Named Entity Recognition for Manipuri Using Support Vector Machine

Thoudam Doren Singh[a,b], Kishorjit Nongmeikapam[b], Asif Ekbal[c], and Sivaji Bandyopadhyay[b]

[a]Center for Development of Advanced Computing
Gulmohar Cross Road No 9, Juhu, Mumbai-400049, India
thoudamds@cdacmumbai.in
[b]Computer Science and Engineering Department
Jadavpur University, Kolkata-700032, India
[c]Department of Computational Linguistics,
University of Heidelberg,
Im Neuenheimer Feld 325, 69120 Heidelberg, Germany
{kishorjit.nongmeikapa, asif.ekbal}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. This paper reports about the development of a Manipuri NER system, a less computerized Indian language. Two different models, one using an active learning technique based on the context patterns generated from an unlabeled news corpus and the other based on the well known Support Vector Machine (SVM), have been developed. The active learning technique has been considered as the *baseline* system. The Manipuri news corpus has been manually annotated with the major NE tags, namely *Person name*, *Location name*, *Organization name* and *Miscellaneous name* to apply SVM. The SVM based system makes use of the different contextual information of the words along with the variety of orthographic word-level features which are helpful in predicting the NE classes. In addition, lexical context patterns generated using the active learning technique have been used as the features of SVM in order to improve performance. The system has been trained and tested with 28,629 and 4,763 wordforms, respectively. Experimental results show the effectiveness of the proposed approach with the overall average *Recall*, *Precision* and *F-Score* values of 93.91%, 95.32% and 94.59% respectively.

1 Introduction

Named Entity Recognition (NER) is an important tool for several Natural Language Processing (NLP) application areas. The objective of NER is to identify and classify every word/term in a document into some predefined categories. The challenge in detection of Named Entities (NE) is that such expressions are hard to analyze using rule-based NLP techniques because they belong to the open class of expressions, i.e., there is an infinite variety and new expressions are constantly being invented. The development of an automatic NER system requires either a comprehensive set of linguistically motivated rules or a large amount of annotated corpora in order to achieve reasonable performance. But such rules or corpora have been developed for a few languages like English, most of the European languages and some of the Asian languages like Chinese, Japanese and Korean. NER systems for Indian languages are not readily available due to the unavailability of such handcrafted rules or annotated corpora.

Rule based approaches focus on extracting names using a number of handcrafted rules. Generally, these systems consist of a set of patterns using grammatical (e.g., part of speech), syntactic (e.g., word precedence) and orthographic features (e.g., capitalization) in combination with dictionaries. One drawback of the rule-based approach is that it lacks the ability of coping with the problems of robustness and portability. Each new source of text requires significant tweaking of rules to maintain optimal performance and the maintenance costs could be quite high. While early studies are mostly based on handcrafted rules, most recent ones use machine learning (ML) models as a way to automatically induce rule based systems or sequence labeling algorithms starting from a collection of training examples. Some of the very effective ML approaches used in NER are HMM (Nymble in Bikel et al. (1999)), ME (Borthwick, 1999), CRFs (Lafferty et al., 2001) and SVM (Yamada et al., 2001). The SVM based NER system was proposed by Yamada et al. (2001) for Japanese.

Manipuri is a Tibeto-Burman, scheduled Indian language and highly agglutinative in behavior, monosyllabic, influenced and enriched by the Indo-Aryan languages of Sanskrit origin and English. The affixes play the most important role in the structure of the language. A clear-cut demarcation between morphology and syntax is not possible in this language. The majority of the roots found in the language are bound and the affixes are the determining factor for the class of the words in the language. Classification of words using the role of affix helps to implement the NE tagger for a resource poor language like Manipuri with high performance. There is no report of NER work prior to our experiment for Manipuri. NE identification in Indian languages as well as in Manipuri is difficult and challenging as:

1 Unlike English and most of the European languages, Manipuri lacks capitalization information, which plays a very important role in identifying NEs.
2 A lot of NEs in Manipuri can appear in the dictionary with some other specific meanings.
3 Manipuri is a highly inflectional language providing one of the richest and most challenging sets of linguistic and statistical features resulting in long and complex wordforms.
4 Manipuri is a relatively free word order language. Thus NEs can appear in subject and object positions making the NER task more difficult compared to others.
5 Manipuri is a resource-constrained language. Annotated corpus, name dictionaries, sophisticated morphological analyzers, POS taggers etc. are not yet available.

In Indian language context, a HMM based NER system for Bengali has been reported in Ekbal et al. (2007), where additional contextual information has been considered for emission probabilities and NE suffixes are used for handling the unknown words. Other works in Bengali NER can be found in Ekbal et al. (2008), and Ekbal and Bandyopadhyay (2008) with the CRF, and SVM approaches, respectively. Other than Bengali, the works on Hindi can be found in Li and McCallum (2004) with CRF. Various works on NER involving Indian languages are reported in IJCNLP-08 NER Shared Task on South and South East Asian Languages (NERSSEAL)[1] using various techniques.

In this paper, Manipuri NER systems have been developed using an active learning technique as well as SVM. We collected the data from http://www.thesangaiexpress.com/ , a popular Manipuri newspaper. Initially, a *baseline* system has been developed based on an active learning technique that generates lexical context patterns from the unlabeled corpus in a bootstrapping manner. This corpus has been manually annotated with the four major NE tags, namely *Person name*, *Location name*, *Organization name* and *Miscellaneous name*. Miscellaneous name includes the festival name, name of objects, name of building, date, time, measurement expression and percentage expression etc. The SVM based system makes use of the different contextual information of the words along with variety of orthographic word-level features that are helpful to identify the various NE classes. A number of experiments have been conducted to identify the best set of features for NER in Manipuri under the SVM framework.

2 Support Vector Machine

The SVM (Vapnik, 1995) is based on discriminative approach and makes use of both positive and negative examples to learn the distinction between the two classes. The SVMs are known to robustly handle large feature sets and to develop models that maximize their generalizability.

[1] http://ltrc.iiit.ac.in/ner-ssea-08/proc/index.html

Suppose we have a set of training data for a two-class problem: $\{(x_1, y_1), ...(x_N, y_N)\}$, where xi ε RD is a feature vector of the i th sample in the training data and yi ε {+1, -1} is the class to which xi belongs. The goal is to find a decision function that accurately predicts class y for an input vector x. A non-linear SVM classifier gives a decision function f (x)= sign (g (x)) for an input vector where, $g(x) = \sum_{i=1}^{m} w_i K(x, z_i) + b$ Here, f(x)=+1 means x is a member of a certain class and f(x)=-1 means x is not a member. zi s are called support vectors and are representatives of training examples, m is the number of support vectors. Therefore, the computational complexity of g(x) is proportional to m. Support vectors and other constants are determined by solving a certain quadratic programming problem. $K(x, z_i)$ is a kernel that implicitly maps vectors into a higher dimensional space. Typical kernels use dot products: $K(x, z_i) = k(x.z)$ .A polynomial kernel of degree d is given by $K(x, z_i)$= (1+x)d We can use various kernels, and the design of an appropriate kernel for a particular application is an important research issue.

Our general NE tagging system includes two main phases: training and classification. The training process has been carried out by YamCha[2] toolkit, an SVM based tool for detecting classes in documents and formulating the NE tagging task as a sequence labeling problem. Here, both *one vs rest* and *pairwise* multi-class decision methods have been used. We have also carried out different experiments with the various degrees of the *polynomial kernel function*. In *one vs rest* strategy, K binary SVM classifiers may be created where each classifier is trained to distinguish one class from the remaining K-1 classes. In pairwise classification, we constructed K (K-1)/2 classifiers (here, K=17, no. of NE tags) considering all pairs of classes, and the final decision is given by their weighted voting. For classification, we have used TinySVM-0.07[3] classifier that seems to be the best optimized among publicly available SVM toolkits.

3 Named Entity Tagset

In the present work, the NE tagset used have been further subdivided into the detailed categories in order to denote the boundaries of NEs properly. Table 1 shows examples.

Table 1: Named entity examples

NE Tag	Meaning	NE Examples
B-LOC	Beginning, Internal or the End of a multiword location name	ইথম (Itham)
I-LOC		মোইরাং (Moirang)
E-LOC		পুরেল (Purel)
PER	Single word person name	ইরাবত (Irabot)
LOC	Single word location name	হিয়াংথাং (Hiyangthang)
ORG	Single word organization name	এআর (AR)

4 Baseline NER System

Here, we describe an active learning based NER system in Manipuri that has been used as the *baseline* model. Stacked framework for learning to predict dependency structures for natural language sentences is reported by (Martins et. at., 2008). The active learning method generates lexical context patterns from the unlabeled Manipuri news corpus of 174,921 wordforms. These ranked lexical context patterns have been also used as the features of the individual supervised classifiers in order to improve their performance. Given a small set of seed examples and an

[2] http://chasen-org/~taku/software/yamcha/

[3] http://cl.aist-nara.ac.jp/~taku-ku/software/TinySVM

unlabeled corpus, the algorithm can generate the lexical context patterns in a bootstrapping manner. Each seed name serves as a positive example for its own NE class, negative example for other NE classes and error example for non-NEs.

4.1 Seed List Preparation

We have collected the frequently occurring words from the unlabeled corpus of 174,921 wordforms to use as the seeds. There are 30, 40 and 30 entries in the person, location and organization seed lists, respectively.

4.2 Lexical Pattern Generation

The unlabeled corpus is tagged with the elements from the seed lists. For example,

<*Person name*> Sonia Gandhi </*Person name*>,
<*Location name*> Kolkata </*Location name*> and
<*Organization name*> Jadavpur University </*Organization name*>

For each tag T inserted in the training corpus, the algorithm generates a lexical pattern p using a context window of maximum width 6 (excluding the tagged NE) around the left and the right tags, e.g., p = [l-3l-2 l-1 <T> ...</T> l+1 l+2 l+3], where, l±i are the context of p. Any of l±i may be a punctuation symbol. In such cases, the width of the lexical patterns will vary. All these patterns, derived from the different tags of the training corpus, are stored in a Pattern Table (or, set P), which has four different fields namely, pattern id (identifies any particular pattern), pattern example (the pattern itself), pattern type (Person name/Location name/Organization name) and relative frequency (indicates the number of times any pattern of a particular type appears in the entire training corpus relative to the total number of patterns generated of that type). This table has 1745 entries, out of which 1249 patterns are distinct.

4.3 Evaluation of Patterns

Every pattern p in the set P is matched against the same unannotated corpus. In a place, where the context of p matches, p predicts the occurrence of the left or the right boundary of name. The POS information of the words as well as some linguistic rules and/or length of the entity have been used in detecting the other boundary of the entity. The extracted entity may fall in one of the following categories:

- positive example: The extracted entity is of same NE type as that of the pattern.
- negative example: The extracted entity is of different NE type as that of the pattern.
- error example: The extracted entity is not at all a NE.

4.4 Candidate Pattern Extraction

For each pattern p, we have maintained three different lists for the positive, negative and error examples. The type of the extracted entity is determined by checking whether it appears in any of the seed lists (person/location/organization); otherwise, its type is determined manually. The positive and negative examples are then added to the appropriate seed lists. We then compute the accuracy of the pattern as follows:

$$accuracy(p)= |positive (p)|/[| positive (p)| + |negative (p)| + |error(p)|],$$

where, positive(p), negative(p) and error(p) are the respective positive, negative and error examples of the pattern p. A threshold value of accuracy has been chosen and the patterns below this threshold value are discarded. A pattern is also discarded if its total positive count is less than a predetermined threshold value. The remaining patterns are ranked by their relative frequency values. The n top high frequent patterns are retained in the pattern set P and this set is denoted as Accept Pattern.

4.5 Generation of New Patterns

All the positive and negative examples extracted by a pattern p in Step 4.4 can be used to generate further patterns from the same training corpus. Each new positive or negative instance (not appearing in the seed lists) is used to further tag the training corpus. We repeat steps 4.2-4.4 for each new NE until no new patterns can be generated. The threshold values of accuracy, positive count and relative frequency are chosen in such a way that in the first iteration of the algorithm at least 5% new patterns are added to the set P. A newly generated pattern may be identical to a pattern that is already in the set P. In such case, the type and relative frequency fields in the set P are updated accordingly. Otherwise, the newly generated pattern is added to the set with the type and relative frequency fields set properly. The algorithm terminates after the 10 iterations and there are 1929 distinct entries in the set P.

5 Features for Manipuri Named Entity Recognition

The main features for the NER task have been identified based on the different possible combination of available words and tags contexts. The features also include prefix and suffix for all words. The term prefix/suffix is a sequence of first/last few characters of a word, which may not be a linguistically meaningful prefix/suffix. The use of prefix/suffix information works well for the highly inflected languages as like the Indian languages. We have considered different combinations from the following set for inspecting the best set of features for NER in Manipuri:

F={ $w_{i-m},..,w_{i-1},w_i,w_{i+1},....,w_{i+n}$, |prefix|<=n, |suffix|<=n, NE tag(s) of previous word(s), POS tag(s) of the current and/or the surrounding word(s), First word, Length of the word, Digit information, Infrequent word}, where w_i is the current word; w_{i-m} is the previous m^{th} word and w_{i+n} is the next n^{th} word. Following are the details of the features:

1 Context word feature: Preceding and following words of a particular word since the surrounding words carry effective information for the identification of NEs.

2 Word suffix: Word suffix information is helpful to identify NEs. This is based on the observation that the NEs share some common suffixes. The fixed length (say, n) word suffix of the current and/or the surrounding word(s) can be treated as the feature. If the length of the corresponding word is less than or equal to n − 1 then the feature values are not defined and are denoted by 'ND'. The feature value is also not defined (ND) if the token itself is a punctuation symbol or contains any special symbol or digit. Word suffixes are the effective features and work well for the highly inflective Indian languages like Manipuri.

3 Word prefix: Word prefixes are also helpful to identify NEs. It is based on the observation that NEs share some common prefix strings. This feature has been defined in a similar way as that of the fixed length suffixes.

4 Named Entity Information: The NE tag(s) of the previous word(s) have been used as the only dynamic feature in the experiment. The output tag of the previous word is very informative in deciding the NE tag of the current word.

5 Digit features: Several binary valued digit features have been defined depending upon the

(i). Presence and/or the exact number of digits in a token.

(a). CntDgtCma: Token consists of digits and comma

(b). CntDgtPrd: Token consists of digits and periods

(ii). Combination of digits and symbols. For example,

(a). CntDgtSlsh: Token consists of digit and slash

(b). CntDgtHph: Token consists of digits and hyphen

(c). CntDgtPrctg: Token consists of digits and percentages

(iii). Combination of digit and special symbols. For example,

(a). CntDgtSpl: Token consists of digit and special symbol such as $, # etc.

These binary valued digit features are helpful in recognizing miscellaneous NEs such as measurement expression and percentage expression.

6 Infrequent word: The frequencies of the words in the training corpus have been calculated. A cut off frequency has been chosen in order to consider the words that occur with less than the cut off frequency in the training corpus. A binary valued feature ‘Infrequent’ is defined to check whether the current word appears in this infrequent word list or not. This is based on the observation that the infrequent words are most probably NEs.

7 Length of a word: This binary valued feature is used to check whether the length of the current word is less than three or not. We have observed that very short words are most probably not the NEs.

8 Part of Speech (POS) information: We have used an SVM-based POS tagger (Doren et al., 2008) that was originally developed with 26 POS tags, defined for the Indian languages. The POS information of the current and/or the surrounding words can be effective for NE identification.

6 Evaluation Results

Manipuri is a resource poor language. Applying statistical models to the NER problem requires large amount of annotated corpus in order to achieve reasonable performance. But, annotated corpus for Manipuri is not available. We have manually annotated approximately 28,629 wordforms of the news corpus with the NE tagset. The annotation has been carried out by us and verified by an expert.

The active learning technique that is used to generate the potential NE patterns has been considered as the *baseline* model. This active learning method is trained on a corpus of 174,921 wordforms, out of which 37,443 wordforms are unique.

The statistics of the training, development and test set is presented in Table 2.

Table 2: Statistics of the training, development and test sets

	Training	Development	Test
# of sentences	1235	732	189
#of wordforms	28,629	15,000	4,763
# of distinct wordforms	8671	4,212	2,207

7 Best Feature Selection for SVM

To identify the best set of features in the SVM model, a number of experiments have been conducted taking the different combinations from the set of features F in order to find out the best combination of features. From our empirical analysis, we found that the following combination of features gives the best result for the development set. F={ $W_{i-2}W_{i-2}W_{i-1}W_iW_{i+1}W_{i+2}$, Prefixes and suffixes of length upto three characters of the current word, dynamic NE tags of the previous two words, POS tags of the previous two and next two words, Digit information, Length of the word, Infrequent word}.The various notations used in the experiments are presented in Table 3.

Table 3: Meaning of the notations

Notation	Meaning
W[-i,+j]	Words spanning from the ith left position to the jth right position
POS[-i, +j]	POS tags of the words spanning from the ith left to the jth right positions
Pre	Prefix of the word
Suf	Suffix of the word
NE [-i, -j]	NE tags of the words spanning from the ith left to the jth left positions

We present the results on the development set in Table 4 in terms of Recall (R), Precision (P) and F-Score (FS) for some of the experiments. In this work, we have considered SVM that parses from left to right.

Table 4: Results on the development set

Feature	R (in %)	P(in %)	FS(in%)
Static: W[-2,+2], POS[-2,+2], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-2,-1]	**94.2**	**98.47**	**96.29**
Static: W[-3,+3], POS[-3,+3], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-3,-1]	88.91	97.82	93.15
Static: W[-3,+2], POS[-3,+2], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-3,-1]	91.3	96.99	94.06
Static: W[-4,+3], POS[-4,+3], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-2,-1]	87.05	97.66	92.05
Static: W[-4,+3], POS[-4,+3], |Pre|<=3, |Suf|<=3, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-3,-1]	85.28	98.03	91.21
Static: W[-2,+2], POS[-2,+2], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-2,-1]	88.70	98.49	93.34
Static: W[-3,+3], POS[-3,+3], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-3,-1]	87.05	97.09	91.79
Static: W[-4,+3], POS[-4,+2], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-2,-1]	78.55	97.54	87.02
Static: W[-4,+4], POS[-4,+4], |Pre|<=4, |Suf|<=4, Length, Infrequent, FirstWord, Digit **Dynamic:** NE[-3,-1]	71.71	97.44	82.62

8 Results on the Test Set

After the selection of the best feature set, the SVM based system is tested on the test set of 4,763 wordforms. Results are presented in Table 5 including the *baseline* model. For the *baseline* model, each pattern of the *Accept Pattern* set is matched with the test set. The identified NE is assigned the tag with the NE tag of the matched pattern. The system has demonstrated the *Recall*, *Precision* and *F-Score* values of 92.32%, 94.03% and 93.17%, respectively. Clearly, this shows significant improvement over the *baseline* model.

Table 5: Results on the test set

Model	Recall	Precision	F-Score
Baseline	60.21	54.12	57.01
SVM	92.32	94.03	93.17

Now, the patterns of the *baseline* model described in Section 4 have been used as the features of the SVM model. Words in the left and/or the right contexts of person, location and organization names carry effective information that could be helpful in their identification. A feature 'ContextInformation' is defined by observing the words in the window [−3, 3] (three words spanning to both left and right) of the current word.

9 Conclusion

In this paper, we have reported a NER system for a resource-constrained language, namely Manipuri. Initially, an active learning technique has been used to develop a baseline NER system. This is based on the lexical context patterns learned from the unlabeled corpus of 174,921 wordforms. This unlabeled corpus is then manually annotated with a coarse-grained tagset of four NE tags in order to apply SVM. The SVM classifier makes use of the different contextual information of the words along with the various orthographic word-level features. The lexical context patterns generated from the unlabeled corpus have been also used as the features of the SVM. A number of experiments have been carried out to find out the best set of features for NER in Manipuri. The system has been trained and tested with the 28,629 and 4,763 wordforms, respectively. We get effective result from news domain. The system has demonstrated the *Recall*, *Precision* and *F-Score* values of 93.91%, 95.32 and % 94.59%, respectively.

References

Bikel, D. M. 1998. Nymble: A High-performance Learning Name-finder. *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, 94-201, Morgan Kaufmann Publishers.

Borthwick, A., J. Sterling, E. Agichtein and R. Grishman, R. 1998. NYU: Description of the MENE Named Entity System as Used in MUC-7. *In Proceedings of the MUC-7*, Fairfax.

Doren Singh, Thoudam, A. Ekbal and S. Bandyopadhyay. 2008. Manipuri POS Tagging using CRF and SVM: A Language Independent Approach. *In Proceedings of the 6th International Conference on Natural Language Processing (ICON-08)*, India.

Ekbal, Asif, S. Naskar and S. Bandyopadhyay. 2007. Named Entity Recognition and Transliteration in Bengali. Named Entities: Recognition, Classification and Use, *Special Issue of Lingvisticae Investigationes Journal*, 30:1 (2007), 95-114.

Ekbal, Asif, R. Haque and S. Bandyopadhyay. 2008. Named Entity Recognition in Bengali: A Conditional Random Field Approach. *In Proceedings of 3rd International Joint Conference on Natural Language Processing* (IJCNLP-08), 589-594.

Ekbal, Asif, and S. Bandyopadhyay. 2008. Bengali Named Entity Recognition using Support Vector Machine. *In Proceedings of the Workshop on Named Entity Recognition on South and South East Asian Languages (NERSSEAL)*, IJCNLP-08, 51-58.

Lafferty, J., A. McCallum and F. Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.* In Proc. of 18th ICML, 282-289.

Li, Wei and Andrew McCallum. 2003. *Rapid Development of Hindi Named Entity Recognition Using Conditional Random Fields and Feature Inductions*. ACM TALIP, 2(3), (2003), 290-294.

Martins, A. F. T., D. Das, N.A. Smith and E.P. Xing., 2008. Stacking Dependency Parsers. *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* 157-166, Hawaii

Vapnik, Vladimir N. 1995. *The Nature of Statistical Learning Theory*. Springer.

Yamada, H., T. Kudo and Y. Matsumoto. 2001. *Japanese Named Entity Extraction using Support Vector Machine*. In Transactions of IPSJ, Volume (43), 44–53.