stop the bootstrapping process at thresholds of 0.6 and 0.8 for English and Bengali respectively.

In our implementation of the self-training algorithm, rather than train a multi-class classifier in each bootstrapping iteration, we train pairwise classifiers (recall that for English, 45 classifiers are formed from 10 POS tags) using the distributional and suffixal features described in the previous section. Again, since we employ distributional features, we apply the 45 pairwise classifiers only to the distributionally reliable words (i.e., words with corpus frequency at least 5). To classify an unlabeled word  $w$ , we apply the 45 pairwise classifiers to independently assign a label to  $w$ .<sup>27</sup> We then count how many times  $w$  is tagged with each of the ten POS tags. If there is a POS tag whose count is nine and all of these nine votes are associated with confidence that exceeds the current threshold, then we add  $w$  to the labeled

---

<sup>27</sup> As in purification, each pairwise classifier is implemented as a set of 10 classifiers, each of which is trained on an equal number of instances from both classes. Testing also proceeds as before: the label of an instance is derived from the average of the confidence values returned by the 10 classifiers, and the confidence value associated with the label is just the average of the 10 confidence values.

data together with its assigned tag. Finally, the number of new labeled instances is 3.8K and 15K for English and Bengali respectively.

### 3.7 Classifying Morphologically and Distributionally Poor Words

So far, we have clustered words which are either morphologically or distributionally reliable. But there are still many words (mostly of frequency less than 5) that could not be labeled reliably using morphological and distributional features. Therefore, we use a multi-class SVM classifier trained on the words labeled so far to classify the words that are both morphologically and distributionally poor.<sup>28</sup> We use both morphological and distributional features (i.e., 1000 left and right context features, 30 suffixal features and 30 signature features) to learn this classifier. Table 16 shows the distribution of the training and test set of the classifier. As we can see, we have a fairly even distribution of morphologically and distributionally strong words in the training set. Note that the words that are only morphologically strong (M-S) are distributionally poor, whereas the words that are only distributionally strong (S-M) are morphologically poor – which suggests that our training set has equal share of morphologically or distributionally poor words too. Intuitively, the presence of morphologically and distributionally poor words will help us in the classification process.

**Table 16.** Distribution of final training and test set

	Training Set				Test Set
	M – S	$M \cap S$	S – M	Total	Total
<b>English</b>	5.3K	4.1K	3.8K	13.2K	7.8K
<b>Bengali</b>	14.7K	16.3K	15K	46K	45K

---

<sup>28</sup> As before, the multi-class classifier is replaced by a set of binary classifiers

This classifier labels all the remaining words in our vocabulary. In fact, at the end of this step, we cluster additional 7.8K English and 45K Bengali word types. So, our final coverage of the POS clusters is 21K and 90K for English and Bengali respectively. In the next section, we will evaluate how accurate our clustering is.

### 3.8 Evaluation

**Corpora.** Recall that our bootstrapping algorithm assumes as input an unannotated corpus from which we (1) extract our *vocabulary* (i.e., the set of words to be labeled) and (2) collect the statistics needed in morphological and distributional clustering. We use as our English corpus the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993). Our Bengali corpus is composed of five years of articles taken from the Bengali newspaper *Prothom Alo*.

**Vocabulary creation.** To extract our English vocabulary, we pre-processed each document in the WSJ corpus by first tokenizing them and then removing the most frequent 500 words (as they are mostly closed class words), punctuations, capitalized words (as they are mostly proper nouns), numbers, words of length two or less and unwanted character sequences (e.g., “\*\*\*”). We also removed any word whose feature vector is too sparse (i.e., less than two contextual, suffixal and signature features have a non-zero value). The resulting English vocabulary consists of approximately 21K word types. We applied similar pre-processing steps to the Prothom Alo articles to generate our Bengali vocabulary, which consists of 91K word types.

**Test set preparation.** Our English test set is composed of all the word types in the vocabulary that have been labeled at least once with an open class tag in the WSJ. The gold-standard POS tags for each word  $w$  are derived automatically from the parse trees in which  $w$

appears. To create the Bengali test set, we randomly chose 6K word types from our vocabulary. Each word in the test set was then labeled with its POS tags by two of our linguists.

**Evaluation metric.** Following Schütze (1995), we report performance in terms of recall, precision, and F1. Recall is the percentage of POS tags correctly proposed, precision is the percentage of POS tags proposed that are correct, and F1 is simply the harmonic mean of recall and precision. To exemplify, suppose the correct tagset for “crowd” is {NN, VB}; if our system outputs {VB, JJ, RB}, then recall is 50%, precision is 33%, and F1 is 40%. Importantly, all of our results will be reported on *word types*. This prevents the frequently occurring words from having a higher influence on the results than their infrequent counterparts.

### 3.8.1 Results and Discussion

**The baseline system.** We use as our baseline system one of the best existing unsupervised POS induction algorithms (Clark, 2003). More specifically, we downloaded from Clark’s website<sup>29</sup> the code that implements a set of POS induction algorithms he proposed. Among these implementations, we chose *cluster\_neyessenmorph*, which combines morphological and distributional information and achieves the best performance in his paper. When running his program, we use WSJ and Prothom Alo as the input corpora. In addition, we set the number of clusters produced to be 128, since this setting yields the best results in his paper.

**Our induction system.** Recall that our unsupervised POS induction algorithm operates in four steps. To better understand the performance contribution of each of these steps, we show

---

<sup>29</sup> <http://www.cs.rhul.ac.uk/home/alex/>

in Tables 17a, 17b, 18a and 18b the results of our system after we (1) morphologically cluster the words, (2) purify the seed set, (3) bootstrap the seed set, and (4) label the distributionally and morphologically poor words. Importantly, the numbers shown for each step are computed over the set of words in the test set that are labeled at the end of that step. For instance, the morphological clustering algorithm labeled 10K English words and 32.7K Bengali words types, and so recall, precision and F1-score are computed over the subset of these labeled words that appear in the test set. Similarly, results of the baseline system are computed over the set of words our clustering algorithm labels at the end of each step.

As we can see from row 2 of Tables 17a and 18a, after morphological clustering, our system achieves F1-scores of 80.8% and 81.8% for English and Bengali, respectively. When measured on exactly the same set of words, the baseline only achieves F-scores of 46.1% and 65.6%. In addition, our F-score for VB is more than 84%, which suggests that verbs in English can be identified easily using morphological signature features. However, this is not true for other two base classes (i.e., NN and JJ), as we got only 62% and 63% F-score for NN and JJ respectively. Even though the precision of the NN and JJ clusters are very good (90%), their low recall levels suggest that many of them were mislabeled. In Table 19, we show the most common English POS confusions after the morphological clustering step. As we can see, a large proportion of the errors occur due to NN vs. VB and NNS vs. VBZ confusions. As mentioned before, many of the commonly used NNs (e.g., “cycle”, “chair”, “document”) in English are identified as VBs by our system as they attach with similar set of suffixes. Although these words can be used as both nouns and verbs, our Gold Standard erroneously labels them as NNs only, as they are only used as NNs in WSJ. If we could hand-label all the words in WSJ, the NN vs. VB confusions would have been much less.

**Table 17a.** POS induction results for English based on word type

	After Morphological Clustering (10K)			After Purification (9.3K)		
	P	R	F1	P	R	F1
<b>Baseline</b>	52.7	40.9	46.1	50.7	39.9	44.6
<b>Ours</b>	86.1	76.3	<b>80.8</b>	88.1	78.7	<b>83.1</b>
<b>JJ</b>	89.7	49.4	63.7	91.4	51.9	66.2
<b>JJR</b>	98.4	86.1	91.8	98.4	92.4	95.3
<b>JJS</b>	100	98.3	99.2	100	100	100
<b>NN</b>	90.1	47.9	62.6	90.8	50	64.4
<b>NNS</b>	89.4	38.2	53.5	89.2	41.3	58.4
<b>RB</b>	100	75.9	86.3	100	81.6	89.6
<b>VB</b>	74.8	97.8	84.7	80.2	97.4	88.1
<b>VBD</b>	96.8	98.8	97.8	96.8	99	97.9
<b>VBG</b>	89.7	100	94.6	89.8	100	94.6
<b>VBZ</b>	60.1	98.9	74.8	63.7	99.3	77.6

**Table 17b.** POS induction results for English based on word type

	After Bootstrapping (13.2K)			After Labeling Morphologically and distributionally Poor Words (21K)		
	P	R	F1	P	R	F1
<b>Baseline</b>	55.4	43.6	48.8	51	42.7	46.5
<b>Ours</b>	86.6	75.8	<b>80.9</b>	74.3	66.3	<b>70.1</b>
<b>JJ</b>	79.3	61.3	69.2	57.2	73.9	64.5
<b>JJR</b>	86.7	90.3	88.4	48.9	81.2	61.1
<b>JJS</b>	100	96.7	98.4	96.9	79.1	91
<b>NN</b>	91.4	51.1	65.6	90.8	45.3	60.1
<b>NNS</b>	93.5	52.9	67.6	95.6	35.8	52.1
<b>RB</b>	99.2	71.2	82.9	97.1	64.2	77.2
<b>VB</b>	77.8	95.2	84.7	73.2	92.7	81.8
<b>VBD</b>	96.9	94.6	95.7	92.4	85.2	88.7
<b>VBG</b>	90.1	95.6	92.7	87.5	87.6	87.5
<b>VBZ</b>	62.7	96.1	75.8	37.4	93.3	53.4

**Table 18a.** POS induction results for Bengali based on word type

	After Morphological Clustering (32.7K)			After Purification (31K)		
	P	R	F1	P	R	F1
<b>Baseline</b>	67	64.2	65.6	67.9	65.2	66.5
<b>Ours</b>	83.5	80.4	<b>81.8</b>	86.6	84.7	<b>85.6</b>
<b>JJ</b>	78.5	54.8	64.5	83.3	62.8	71.6
<b>NN</b>	70.1	95.7	80.7	79.6	90.9	84.9
<b>NN2</b>	98.1	100	99.1	98.1	100	99.1
<b>NN6</b>	95.9	90.3	93	96.8	90.6	93.6
<b>NN7</b>	85.8	98.5	91.7	89.1	98.8	93.7
<b>NNP</b>	83.1	36	50.2	87.8	46.8	61.1
<b>NNS</b>	93.1	97.3	95.2	93.1	98.2	95.6
<b>NNSH</b>	98	100	99	98	100	99
<b>VB</b>	74.6	87.8	80.7	82.2	89.5	85.7
<b>VBN</b>	84.3	54.4	66.2	76.8	57.3	65.6

**Table 18b.** POS induction results for Bengali based on word type

	After Bootstrapping (46K)			After Labeling Morphologically and Distributionally Poor Words (91K)		
	P	R	F1	P	R	F1
<b>Baseline</b>	68	65.1	66.5	62.6	60.1	61.3
<b>Ours</b>	84.8	82.3	<b>83.6</b>	74.1	72.3	<b>73.2</b>
<b>JJ</b>	87.2	65.6	74.9	82.9	50.1	62.4
<b>NN</b>	74.7	89.3	81.3	63.1	83.4	71.8
<b>NN2</b>	98.1	96.5	97.3	97.4	94.1	95.7
<b>NN6</b>	96.9	91.9	94.4	95.6	88.6	91.9
<b>NN7</b>	87.2	95.2	91.1	85.7	84.4	85.1
<b>NNP</b>	88.8	44.8	59.6	83.6	36.5	50.8
<b>NNS</b>	90.9	96.2	93.5	85.2	89.7	87.4
<b>NNSH</b>	98	100	99	98.3	96.6	97.4
<b>VB</b>	84.4	88.4	86.4	73.6	79.4	76.4
<b>VBN</b>	59.1	57.1	58.1	33.9	53.5	41.5

**Table 19.** Most common POS confusions after morphological clustering

English			Bengali		
Correct Tag	Predicted	% of Errors	Correct Tag	Predicted	% of Errors
NN	VB	35%	NNP	NN	47%
NNS	VBZ	29%	JJ	NN	8%
NN	VBG	7%	NN	VB	4%
NN	JJ	5%	VCN	NN7	4%
JJ	VBD	3%	NN	NNP	3%

For Bengali, the F-scores for NN and VB are approximately 80%, whereas the F-score for JJ is only 65%. This suggests many of the JJs in Bengali are mislabeled. The biggest error cluster, though, is NNP (with an F-score of 50%), as many of them are incorrectly labeled as common nouns. The reason is that (1) there are not enough suffixes to induce proper noun clusters in the initial list of morphologically induced clusters, and (2) NN and NNP share almost similar morphological signature features. As we can see from Table 19, 47% of the errors in Bengali occur due to proper noun vs. common noun confusions.

**Table 20.** Morphological clustering according to secondary key constraints

Secondary Keys	English		Bengali	
	F-score	Coverage	F-score	Coverage
Zero	76%	12.3K	78.9%	25.6K
Half	80.8%	10.1K	81.8%	32.7K
All	81%	7.1K	81.8%	25.6K

We also checked whether the key constraints as described in Section 3.4 can improve the morphological clusters. The morphological clusters used so far are formed by checking all the primary keys and half of secondary keys. As shown in Table 20, if we check all the secondary keys, we get a small improvement in terms of F-score (81% compared to 80.8%) for English, but end up clustering only 7K word types, which is too small for bootstrapping. As expected, if we do not check any of the secondary keys, the F-score drops from 80.8% to

76% for English. Similar results have been reported for Bengali. These results seem to suggest that it would be best to use half of the secondary keys.

The clusters described thus far are formed using only morphological features. Next we check how much improvement we can get when distributional and suffixal features are employed to purify the initial clusters. Given 10.1K English and 32.7K Bengali word types in our morphological clusters, the purification step found 14% and 10% of the word types to be incorrectly labeled and subsequently removed them from the initial clusters. As shown in the second column of Tables 17a and 18a, the F-score improves significantly (2.3% and 3.8% for English and Bengali respectively) after the purification step, showing that the removed words are indeed incorrect. A closer examination of the results reveals that, for English, the purification procedure successfully identifies many nouns in the VB cluster, as they are not distributionally similar to other words in the cluster. Similarly, F-scores of the NNS, VBZ and JJ classes improve, as many of the <NNS, VBZ> and <NN, JJ> confusions are correctly identified and eliminated. For Bengali, the F-score of the JJ class improves from 64% to 71%, while the NN and VB classes show similar improvements (4-5% in F-score). The biggest improvement, though, comes from the proper noun class, where the F-score increases from 51% to 61%. Still, the poor accuracy of the proper noun class (only 61%) shows that proper nouns in Bengali are hard to distinguish from common nouns even when both distributional and morphological features are used.

Tables 17b and 18b show the results of our system after bootstrapping and final augmentation with distributionally and morphologically poor words. During bootstrapping we labeled 4K English word types, thus increasing our overall coverage to 13.2K word types. Among the newly added words, 42% are JJ, 31% are NN, 8% are NNS and 7% are VB. As

we can see, the F-scores for JJ, NN and NNS improve during bootstrapping, while those for VB, VBZ, RB and others drop considerably. The F-score on 13.2K English word types after the bootstrapping step is 80.9%, which is lower than the F-score we achieved after the purification step (83.1%). As expected, the F-score drops further with the addition of words that are both morphologically and distributionally poor. As we can see, the final F-score for English is only 70% after labeling all the words in our vocabulary.

**Table 21.** Most common POS confusions after the final augmentation

English			Bengali		
Correct Tag	Predicted	% of Errors	Correct Tag	Predicted	% of Errors
NNS	VBZ	26%	NNP	NN	31%
NN	JJ	19%	JJ	NN	15%
NN	VB	19%	NN7	VCN	5%
NN	VBG	4%	NN	VCN	4%
JJ	VBD	3%	NN	JJ	3%

While the F-scores of all the POS classes drop after the final augmentation step, those of NNS, VBZ and JJR drop particularly abruptly for English. The poor results for VBZ (the third person singular form of a verb) and NNS (plural noun) can be attributed to the fact that both of the POS classes end with the suffix “s” and it is hard to classify them using just distributional features. The situation is further complicated by the fact that our initial VBZ clusters were originally populated with many NNS, as the two POS classes were found to be similar morphologically. Unfortunately, many of those words could not be removed in the purification step, and as a result, when we bootstrapped and augmented, the VBZ cluster was further contaminated with more NNS. Another major source of error stems from nouns ending with the suffix “er” (e.g., backer, beeper) being labeled as JJR. Many of these nouns have poor distributional features, thus causing SVM to classify based on suffixal features

rather than distributional features. Table 21 shows the most common POS induction errors after we cluster all the words in our vocabulary.

For Bengali, the F-score after bootstrapping is almost similar (83.6% compared to 85% as achieved after the purification step), even though we added 15K words. As we can see from Table 18b, both the F-scores for JJ and VB get improved, whereas that for NN drops after the bootstrapping (because many NNPs are mislabeled as NNs). The F-score for proper nouns is still poor for the same reason. On the other hand, the F-score for VBN drops particularly significantly during bootstrapping, as many of the words in NN7 ending with suffix “E” are identified as VBN. As expected, the F-score dropped further after we augmented with morphologically and distributionally poor words. Nevertheless, considering the fact that 91K word types POS-tagged at the end of final augmentation, the accuracy of 73.2% is not particularly bad.

We also checked the most common POS errors for Bengali (see Table 21). Expectedly, proper noun vs. common noun confusion constitutes the biggest percentage of the error (31%). Even though we achieve a high precision for NNP, its low recall implies that many NNPs are mislabeled as NNs. Results for adjectives are not good either, since (1) adjectives and nouns have very similar distributional property in Bengali and (2) there are not enough suffixes to induce the adjectives morphologically. Finally, the F-score for VBN drops to 41%, mainly because words ending with the suffixes “i” (ই) and “o” (ও) are mostly identified as VBN. Note that “i” and “o” can grammatically attach to any of the Bengali POS tags. However, as the percentage of the words ending with “i” and “o” in the VBN cluster is bigger than that of other clusters after the bootstrapping step, SVM tends to classify distributionally poor words ending with suffix “i” and “o” as VBN.

**Comparison against the baseline.** Comparing against the Clark’s unsupervised POS induction system (see rows 1 and 2 of Tables 17a, 17b, 18a and 18b), it is clear that we outperform the baseline in each of the four steps quite significantly. In particular, our system yields F1-scores of 70% and 73% after the final augmentation step, thus outperforming the baseline by 24% and 10% for English and Bengali, respectively. Clark’s algorithm performs particularly poorly on English. This can be attributed to the fact that Clark’s algorithm performs poorly on the low frequency words (see Section 3.8.2), which form more than 50% of our English test cases. In fact, when we tested only on words of frequency greater than 1, Clark’s algorithm performs significantly better, achieving an F-score of 61% for English. On the other hand, Clark’s algorithm performs much better on Bengali than English owing to the fact that low frequency words comprise only 20% of the test cases in Bengali.

### 3.8.2 Additional Experiments

**Labeling rare words.** Although our discussion thus far has focused on words of any frequency, it would be informative to examine how well our algorithm performs on *rare* words only (i.e., words with corpus frequency less than five). In fact, these rare words comprise more than 50% of the English and Bengali words in our vocabulary. While measuring the performance on just rare words, our algorithm achieves F1-scores of 64.5% and 69.1% for English and Bengali, respectively. Interestingly, when we consider the rare words that are in the morphologically strong clusters only (i.e., that are morphologically strong), F-score shoots up to 81% and 85% for English and Bengali respectively. This provides empirical support that morphological information plays a very important role in labeling the rare words (Clark, 2003). We also checked how Clark’s algorithm (baseline) performs on the rare words. Interestingly, it achieves F-scores of only 30.1% and 39% for

English and Bengali respectively. This shows that it fails considerably on low frequency words. In fact, we outperform Clark’s algorithm by more than 30% while testing only on rare words. Presumably, our algorithm’s effective use of morphological features (both signature and suffixal features) gives us better results on rare words.

**Performance vs. SVM confidence.** In Tables 17b and 18b we show that the F-scores for all English and Bengali word types are 70.1% and 73.2% respectively. These F-scores may not seem satisfactory to some readers. These mediocre results can be attributed in large part to the inaccurate labeling of the morphologically and distributionally poor words. Proper handling of these words is a big challenge to any unsupervised POS induction system. In our case, we trained a SVM multi-class classifier using morphologically or distributionally strong words, and used it to label the words that are distributionally and morphologically poor. One appealing feature of our approach is that when labeling each word with its POS tag, an SVM classifier also returns a numeric value that indicates how confident the word is labeled. We took advantage of this feature and checked how our system performs on the distributionally and morphologically poor words for which SVM is confident (i.e., the SVM threshold is greater than some predefined threshold). As you can see from Table 22, when we increase the SVM threshold from 0.2 to 0.8, F-score improves by 6-7% for both English and Bengali, although the overall coverage (i.e., the number of words clustered) goes down. Nevertheless, by setting a threshold of 0.8, we clustered in total 14.5K English word types with a F-score of 78.9% and 70K Bengali word types with a F-score of 81.9%.

**Table 22.** Performance vs. SVM confidence

SVM Confidence	English		Bengali	
	F-score	Coverage	F-score	Coverage
0 (considering all)	70.1	21K	73	91K
0.2	71	19.5K	75.5	88K
0.5	73.6	18K	77.9	80K
0.6	74.8	17K	79.5	78K
0.8	78.9	14.5K	81.9	70K

**Performance vs. corpus frequency.** Many of the morphologically and distributionally poor words are low frequency words. However, in many practical situations, we may only be concerned with labeling the frequent words. To get an idea of how our system performs on the frequent words, we show in Table 23a the F-scores of those words that occur with a certain corpus frequency. As you can see, when tested on words with frequency at least 5, we achieve F-scores of 76% and 75% for English and Bengali respectively. For Bengali, F-score improves only slightly (73% to 75%), as only 20% of words in the Bengali test set are low frequency words. Interestingly, Clark’s algorithm achieves F-scores of 66.8% and 66.4% on words with frequency at least 5 (see Table 23b). This shows that even for high frequency words, our algorithm outperforms Clark’s algorithm by approximately 10% for both the languages. Note that Clark’s algorithm depends on distributional features for high frequency words, whereas morphology is used only for labeling low frequency words. However, our result suggests that morphology is equally important for high frequency words.

**Table 23a.** Our algorithm’s performance vs. corpus frequency

Frequency	English		Bengali	
	F-score	Coverage	F-score	Coverage
>0 (considering all)	70.1	21K	73.2	91K
>1	73.2	14.5K	74.4	67K
>2	74.8	11.3K	74.8	54.9K
>4	76.4	9K	75.2	41.6K

**Table 23b.** Clark’s algorithm’s performance vs. corpus frequency

Frequency	English F-score	Bengali F-score
>0 (considering all)	46.5%	61.3%
>1	52.4%	63.5%
>2	58.2%	65.6%
>4	66.8%	66.4%

**Tokenized vs. typed result.** So far, we have reported the F-scores on word types. This ensures that the frequently occurring words do not have a higher influence on the results than their infrequent counterparts. In fact, when we checked the tokenized F-scores (by multiplying each typed F-score with the corresponding word frequency), we found that the tokenized F-scores are 3-5% better than the typed F-scores for both of the languages (see Table 24). This can be attributed to our system’s failure to correctly label low frequency words. Similar results have been reported for Clark’s algorithm, which achieves F-scores of 64.8% and 66.5% for English and Bengali respectively. These results are significantly better than what it achieves on the typed version (46.5% and 61.3% respectively).

**Table 24.** Typed vs. tokenized results

	English			Bengali		
	P	R	F1	P	R	F1
<b>Typed</b>	74.3	66.3	70.1	74.1	72.3	73.2
<b>Tokenized</b>	85.9	66.7	75.1	74.1	78.9	76.5

**Most frequent tagging.** As many words in English have more than one POS tag (i.e., ambiguous), it would be interesting to see whether our unsupervised system can actually induce the most frequent tag of a word. To investigate this question, we collected the frequency of each tag of each word type from WSJ, and formed our new gold standard by keeping only the most frequent tag for each word type. The typed precision, recall, F1-score shown in Table 25 are computed by comparing our output against the most frequent tag gold standard. As we can see, F-score drops by 3% while testing on the new most frequent gold

standard, revealing our system’s inability to produce the most frequent tag. A closer examination of the POS classes shows that while the F-scores of JJ, NN and RB improve, those of VB and VBZ drop significantly. In particular, the precision of VB drops by more than 20%. This can be attributed to the fact that many <NN, VB> ambiguous words (e.g., “chair”, “document” etc.) are labeled as NN in the new gold standard, as they are most commonly used as NN. However, as described before, our clustering algorithm tags many of those as VB, as they are morphologically similar to VB, thus failing to capture the most frequent tag. Note that our initial clusters, which were formed using morphological features, were later enriched with words that are distributionally similar. However, as the seeds are morphologically strong (although we purified it to some extent using distribution), morphology has a more impact on the results than context, which is the reason why we fail to get the most frequent tags in some cases.

Interestingly, when we compared Clark’s output against the most frequent tag Gold standard, we found out that F-score (46.3%) does not drop. The reason can be attributed to the fact that, Clark’s algorithm depends more on contextual features than on morphology (which is predominantly used to tag only low frequency words) and hence it can capture the most frequent tag more often.

**Table 25.** Most frequent tagging results for English

	<b>P</b>	<b>R</b>	<b>F1</b>
<b>Gold Standard</b>	74.3	66.3	70.1
<b>Most Frequent Tag Gold Standard</b>	66.8	67.5	67.2

**Soft clustering.** Although our morphological clustering algorithm can do soft clustering i.e., give more than one tags to each word type, it occurs only at the “big” cluster level (see Section 3.2). At the sub-cluster level, the algorithm imposes a hard clustering on the words.

In other words, no word appears in more than one sub-cluster, which means words like “received” are tagged as VBD only, although they can be JJ too. Ideally, a POS induction algorithm should produce soft clusters due to lexical ambiguity. In fact, Jardino and Adda (1994), Schütze (1997) and Clark (2000) have attempted to address the ambiguity problem to a certain extent.

We have also experimented with a very simple method for handling ambiguity in our bootstrapping algorithm: when augmenting the seed set, instead of labeling a word with a tag that receives 9 votes from the 45 pair-wise classifiers, we label a word with any tag that receives at least 8 votes, effectively allowing the assignment of more than one label to a word. However, as shown in Table 26, the result is not promising: improvements in recall are accompanied by larger drops in precision, causing the resulting F-score to drop. For Bengali, the result is even worse, as Bengali part-of-speech is mostly unambiguous.

**Table 26.** Results for English after incorporating ambiguous words

	<b>After Bootstrapping</b>			<b>After Final Augmentation</b>		
	<b>P</b>	<b>R</b>	<b>F</b>	<b>P</b>	<b>R</b>	<b>F</b>
<b>No Ambiguity Scheme</b>	86.6	75.8	80.9	74.3	66.3	70.1
<b>With Ambiguity Scheme</b>	78.3	79.8	79.1	55.4	85.1	67.1

## **CHAPTER 4**

### **CONCLUSIONS**

This thesis attempts to address two of the most important and fundamental tasks in natural language processing, namely morphological segmentation and part-of-speech induction. In contrast to existing algorithms developed for these problems, our algorithms are designed to operate under a resource-scarce setting in which no language-specific tools or resources are available. Specifically, we have presented a learning system where morphological analyzer and part-of-speech lexicon can be built automatically from just a text corpus without using any additional language specific grammatical knowledge. We also give the empirical support that our system is totally language independent i.e., it can be extended to many different languages.

Although there has been considerable work on unsupervised morphological segmentation, our approach is different from them in many aspects. First, we propose two novel techniques (i.e., relative corpus frequency and suffix level similarity) to detect word over-segmentation; they improve the precision quite significantly. Second, we automatically acquire allomorphs and orthographic change rules from an unannotated text corpus; this allows us to output the actual segmentation of the words that exhibit spelling changes during morpheme attachment. Finally, our system exhibits language independence, as it achieves robust performance across four different languages with different levels of morphological complexity. In fact, our algorithm not only outperforms *Linguistica* and *Morfessor* for English and Bengali, but also compares favorably to the best-performing PASCAL

morphological parsers when evaluated against all three target languages --English, Turkish, and Finnish -- in the Challenge.

In addition, we have proposed a new bootstrapping algorithm for unsupervised POS induction. In contrast to existing algorithms developed for this problem, our algorithm is designed to (1) operate under a resource-scarce setting, (2) more tightly integrate morphological information with the distributional POS induction framework, and (3) adjust well to languages where distributional features are not reliable enough. In particular, our algorithm (1) improves the quality of the seed clusters by employing seed words that are distributionally and morphologically reliable and (2) uses support vector learning to combine morphological and distributional information. Our results show that it outperforms Clark's algorithm for English and Bengali, suggesting that it is applicable to both morphologically impoverished and highly inflectional languages.

## BIBLIOGRAPHY

- Samit Bhattacharya, Monojit Choudhury, Sudeshna Sarkar and Anupam Basu. 2005. Inflectional morphology synthesis for Bengali noun, pronoun and verb systems. In *Proceedings of the National Conference on Computer Processing of Bangla (NCCPB 05)*, pp. 34 - 43.
- M. Banko and R. Moore. 2004. Part-of-speech tagging in context. In *Proceedings of COLING-2004*, Geneva, pp. 556-561.
- R. H. Baayen, R. Piepenbrock and L. Gulikers. 1996. The CELEX2 lexical database (CD-ROM), LDC, Univ of Pennsylvania, Philadelphia, PA.
- D. Bernhard. 2006. Unsupervised morphological segmentation based on segment predictability and word segment alignment. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.
- Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*.
- Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24(2):123-140.
- M. R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. In *Machine Learning*, 34, pp. 71-106.
- M. R. Brent, S. K. Murthy and A. Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*.
- Damir Cavar, Paul Rodriguez, and Giancarlo Schrementi. 2006. Unsupervised morphology induction for part-of-speech-tagging. In *Penn Working Papers in Linguistics: Proceedings of the 29th Annual Penn Linguistics Colloquium*. Vol. 12.1.
- Bidyut Baran Chaudhuri, Niladri. Sekhar. Dash and P. K. Kundu. 1997. Computer parsing of Bangla verbs. *Linguistics Today*, Vol. 1, No.1, pp. 64-86.
- Alexander Clark. 2000. Inducing syntactic categories by context distributional clustering. In *Proceedings of CoNLL*, pp. 91-94.
- Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *Proceedings of the European Chapter of the Association of Computational Linguistics (EACL)*.

- Mathias Creutz. 2003. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL)*, pp. 280-287.
- Mathias Creutz and Krista Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. In *Computer and Information Science, Report A81*, Helsinki University of Technology.
- Silviu Cucerzan and David Yarowsky. 2000. Language independent, minimally supervised induction of lexical probabilities. In *Proceedings of the Association of Computational Linguistics (ACL)*, pp. 270-277.
- Sajib Dasgupta and Mumit Khan. 2004. Feature unification for morphological parsing in Bangla. In *Proceedings of International Conference on Computer and Information Technology (ICCIT)*.
- Sajib Dasgupta and Vincent Ng. 2007a. High-performance, language-independent morphological segmentation. In *Proceedings of North American Chapter of the Association of the Computational Linguistics (NAACL-HLT)*.
- Sajib Dasgupta and Vincent Ng. 2007b. Unsupervised part-of-speech acquisition for resources-scarce languages. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-CoNLL)*.
- Niladri Sekhar Dash. 2006. The Morphodynamics of Bengali Compounds decomposing them for lexical processing. In *Language in India* ([www.languageinindia.com](http://www.languageinindia.com)), Vol 6:7.
- Kevin Duh and Katrin Kirchhoff. 2006. Lexicon acquisition for dialectal Arabic using transductive learning. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pp. 399-407.
- H. DéJean. 1998. Morphemes as necessary concepts for structures: Discovery from untagged corpora. In *Workshop on Paradigms and Grounding in Natural Language Learning*, pp. 295-299.
- Dayne Freitag. 2004. Toward unsupervised whole-corpus tagging. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pp. 357-363.
- Dayne Freitag. 2005. Morphology induction from term clusters. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL)*, pp. 128-135.
- John Goldsmith. 1997. Unsupervised learning of the morphology of a natural language. *University of Chicago*. <http://humanities.uchicago.edu/faculty/goldsmith>.
- John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. In *Computational Linguistics* 27(2), pp. 153-198.

- M. A. Hafer and S. F. Wess. 1974. Word segmentation by letter successor varities. *Information Storage and Retrieval*, 10: 371-385.
- Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the North American Chapter of the Association of Computational Linguistics (NAACL-HLT) HLT-NAACL*, pp. 320-327.
- Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pp. 94-101.
- Zellig Harris. 1954. Distributional structure. *Word*, 10(2/3):146-162.
- Z. Harris. 1955. From Phoneme to Morpheme. *Language*, 31(2): 190-222.
- Michele Jardino and Gilles Adda. 1994. Automatic determination of a stochastic bi-gram class language model. In *Proceedings of Grammatical Inference and Applications, Second International Colloquium, ICGI-94*, pp. 57-65. Springer-Verlag.
- Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*, pp. 44-56. MIT Press.
- S. Keshava and E. Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.
- K. Koskenniemi. 1983. Two-level morphology: a general computational model for word-form recognition and production. *Publication No. 11. Helsinki: University of Helsinki Department of General Linguistics*.
- J. Kupiec. 1992. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, Vol 6, pp. 225-242.
- Sydney Lamb. 1961. On the mechanization of syntactic analysis. In *Proceedings of the 1961 Conference on Machine Translation of Languages and Applied Language Analysis*, Volume 2, pp. 674-685. HMSO, London.
- B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, vol. 20, pp. 155-171.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313-330.

- Andrei Mikheev. 1997. Automatic rule induction for unknown word-guessing. *Computational Linguistics*, 23(3):405-423.
- Noah Smith and Jason Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proceedings of the Association of Computational Linguistics (ACL)*, pp. 354-362.
- Noah Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Association of Computational Linguistics (ACL)*, pp. 354-362.
- Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the Second Meeting of the North American Chapter of Association of Computational Linguistics (NAACL)*, pp. 183-191.
- Goutam Kumar Saha, Amiya Baran Saha, and Sudipto Debnath. 2004. Computer assisted Bangla words POS tagging. In *Proceedings of the International Symposium on Machine Translation NLP and TSS (iTRANS, 2004)*.
- Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the European Chapter of Association of Computational Linguistics (EACL)*, pp. 141-148.
- Hinrich Schütze. 1997. *Ambiguity Resolution in Language Learning*. CSLI Publications.
- M. G. Snover and M. R. Brent. 2001. A Bayesian model for morpheme and paradigm identification. In *Proceedings of the 39th Annual Meeting of the Association of Computational Linguistics (ACL)*, pp. 482-490.
- I. Wang and D. Schuurmans. 2005. Improved estimation for unsupervised part-of-speech tagging. In *Proceedings of the IEEE International Conference on Natural Language Processing and Knowledge Engineering*.
- David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS-taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the 2nd Meeting of North American Chapter of the Association of Computational Linguistics (NAACL)*, pp. 200-207.
- David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*, pp. 207-216.

## VITA

Sajib Dasgupta obtained his M.S. in Computer Science in 2007 from the University of Texas at Dallas (UTD) and B.Sc. in Computer Science and Engineering in 2004 from Bangladesh University of Engineering and Technology (BUET). Sajib's areas of interest in Computer Science are natural language processing, machine learning, computational linguistics and information retrieval. From 2005-2007, he worked as a research assistant under Dr. Vincent Ng. in the Human Language Technology Research Institute (HLTRI) at the University of Texas at Dallas, where his main task was to learn linguistic knowledge sources from raw text documents using machine learning techniques. Initially, he researched on movie review classification and then, as part of his masters thesis, he worked on building a language-independent learning system, which can automatically segment the words into prefixes, suffixes and roots and can predict the part-of-speech of a word without using any language-specific grammatical knowledge. He tested his system on several languages including English, Finnish, Turkish and Bengali. Prior to moving to UTD, he worked a research programmer from 2004 to 2005 in the Center for Research on Bangla Language Processing (CRBLP), BRAC University, Bangladesh, where he built smart word processing tools for Bengali. He also served as a Lecturer in BRAC University, Bangladesh from 2004 to 2005.

Sajib has published nine research papers in different conferences and journals, including the Annual Meeting of the Association of Computational Linguistics (ACL), the Conference of the North American Chapter of Association of Computational Linguistics

(NAACL) and the Conference on Empirical Methods in Natural Language Processing (EMNLP), the most distinguished conferences in the area of natural language processing.

Sajib has a great academic record too. He received his undergraduate with honors from Bangladesh University of Engineering and Technology (BUET), which is arguably the best university in Bangladesh. He was selected for the Dean's list for each academic year of his undergraduate study in BUET. Sajib also secured the second position out of 150,000 students in the higher secondary school certificate examination in 1997, and earned the Talent Pool Scholarship (the highest ranked scholarship given by the education board) for four academic years.