Proceedings of the Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL), Athens, Greece, March 2009, pp. 354-362.

Learning-Based Named Entity Recognition for Morphologically-Rich, Resource-Scarce Languages

Kazi Saidul Hasan and **Md. Altaf ur Rahman** and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{saidul,altaf,vince}@hlt.utdallas.edu

Abstract

Named entity recognition for morphologically rich, case-insensitive languages, including the majority of semitic languages, Iranian languages, and Indian languages, is inherently more difficult than its English counterpart. Worse still, progress on machine learning approaches to named entity recognition for many of these languages is currently hampered by the scarcity of annotated data and the lack of an accurate part-of-speech tagger. While it is possible to rely on manually-constructed gazetteers to combat data scarcity, this gazetteer-centric approach has the potential weakness of creating irreproducible results, since these name lists are not publicly available in general. Motivated in part by this concern, we present a learning-based named entity recognizer that does not rely on manually-constructed gazetteers, using Bengali as our representative resource-scarce, morphologically-rich language. Our recognizer achieves a relative improvement of 7.5% in F-measure over a baseline recognizer. Improvements arise from (1) using induced affixes, (2) extracting information from online lexical databases, and (3) jointly modeling part-of-speech tagging and named entity recognition.

1 Introduction

While research in natural language processing has gained a lot of momentum in the past several decades, much of this research effort has been focusing on only a handful of politically-important languages such as English, Chinese, and Arabic. On the other hand, being the fifth most spoken language[1] with more than 200 million native speakers residing mostly in Bangladesh and the Indian state of West Bengal, Bengali has far less electronic resources than the aforementioned languages. In fact, a major obstacle to the automatic processing of Bengali is the scarcity of annotated corpora.

One potential solution to the problem of data scarcity is to hand-annotate a small amount of data with the desired linguistic information and then develop bootstrapping algorithms for combining this small amount of labeled data with a large amount of unlabeled data. In fact, co-training (Blum and Mitchell, 1998) has been successfully applied to English named entity recognition (NER) (Collins & Singer [henceforth C&S] (1999)). In C&S's approach, consecutive words tagged as proper nouns are first identified as potential NEs, and each such NE is then labeled by combining the outputs of two co-trained classifiers. Unfortunately, there are practical difficulties in applying this technique to Bengali NER. First, one of C&S's co-trained classifiers uses features based on capitalization, but Bengali is case-insensitive. Second, C&S identify potential NEs based on proper nouns, but unlike English, (1) proper noun identification for Bengali is non-trivial, due to the lack of capitalization; and (2) there does not exist an accurate Bengali part-of-speech (POS) tagger for providing such information, owing to the scarcity of annotated data for training the tagger.

In other words, Bengali NER is complicated not only by the scarcity of annotated data, but also by the lack of an accurate POS tagger. One could imagine building a Bengali POS tagger using un-

[1] See http://en.wikipedia.org/wiki/Bengali_language.

supervised induction techniques that have been successfully developed for English (e.g., Schütze (1995), Clark (2003)), including the recently-proposed prototype-driven approach (Haghighi and Klein, 2006) and Bayesian approach (Goldwater and Griffiths, 2007). The majority of these approaches operate by clustering distributionally similar words, but they are unlikely to work well for Bengali for two reasons. First, Bengali is a relatively free word order language, and hence the distributional information collected for Bengali words may not be as reliable as that for English words. Second, many closed-class words that typically appear in the distributional representation of an English word (e.g., prepositions and particles such as "in" and "to") are realized as inflections in Bengali, and the absence of these informative words implies that the context vector may no longer capture sufficient information for accurately clustering the Bengali words.

In view of the above problems, many learning-based Bengali NE recognizers have relied heavily on manually-constructed name lists for identifying persons, organizations, and locations. There are at least two weaknesses associated with this gazetteer-centric approach. First, these name lists are typically not publicly available, making it difficult to reproduce the results of these NE recognizers. Second, it is not clear how comprehensive these lists are. Relying on comprehensive lists that comprise a large portion of the names in the test set essentially reduces the NER problem to a dictionary-lookup problem, which is arguably not very interesting from a research perspective.

In addition, many existing learning-based Bengali NE recognizers have several common weaknesses. First, they use as features *pseudo-affixes*, which are created by extracting the first n and the last n characters of a word (where $1 \leq n \leq 4$) (e.g., Dandapat et al. (2007)). While affixes encode essential grammatical information in Bengali due to its morphological richness, this extraction method is arguably too ad-hoc and does not cover many useful affixes. Second, they typically adopt a *pipelined* NER architecture, performing POS tagging prior to NER and encoding the resulting not-so-accurate POS information as a feature. In other words, errors in POS tagging are propagated to the NE recognizer via the POS feature, thus limiting its performance.

Motivated in part by these weaknesses, we investigate how to improve a learning-based NE recognizer that does *not* rely on manually-constructed gazetteers. Specifically, we investigate two learning architectures for our NER system. The first one is the aforementioned pipelined architecture in which the NE recognizer uses as features the output of a POS tagger that is trained independently of the recognizer. Unlike existing Bengali POS and NE taggers, however, we examine two new knowledge sources for training these taggers: (1) affixes induced from an unannotated corpus and (2) semantic class information extracted from Wikipedia. In the second architecture, we *jointly* learn the POS tagging and the NER tasks, allowing features for one task to be accessible to the other task during learning. The goal is to examine whether any benefits can be obtained via joint modeling, which could address the error propagation problem with the pipelined architecture.

While we focus on Bengali NER in this paper, none of the proposed techniques are language-specific. In fact, we believe that these techniques are of relevance and interest to the EACL community because they can be equally applicable to the numerous resource-scarce European and Middle Eastern languages that share similar linguistic and extra-linguistic properties as Bengali. For instance, the majority of semitic languages and Iranian languages are, like Bengali, morphologically productive; and many East European languages such as Czech and Polish resemble Bengali in terms of not only their morphological richness, but also their relatively free word order.

The rest of the paper is organized as follows. In Section 2, we briefly describe the related work. Sections 3 and 4 show how we induce affixes from an unannotated corpus and extract semantic class information from Wikipedia. In Sections 5 and 6, we train and evaluate a POS tagger and an NE recognizer independently, augmenting the feature set typically used for these two tasks with our new knowledge sources. Finally, we describe and evaluate our joint model in Section 7.

2 Related Work

Cucerzan and Yarowsky (1999) exploit morphological and contextual patterns to propose a language-independent solution to NER. They use affixes based on the paradigm that named entities corresponding to a particular class have similar morphological structure. Their bootstrapping

approach is tested on Romanian, English, Greek, Turkish, and Hindi. The recall for Hindi is the lowest (27.84%) among the five languages, suggesting that the lack of case information can significantly complicate the NER task.

To investigate the role of gazetteers in NER, Mikheev et al. (1999) combine grammar rules with maximum entropy models and vary the gazetteer size. Experimental results show that (1) the F-scores for NE classes like person and organization are still high without gazetteers, ranging from 85% to 92%; and (2) a small list of country names can improve the low F-score for locations substantially. It is worth noting that their recognizer requires that the input data contain POS tags and simple semantic tags, whereas ours automatically acquires such linguistic information. In addition, their approach uses part of the dataset to extend the gazetteer. Therefore, the resulting gazetteer list is specific to a particular domain; on the other hand, our approach does not generate a domain-specific list, since it makes use of Wikipedia articles.

Kozareva (2006) generates gazetteer lists for person and location names from unlabeled data using common patterns and a graph exploration algorithm. The location pattern is essentially a preposition followed by capitalized context words. However, this approach is inadequate for a morphologically-rich language like Bengali, since prepositions are often realized as inflections.

3 Affix Induction

Since Bengali is morphologically productive, a lot of grammatical information about Bengali words is expressed via affixes. Hence, these affixes could serve as useful features for training POS and NE taggers. In this section, we show how to induce affixes from an unannotated corpus.

We rely on a simple idea proposed by Keshava and Pitler (2006) for inducing affixes. Assume that (1) V is a vocabulary (i.e., a set of distinct words) extracted from a large, unannotated corpus, (2) α and β are two character sequences, and (3) $\alpha\beta$ is the concatenation of α and β. If $\alpha\beta$ and α are found in V, we extract β as a suffix. Similarly, if $\alpha\beta$ and β are found in V, we extract α as a prefix.

In principle, we can use all of the induced affixes as features for training a POS tagger and an NE recognizer. However, we choose to use only those features that survive our feature selection process (to be described below), for the following reasons. First, the number of induced affixes is large, and using only a subset of them as features could make the training process more efficient. Second, the above affix induction method is arguably overly simplistic and hence many of the induced affixes could be spurious.

Our feature selection process is fairly simple: we (1) score each affix by multiplying its *frequency* (i.e., the number of distinct words in V to which each affix attaches) and its *length*[2], and (2) select only those whose score is above a certain threshold. In our experiments, we set this threshold to 50, and generate our vocabulary of 140K words from five years of articles taken from the Bengali newspaper *Prothom Alo*. This enables us to induce 979 prefixes and 975 suffixes.

4 Semantic Class Induction from Wikipedia

Wikipedia has recently been used as a knowledge source for various language processing tasks, including taxonomy construction (Ponzetto and Strube, 2007a), coreference resolution (Ponzetto and Strube, 2007b), and English NER (e.g., Bunescu and Paşca (2006), Cucerzan (2007), Kazama and Torisawa (2007), Watanabe et al. (2007)). Unlike previous work on using Wikipedia for NER, our goal here is to (1) generate a list of phrases and tokens that are potentially named entities from the 16914 articles in the Bengali Wikipedia[3] and (2) heuristically annotate each of them with one of four classes, namely, PER (person), ORG (organization), LOC (location), or OTHERS (i.e., anything other than PER, ORG and LOC).

4.1 Generating an Annotated List of Phrases

We employ the steps below to generate our annotated list.

Generating and annotating the titles Recall that each Wikipedia article has been optionally assigned to one or more categories by its creator and/or editors. We use these categories to help annotate the title of an article. Specifically, if an article has a category whose name starts with "Born on" or "Death on," we label the corresponding title with PER. Similarly, if it has a category whose name starts with "Cities of" or "Countries of," we

[2]The dependence on frequency and length is motivated by the observation that less frequent and shorter affixes are more likely to be erroneous (see Goldsmith (2001)).

[3]See http://bn.wikipedia.org. In our experiments, we used the Bengali Wikipedia dump obtained on October 22, 2007.

NE Class	Keywords
PER	"born," "died," "one," "famous"
LOC	"city," "area," "population," "located," "part of"
ORG	"establish," "situate," "publish"

Table 1: Keywords for each named entity class

label the title as LOC. If an article does not belong to one of the four categories above, we label its title with the help of a small set of seed keywords shown in Table 1. Specifically, for each of the three NE classes shown on the left of Table 1, we compute a weighted sum of its keywords: a keyword that appears in the first paragraph has a weight of 3, a keyword that appears elsewhere in the article has a weight of 1, and a keyword that does not appear in the article has a weight of 0. The rationale behind using different weights is simple: the first paragraph is typically a brief exposition of the title, so it should in principle contain words that correlate more closely with the title than words appearing in the rest of the article. We then label the title with the class that has the largest weighted sum. Note, however, that we ignore any article that contains fewer than two keywords, since we do not have reliable evidence for labeling its title as one of the NE classes. We put all these annotated titles into a *title list*.

Getting more location names To get more location names, we search for the character sequences "birth place:" and "death place:" in each article, extracting the phrase following any of these sequences and label it as LOC. We put all such labeled locations into the title list.

Generating and annotating the tokens in the titles Next, we extract the word tokens from each title in the title list and label each token with an NE class. The reason for doing this is to improve generalization: if "Dhaka University" is labeled as ORG in the title list, then it is desirable to also label the token "University" as ORG, because this could help identify an unseen phrase that contains the term "University" as an organization. Our token labeling method is fairly simple. First, we generate the tokens from each title in the title list, assigning to each token the same NE label as that of the title from which it is generated. For instance, from the title "Anna Frank," "Anna" will be labeled as PER; and from "Anna University," " Anna" will be labeled as LOC. To resolve such ambiguities (i.e., assigning different labels to the same token), we keep a count of how many times "Anna" is labeled with each NE class, and set its final label to be the most frequent NE class. We put all these annotated tokens into a *token list*. If the title list and the token list have an element in common, we remove the element from the token list, since we have a higher confidence in the labels of the titles.

Merging the lists Finally, we append the token list to the title list. The resulting title list contains 4885 PERs, 15176 LOCs, and 188 ORGs.

4.2 Applying the Annotated List to a Text

We can now use the title list to annotate a text. Specifically, we process each word w in the text in a left-to-right manner, using the following steps:

1. Check whether w has been labeled. If so, we skip this word and process the next one.
2. Check whether w appears in the Samsad Bengali-English Dictionary[4]. If so, we assume that w is more likely to be used as a non-named entity, thus leaving the word unlabeled and processing the next word instead.
3. Find the longest unlabeled word sequence[5] that begins with w and appears in the title list. If no such sequence exists, we leave w unlabeled and process the next word. Otherwise, we label it with the NE tag given by the title list. To exemplify, consider a text that starts with the sentence "Smith College is in Massachusetts." When processing "Smith," "Smith College" is the longest sequence that starts with "Smith" and appears in the title list (as an ORG). As a result, we label *all* occurrences of "Smith College" in the text as an ORG. (Note that without using the longest match heuristic, "Smith" would likely be mislabeled as PER.) In addition, we take the last word of the ORG (which in this case is "College") and annotate each of its occurrence in the rest of the text as ORG.[6]

These automatic annotations will then be used to derive a set of WIKI features for training our POS tagger and NE recognizer. Hence, unlike existing Bengali NE recognizers, our "gazetteers" are induced rather than manually created.

[4] See http://dsal.uchicago.edu/dictionaries/biswasbengali/.

[5] This is a sequence in which each word is unlabeled.

[6] However, if we have a PER match (e.g., "Anna Frank") or a LOC match (e.g., "Las Vegas"), we take *each* word in the matched phrase and label each of its occurrence in the rest of the text with the same NE tag.

Current word	w_i
Previous word	w_{i-1}
2nd previous word	w_{i-2}
Next word	w_{i+1}
2nd next word	w_{i+2}
Current pseudo-affixes	pf_i (prefix), sf_i (suffix)
Current induced affixes	pi_i (prefix), si_i (suffix)
Previous induced affixes	pi_{i-1} (prefix), si_{i-1} (suffix)
Induced affix bigrams	$pi_{i-1}pi_i$ (prefix), $si_{i-1}si_i$ (suffix)
Current Wiki tag	$wiki_i$
Previous Wiki tag	$wiki_{i-1}$
Wiki bigram	$wiki_{i-1}wiki_i$
Word bigrams	$w_{i-2}w_{i-1}$, $w_{i-1}w_i$, w_iw_{i+1}, $w_{i+1}w_{i+2}$
Word trigrams	$w_{i-2}w_{i-1}w_i$
Current number	q_i

Table 2: Feature templates for the POS tagging experiments

5 Part-of-Speech Tagging

In this section, we will show how we train and evaluate our POS tagger. As mentioned before, we hypothesize that introducing our two knowledge sources into the feature set for the tagger could improve its performance: using the induced affixes could improve the extraction of grammatical information from the words, and using the Wikipedia-induced list, which in principle should comprise mostly of names, could help improve the identification of proper nouns.

Corpus Our corpus is composed of 77942 words and is annotated with one of 26 POS tags in the tagset defined by IIIT Hyderabad[7]. Using this corpus, we perform 5-fold cross-validation (CV) experiments in our evaluation. It is worth noting that this dataset has a high unknown word rate of 15% (averaged over the five folds), which is due to the small size of the dataset. While this rate is comparable to another Bengali POS dataset described in Dandapat et al. (2007), it is much higher than the 2.6% unknown word rate in the test set for Ratnaparkhi's (1996) English POS tagging experiments.

Creating training instances Following previous work on POS tagging, we create one training instance for each word in the training set. The class value of an instance is the POS tag of the corresponding word. Each instance is represented by a set of linguistic features, as described next.

[7]A detailed description of these POS tags can be found in http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf, and are omitted here due to space limitations. This tagset and the Penn Treebank tagset differ in that (1) nouns do not have a number feature; (2) verbs do not have a tense feature; and (3) adjectives and adverbs are not subcategorized.

Features Our feature set consists of (1) *baseline* features motivated by those used in Dandapat *et al.*'s (2007) Bengali POS tagger and Singh *et al.*'s (2006) Hindi POS tagger, as well as (2) features derived from our induced affixes and the Wikipedia-induced list. More specifically, the baseline feature set has (1) word unigrams, bigrams and trigrams; (2) pseudo-affix features that are created by taking the first three characters and the last three characters of the current word; and (3) a binary feature that determines whether the current word is a number. As far as our new features are concerned, we create one induced prefix feature and one induced suffix feature from both the current word and the previous word, as well as two bigrams involving induced prefixes and induced suffixes. We also create three WIKI features, including the Wikipedia-induced NE tag of the current word and that of the previous word, as well as the combination of these two tags. Note that the Wikipedia-induced tag of a word can be obtained by annotating the test sentence under consideration using the list generated from the Bengali Wikipedia (see Section 4). To make the description of these features more concrete, we show the feature templates in Table 2.

Learning algorithm We used CRF++[8], a C++ implementation of conditional random fields (Lafferty et al., 2001), as our learning algorithm for training a POS tagging model.

Evaluating the model To evaluate the resulting POS tagger, we generate test instances in the same way as the training instances. 5-fold CV results of the POS tagger are shown in Table 3. Each row consists of three numbers: the overall accuracy, as well as the accuracies on the seen and the unseen words. Row 1 shows the accuracy when the baseline feature set is used; row 2 shows the accuracy when the baseline feature set is augmented with our two induced affix features; and the last row shows the results when both the induced affix and the WIKI features are incorporated into the baseline feature set. Perhaps not surprisingly, (1) adding more features improves performance, and (2) accuracies on the seen words are substantially better than those on the unseen words. In fact, adding the induced affixes to the baseline feature set yields a 7.8% reduction in relative error in overall accuracy. We also applied a two-tailed paired t-test ($p < 0.01$), first to the overall accura-

[8]Available from http://crfpp.sourceforge.net

Experiment	Overall	Seen	Unseen
Baseline	89.83	92.96	72.08
Baseline+Induced Affixes	90.57	93.39	74.64
Baseline+Induced Affixes+Wiki	**90.80**	93.50	75.58

Table 3: 5-fold cross-validation accuracies for POS tagging

Predicted Tag	Correct Tag	% of Error
NN	NNP	22.7
NN	JJ	9.6
JJ	NN	7.4
NNP	NN	5.0
NN	VM	4.9

Table 4: Most frequent errors for POS tagging

cies in rows 1 and 2, and then to the overall accuracies in rows 2 and 3. Both pairs of numbers are statistically significantly different from each other, meaning that incorporating the two induced affix features and then the WIKI features both yields significant improvements.

Error analysis To better understand the results, we examined the errors made by the tagger. The most frequent errors are shown in Table 4. From the table, we see that the largest source of errors arises from mislabeling proper nouns as common nouns. This should be expected, as proper noun identification is difficult due to the lack of capitalization information. Unfortunately, failure to identify proper nouns could severely limit the recall of an NE recognizer. Also, adjectives and common nouns are difficult to distinguish, since these two syntactic categories are morphologically and distributionally similar to each other. Finally, many errors appear to involve mislabeling a word as a common noun. The reason is that there is a larger percentage of common nouns (almost 30%) in the training set than other POS tags, thus causing the model to prefer tagging a word as a common noun.

6 Named Entity Recognition

In this section, we show how to train and evaluate our NE recognizer. The recognizer adopts a traditional architecture, assuming that POS tagging is performed prior to NER. In other words, the NE recognizer will use the POS acquired in Section 5 as one of its features. As in Section 5, we will focus on examining how our knowledge sources (the induced affixes and the WIKI features) impact the performance of our recognizer.

Corpus The corpus we used for NER evaluation is the same as the one described in the previous section. Specifically, in addition to POS information, each sentence in the corpus is annotated with NE information. We focus on recognizing the three major NE types in this paper, namely persons (PER), organizations (ORG), and locations (LOC). There are 1721 PERs, 104 ORGs, and 686 LOCs in the corpus. As far as evaluation is concerned, we conduct 5-fold CV experiments, dividing the corpus into the same five folds as in POS tagging.

POS of current word	t_i
POS of previous word	t_{i-1}
POS of 2nd previous word	t_{i-2}
POS of next word	t_{i+1}
POS of 2nd next word	t_{i+2}
POS bigrams	$t_{i-2}t_{i-1}, t_{i-1}t_i, t_it_{i+1}, t_{i+1}t_{i+2}$
First word	fw_i

Table 5: Additional feature templates for the NER experiments

Creating training instances We view NE recognition as a sequence labeling problem. In other words, we combine NE identification and classification into one step, labeling each word in a test text with its NE tag. Any word that does not belong to one of our three NE tags will be labeled as OTHERS. We adopt the IOB convention, preceding an NE tag with a B if the word is the first word of an NE and an I otherwise. Now, to train the NE recognizer, we create one training instance from each word in a training text. The class value of an instance is the NE tag of the corresponding word, or OTHERS if the word is not part of an NE. Each instance is represented by a set of linguistic features, as described next.

Features Our feature set consists of (1) *baseline* features motivated by those used in Ekbal *et al.*'s (2008) Bengali NE recognizer, as well as (2) features derived from our induced affixes and the Wikipedia-induced list. More specifically, the baseline feature set has (1) word unigrams; (2) pseudo-affix features that are created by taking the first three characters and the last three characters of the current word; (3) a binary feature that determines whether the current word is the first word of a sentence; and (4) a set of POS-related features, including the POS of the current word and its surrounding words, as well as POS bigrams formed from the current and surrounding words. Our induced affixes and WIKI features are incorporated into the baseline NE feature set in the same manner as in POS tagging. In essence, the feature tem-

Experiment	R	P	F
Baseline	60.97	74.46	67.05
Person	66.18	74.06	69.90
Organization	29.81	44.93	35.84
Location	52.62	80.40	63.61
Baseline+Induced Affixes	60.45	73.30	66.26
Person	65.70	72.61	69.02
Organization	31.73	46.48	37.71
Location	51.46	80.05	62.64
Baseline+Induced Affixes+Wiki	63.24	75.19	**68.70**
Person	66.47	75.16	70.55
Organization	30.77	43.84	36.16
Location	60.06	79.69	68.50

Table 6: 5-fold cross-validation results for NER

plates employed by the NE recognizer are the top 12 templates in Table 2 and those in Table 5.

Learning algorithm We again use CRF++ as our sequence learner for acquiring the recognizer.

Evaluating the model To evaluate the resulting NE tagger, we generate test instances in the same way as the training instances. To score the output of the recognizer, we use the CoNLL-2000 scoring program[9], which reports performance in terms of recall (R), precision (P), and F-measure (F). All NE results shown in Table 6 are averages of the 5-fold CV experiments. The first block of the Table 6 shows the overall results when the baseline feature set is used; in addition, we also show results for each of the three NE tags. As we can see, the baseline achieves an F-measure of 67.05. The second block shows the results when the baseline feature set is augmented with our two induced affix features. Somewhat unexpectedly, F-measure drops by 0.8% in comparison to the baseline. Additional experiments are needed to determine the reason. Finally, when the WIKI features are incorporated into the augmented feature set, the system achieves an F-measure of 68.70 (see the third block), representing a statistically significant increase of 1.6% in F-measure over the baseline. As we can see, improvements stem primarily from dramatic gains in recall for locations.

Discussions Several points deserve mentioning. First, the model performs poorly on the ORGs, owing to the small number of organization names in the corpus. Worse still, the recall drops after adding the WIKI features. We examined the list of induced ORG names and found that it is fairly noisy. This can be attributed in part to the difficulty in forming a set of seed words that can extract ORGs with high precision (e.g., the ORG seed "situate" extracted many LOCs). Second, using the WIKI features does not help recalling the PERs. A closer examination of the corpus reveals the reason: many sentences describe fictitious characters, whereas Wikipedia would be most useful for articles that describe famous people. Overall, while the WIKI features provide our recognizer with a small, but significant, improvement, the usefulness of the Bengali Wikipedia is currently limited by its small size. Nevertheless, we believe the Bengali Wikipedia will become a useful resource for language processing as its size increases.

[9]http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

7 A Joint Model for POS Tagging and NER

The NE recognizer described thus far has adopted a pipelined architecture, and hence its performance could be limited by the errors of the POS tagger. In fact, as discussed before, the major source of errors made by our POS tagger concerns the confusion between proper nouns and common nouns, and this type of error, when propagated to the NE recognizer, could severely limit its recall. Also, there is strong empirical support for this argument: the NE recognizers, when given access to the correct POS tags, have F-scores ranging from 76-79%, which are 10% higher on average than those with POS tags that were automatically computed. Consequently, we hypothesize that modeling POS tagging and NER jointly would yield better performance than learning the two tasks separately. In fact, many approaches have been developed to jointly model POS tagging and noun phrase chunking, including transformation-based learning (Ngai and Florian, 2001), factorial HMMs (Duh, 2005), and dynamic CRFs (Sutton et al., 2007). Some of these approaches are fairly sophisticated and also require intensive computations during inference. For instance, when jointly modeling POS tagging and chunking, Sutton et al. (2007) reduce the number of POS tags from 45 to 5 when training a factorial dynamic CRF on a small dataset (with only 209 sentences) in order to reduce training and inference time.

In contrast, we propose a relatively simple model for jointly learning Bengali POS tagging and NER, by exploiting the limited dependencies between the two tasks. Specifically, we make the observation that most of the Bengali words that are part of an NE are also proper nouns. In fact, based on statistics collected from our evaluation corpus (see Sections 5 and 6), this observation is correct

Experiment	R	P	F
Baseline	54.76	81.70	65.57
Baseline+Induced Affixes	56.79	88.96	69.32
Baseline+Induced Affixes+Wiki	61.73	86.35	**71.99**

Table 7: 5-fold cross-validation joint modeling results for NER

97.3% of the time. Note, however, that this observation does not hold for English, since many prepositions and determiners are part of an NE. On the other hand, this observation largely holds for Bengali because prepositions and determiners are typically realized as noun suffixes.

This limited dependency between the POS tags and the NE tags allows us to develop a simple model for jointly learning the two tasks. More specifically, we will use CRF++ to learn the joint model. Training and test instances are generated as described in the previous two subsections (i.e., one instance per word). The feature set will consist of the union of the features that were used to train the POS tagger and the NE tagger independently, minus the POS-related features that were used in the NE tagger. The class value of an instance is computed as follows. If a word is not a proper noun, its class is simply its POS tag. Otherwise, its class is its NE tag, which can be PER, ORG, LOC, or OTHERS. In other words, our joint model exploits the observation that we made earlier in the section by assuming that only proper nouns can be part of a named entity. This allows us to train a joint model without substantially increasing the number of classes.

We again evaluate our joint model using 5-fold CV experiments. The NE results of the model are shown in Table 7. The rows here can be interpreted in the same manner as those in Table 6. Comparing these three experiments with their counterparts in Table 6, we can see that, except for the baseline, jointly modeling offers a significant improvement of 3.3% in overall F-measure.[10] In particular, the joint model benefits significantly from our two knowledge sources, achieving an F-measure of 71.99% when both of them are incorporated.

Finally, to better understand the value of the induced affix features in the joint model as well as the pipelined model described in Section 6, we conducted an ablation experiment, in which we incorporated only the WIKI features into the baseline feature set. With pipelined modeling, the F-measure for NER is 68.87%, which is similar to the case where both induced affixes and the WIKI features are used. With joint modeling, however, the F-measure for NER is 70.87%, which is 1% lower than the best joint modeling score. These results provide suggestive evidence that the induced affix features play a significant role in the improved performance of the joint model.

8 Conclusions

We have explored two types of linguistic features, namely the induced affix features and the Wikipedia-related features, to improve a Bengali POS tagger and NE recognizer. Our experimental results have demonstrated that (1) both types of features significantly improve a baseline POS tagger and (2) the Wikipedia-related features significantly improve a baseline NE recognizer. Moreover, by exploiting the limited dependencies between Bengali POS tags and NE tags, we proposed a new model for jointly learning the two tasks, which not only avoids the error-propagation problem present in the pipelined system architecture, but also yields statistically significant improvements over the NE recognizer that is trained independently of the POS tagger. When applied in combination, our three extensions contributed to a relative improvement of 7.5% in F-measure over the baseline NE recognizer. Most importantly, we believe that these extensions are of relevance and interest to the EACL community because many European and Middle Eastern languages resemble Bengali in terms of not only their morphological richness but also their scarcity of annotated corpora. We plan to empirically verify our belief in future work.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on the paper. We also thank CRBLP, BRAC University, Bangladesh, for providing us with Bengali resources. This work was supported in part by NSF Grant IIS-0812261.

[10]The POS tagging results are not shown due to space limitations. Overall, the POS accuracies drop insignificantly as a result of joint modeling, for the following reason. Recall from Section 5 that the major source of POS tagging errors arises from the mislabeling of many proper nouns as common nouns, due primarily to the large number of common nouns in the corpus. The joint model aggravates this problem by subcategorizing the proper nouns into different NE classes, causing the tagger to have an even stronger bias towards labeling a proper noun as a common noun than before. Nevertheless, as seen from the results in Tables 6 and 7, such a bias has yielded an increase in NER precision.

References

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*, pages 92–100.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of EACL*, pages 9–16.

Alexander Clark. 2003. Combining distributional and morphological information for part-of-speech induction. In *Proceedings of EACL*, pages 59–66.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP/VLC*, pages 100–110.

Silviu Cucerzan and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of EMNLP/VLC*, pages 90–99.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL*, pages 708–716.

Sandipan Dandapat, Sudeshna Sarkar, and Anupam Basu. 2007. Automatic part-of-speech tagging for Bengali: An approach for morphologically rich languages in a poor resource scenario. In *Proceedings of the ACL Companion Volume*, pages 221–224.

Kevin Duh. 2005. Jointly labeling multiple sequences: A factorial HMM approach. In *Proceedings of the ACL Student Research Workshop*, pages 19–24.

Asif Ekbal, Rejwanul Haque, and Sivaji Bandyopadhyay. 2008. Named entity recognition in Bengali: A conditional random field approach. In *Proceedings of IJCNLP*, pages 589–594.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*, pages 320–327.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of EMNLP-CoNLL*, pages 698–707.

Samarth Keshava and Emily Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

Zornitsa Kozareva. 2006. Bootstrapping named entity recognition with automatically generated gazetteer lists. In *Proceedings of the EACL Student Research Workshop*, pages 15–22.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of EACL*, pages 1–8.

Grace Ngai and Radu Florian. 2001. Transformation based learning in the fast lane. In *Proceedings of NAACL*, pages 40–47.

Simone Paolo Ponzetto and Michael Strube. 2007a. Deriving a large scale taxonomy from wikipedia. In *Proceedings of AAAI*, pages 1440–1445.

Simone Paolo Ponzetto and Michael Strube. 2007b. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of EACL*, pages 141–148.

Smriti Singh, Kuhoo Gupta, Manish Shrivastava, and Pushpak Bhattacharyya. 2006. Morphological richness offsets resource demand — Experiences in constructing a POS tagger for Hindi. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 779–786.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8:693–723.

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2007. A graph-based approach to named entity categorization in Wikipedia using conditional random fields. In *Proceedings of EMNLP-CoNLL*, pages 649–657.