A Comparison of Performance of Sequential Learning Algorithms on the Task of Named Entity Recognition for Indian Languages

Awaghad Ashish Krishnarao, Himanshu Gahlot, Amit Srinet, and D.S. Kushwaha

Motilal Nehru National Institute of Technology,
Allahabad – 211004,
India
{awaghad.ashish,himanshu.gahlot86,amit.vers}@gmail.com,
dsk@mnnit.ac.in

Abstract. We have taken up the issue of named entity recognition of Indian languages by presenting a comparative study of two sequential learning algorithms viz. Conditional Random Fields (CRF) and Support Vector Machine (SVM). Though we only have results for Hindi, the features used are language independent, and hence the same procedure could be applied to tag the named entities in other Indian languages like Telgu, Bengali, Marathi etc. that have same number of vowels and consonants. We have used CRF++ for implementing CRF algorithm and Yamcha for implementing SVM algorithm. The results show a superiority of CRF over SVM and are just a little lower than the highest results achieved for this task. This can be attributed to the non-usage of any pre-processing and post-processing steps. The system makes use of the contextual information of words along with various language independent features to label the Named Entities (NEs).

Keywords: Support Vector Machines, Conditional Random Fields, Maxent, NER, Hindi, Named Entities.

1 Introduction

Named Entity Recognition for European Languages has already achieved a significant accuracy (Sang, 2002 [6]; Sang and De Meulder, 2003 [5]), which is especially high for English, and even for East Asian languages (Sassano and Utsuro, 2000 [16]). But the same task for Indian languages like Devnagiri (Hindi) is lagging far behind due to its various intricacies like, missing capitalizatoin information, which is the single most important information for identifying NEs. Indian languages also lack a formal and large list of gazetteers which makes pre-processing inefficient. Indian languages also have the problem of disambiguation of common nouns from proper nouns. A fairly large no. of frequently used words (common nouns) in Indian languages can also be used as named entities. For example, words like *Vivek, Kamal, Pankaj, Deepak* etc. which are proper nouns can also confer some meaning and can be used as common nouns. While in english almost all proper nouns are meaningless viz. *Harry, Kate, Leonardo, Jessica* etc. Hence the disambiguation in usage of a word as common

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 123–132, 2009.

noun or proper noun using its contextual features in Indian languages is more difficult and important than that for European languages.

Due to the above mentioned problems NER task for Indian languages is more difficult. Hence, even though at least four of them figure in the top ten spoken languages of the world these languages are less studied by researchers. Our comparative study of NER for these languages using two sequential labelling algorithms will provide an insight to researchers to work on the best method and shape their research accordingly. As opposed to the four tag tagset used in CoNLL 2003 shared task we have used a twelve tag tagset which was used in NERSSEAL-08 shared task. The system first identifies the named entities and then predicts their correct tags using the model generated through training.

2 Previous Work

The latest research work on named entity recognition for Indian languages was reported in the NERSSEAL-08 shared task against a baseline that uses maximum entropy based name finder tuned for English but trained on data from five South Asian languages viz. Hindi, Telugu, Bengali, Oriya and Urdu. The highest F-measure of 65.13% was reported by Saha et al. while the second highest was 50.06%, reported by Karthik et al. Our tagset and test data is the same as was used in this workshop.

Mainly two approaches are applied for NER :

- Linguistic approach
- Machine Learning approach

Linguistic approach works on handcrafted rules which are written by skilled linguists. Previous rule based NER systems, containing mainly lexicalized grammar, gazetteer lists, and list of trigger words, are capable of providing upto 92% F-measure accuracy for English (Grishman, 1995 [8]; McDonald, 1996 [17]). These systems have a disadvantage that they require huge experience and grammatical knowledge of the particular language or domain and are not transferable to other languages or domains. However, given the closer nature of many Indian languages, the cost of adaptation of a resource from one language to another could be quite less (Singh and Surana, 2007 [2]).

The machine learning techniques tried for NER include the following:

- Hidden Markov Models or HMM (Zhou and Su, 2001 [7])
- Decision Trees (Isozaki, 2001 [9])
- Maximum Entropy (Borthwick et al., 1998 [1])
- Support Vector Machines or SVM (Takeuchi and Collier, 2002 [11])
- Conditional Random Fields or CRF (Settles, 2004 [12], Li and Mccallum, 2004 [15])

These techniques make use of a large amount of NE annotated training data to acquire high level language knowledge.

Srihari et al. (2000) [19] used a combination of different ML approaches.They combined MaxEnt, HiddenMarkov Model (HMM) and handcrafted rules to build an NER system. Such hybrid systems have been generally more effective at the task of NER.

Previous work on NER for Indian and some other South and South East Asian Languages has been done by McCallum and Li, 2003 [3] and Cucerzan and Yarowsky, 1999 [18] but this work was just an extension of the work done on European Languages. Cucerzan and Yarowsky [18] in their language independent NER work used morphological and contextual evidences. They ran their experiment with 5 languages - Romanian, English, Greek, Turkish and Hindi. Among these the accuracy for Hindi was the worst. For Hindi the system achieved 41.70% F- measure with a very low recall of 27.84% and about 85% precision. The F-measure of the system developed by Wei Li and Andrew McCallum (2004) [15] was a lot better, giving an F measure of 71.50% against a training data of 340000 words. The maximum accuracy for NER in Hindi is reported by Kumar and Bhattacharyya, (2006) [14]. They achieved an F measure of 79.7% using a Maximum Entropy Markov Model.

3 Named Entity Tagset

We have used a twelve tag tagset for our classification task. This tagset gives a finer classification which in turn is helpful for better machine translation task. These tags are briefly explained in Table 1.

Table 1. The twelve tag tagset used in our task of named entity recognition

Tag	Name	Description
NEP	Person	Deepak, Gandhi
NED	Designation	Secretary, Director
NEO	Organization	Municipal Corporation
NEA	Abbreviation	C.E.O., A.D.A.
NEB	Brand	Honda, Reebok (ambiguous)
NETP	Title-Person	Mr., Mrs., Dr.
NETO	Title-Object	Hamlet, Othello
NEL	Location	Mumbai, Bangalore, Delhi
NETI	Time	18th September, 1984 (ambiguous)
NEN	Number	7.94, 8.56, 420
NEM	Measure	4 grams, Rs 4,000
NETE	Terms	Botany, Maximum Entropy

4 Data Description and Features Used

The data for our task was in *Shakti Standard Format (SSF)* which is shown below :

```
0     ((        SSF
1.    ((        NP  <ne=NEP>
1.1   Mahatma
1.2   Gandhi
      ))
2.    ke
```

3. kehne
4. par
))

We have converted this data to BIO format (B=Begin of chunk, I=Inside chunk, O=Outside the chunk) as follows:

Mahatma	B-NEP
Gandhi	I-NEP
Ke	O-NOT
kehne	O-NOT
par	O-NOT

The input format for both CRF++ and Yamcha SVM is same i.e. the word to be classified is followed by its features and then the class label. The initial training data for our classification task consisted of 503179 instances (words). The testing data had 40543 instances. This data had too many sentences which did not contain even a single named entity hence we refined the data by eliminating these sentences. Hence our refined data had all those sentences which had one or more named entities. The refined data had 301620 instances. We have used the following features :

- Word Window – A window size of 3 has been used to take into account the words in context of the current word.
- Prefix information – Prefix information upto length 3 is used.
- Suffix information – Suffix information upto length 3 is used.
- Length of the word – Word length information is used.
- Sentence start information – Whether the current word is the start of the sentence or not.
- Two consecutive digits – Occurrence of two consecutive numbers is quite frequent and hence it served as a good feature.
- Four consecutive digits – It mainly indicated the occurrence of ‘years’. Served as a good feature.
- Word class and Brief class - Words are also assigned a generalized ‘word class (WC)’, which replaces all letters with ‘a’, digits with ‘0’, punctuation marks with ‘p’, and other characters with ‘-’. There is a similar ‘brief class (BWC) (Settles 2004 [12])’ which collapses consecutive characters into one. Thus the words “D.D.T.” and “AB-1946” would both be given the features WC=apapap, BWC=apapap.

5 Learning Algorithms Applied

We used two learning algorithms for our classification task : Conditional Random Fields (CRFs) (Lafferty et al., 2001 [10]) and Support Vector Machines (SVMs) (Cortes and Vapnik, 1995 [4]). CRFs are implemented using CRF++ while SVMs are implemented using Yamcha. These algorithms are discussed below.

5.1 Conditional Random Fields

Conditional Random Fields (CRFs) are undirected graphical models used to calculate the conditional probability of values on designated output nodes given values assigned to other designated input nodes.

In the special case in which the output nodes of the graphical model are linked by edges in a linear chain, CRFs make a first-order Markov independence assumption, and thus can be understood as conditionally trained finite state machines (FSMs). Let $o = (o_1,o_2,o_3 ,o_4 ,... o_T)$ be some observed input data sequence, such as a sequence of words in text in a document,(the values on n input nodes of the graphical model). Let S be a set of FSM states, each of which is associated with a label, l Є £.

Let $s = (s_1,s_2 ,s_3 ,s_4 ,... s_T)$ be some sequence of states, (the values on T output nodes). By the Hammersley Clifford theorem, CRFs define the conditional probability of a state sequence given an input sequence to be:

$$P(s|o) = 1/Z_0 * exp(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)) .$$

where *Zo* is a normalization factor over all state sequences is an arbitrary feature function over its arguments, and λk is a learned weight for each feature function. A feature function may, for example, be defined to have value 0 or 1. Higher λ weights make their corresponding FSM transitions more likely. CRFs define the conditional probability of a label sequence based on the total probability over the state sequences,

$$P(l|o) = \sum_{s:l(s)=l} P(s|o) .$$

where *l(s)* is the sequence of labels corresponding to the labels of the states in sequence s.

Note that the normalization factor, Zo, (also known in statistical physics as the partition function) is the sum of the scores of all possible states.

$$Z_0 = \sum_{s \varepsilon s^T} exp(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)) .$$

And that the number of state sequences is exponential in the input sequence length, T. In arbitrarily-structured CRFs, calculating the partition function in closed form is intractable, and approximation methods such as Gibbs sampling or loopy belief propagation must be used. In linear-chain structured CRFs (in use here for sequence modeling), the partition function can be calculated efficiently by dynamic programming.

5.2 Support Vector Machines

Support Vector Machines (SVMs) are supervised machine learning algorithm for binary classification on a feature vector space $x \varepsilon R^L$.

$$w \cdot x + b = 0, \qquad w \varepsilon R^L, \quad b \varepsilon R . \tag{1}$$

Suppose the hyperplane (1) separates the training data, $\{(xi, yi) \mid xi \varepsilon R^L, \ y_i \varepsilon \{\pm 1\}, 1 \leq i \leq l\}$, into two classes such that

$$y_i.(w.x_i + b) \geq 1 . \tag{2}$$

While several of such separating hyperplanes exist, SVMs find the optimal hyperplane that maximizes the margin (the distance between the hyperplane and the nearest points). Such a hyperplane is known to have the minimum expected test error and can be solved by quadratic programming. Given a test example x, its label y is decided by the sign of discriminant function $f(x)$ as follows:

$$f(x) = w \cdot x + b, \tag{3}$$
$$y = \text{sgn}(f(x)), \tag{4}$$

For linearly non-separable cases, feature vectors are mapped into a higher dimensional space by a nonlinear function $\Phi(x)$ and linearly separated there. In SVMs' formula, since all data points appear as a form of inner product, we only need the inner product of two points in the higher dimensional space. Those values are calculated in R^L without mapping to the higher dimensional space by the following function $K(x_i, x_j)$ called a kernel function,

$$\Phi(x_i) \cdot \Phi(x_j) = K(x_i, x_j) , \tag{5}$$

Since SVMs are binary classifiers, we must extend them to multi-class classifiers to predict $k > 2$ POS tags.

Among several methods of multi-class classification, we employ the one-versus-rest approach. In training, k classifiers $f_i(x)$ $(1 \leq i \leq k)$ are created to classify the class i from all other classes,

$f_i(x) \geq +1$ x belongs to the class i,
$f_i(x) \leq -1$ otherwise.

Given a test example x, its class c is determined by the classifier that gives the largest discriminating function value,

$$c = \text{argmax } f_i(x) \tag{6}$$

6 Results

Table 2 shows the resulting classification of individual tags using both CRFs and SVMs on full data, on full data using a word window and on refined data. It is clearly evident from the results that a refined training data i.e. a data consisting of only those sentences which have at least one named entity outperforms the accruacy

Table 2. Comparison of CRFs and SVMs with full data and refined data. Refined data shows a better accuracy than full data while overall CRFs outperform SVMs.

Algo	Tag	NEP t=245	NED t=72	NEO t=103	NEA t=8	NETO t=325	NEL t=235	NETI t=92	NEN t=514	NEM t=31	NETE t=1080	NEB t=0; NETP t=5
	Data	*c*	*c*	*c*	*c*	*c*	*c*	*c*	*c*	*c*	*c*	*c*
CRFs	Full Data	90	20	26	2	12	110	53	368	14	90	0
	Word Window Four	94	21	23	3	18	109	53	364	15	90	0
	Ref. Data	98	19	27	3	36	129	59	421	17	127	0
SVMs	Full Data	33	18	10	1	8	62	34	326	13	51	0
	Word Window Four	38	17	10	1	9	63	36	333	13	67	0
	Ref. Data	48	17	12	1	9	70	37	367	13	76	0

t = total count ; c = correct count; Full Data = 503179 instances ;
Ref. Data = Refined Data = 301620 instances; Experiment with word window four is performed on full data.

of the data having a large number of sentences without a single named entity. It is also clearly evident that CRFs outperform SVMs when it comes to Named Entity Resolution.

6.1 Calculation of F-Measure, Precision and Recall

We present our result on three measures of performance calculated for three cases : maximal named entities, nested named entities and lexical matches. Thus, there are nine measures in total :

- Maximal Precision: $P_m = C_m / R_m$
- Maximal Recall: $R_m = C_m / T_m$
- Maximal F-Measure: $F_m = (2 * P_m * R_m) / (P_m + R_m)$
- Nested Precision: $P_n = C_n / R_n$
- Nested Recall: $R_m = C_n / T_n$
- Nested F-Measure: $F_n = (2 * P_n * R_n) / (P_n + R_n)$
- Lexical Precision: $P_l = C_l / R_l$
- Lexical Recall: $R_l = C_l / T_l$
- Lexical F-Measure: $F_l = (2 * P_l * R_l) / (P_l + R_l)$

where *C* is the number of correctly retrieved (identified) named entities, *R* is the total number of named entities retrieved by the system being evaluated (correct plus incorrect) and *T* is the total number of named entities in the reference data.

Table 3 lists these measures for various experiments performed.

Table 3. The nine performance measures for each experiment (P=Precision; R=Recall)

Algo	Type	Maximal			Nested			Lexical		
	Data	P	R	Fβ	P	R	Fβ	P	R	Fβ
CRFs	**Full Data**	0.55	0.27	0.37	0.58	0.27	0.37	0.79	0.29	0.42
	Word Window four	0.56	0.27	0.37	0.60	0.27	0.37	0.81	0.29	0.43
	Ref. Data	0.52	0.32	0.40	0.56	0.32	0.40	0.76	0.34	**0.47**
SVMs	**Full Data**	0.61	0.21	0.31	0.66	0.20	0.31	0.86	0.20	0.33
	Word Window four	0.56	0.23	0.35	0.57	0.22	0.32	0.83	0.26	0.35
	Ref. Data	0.59	0.25	0.35	0.63	0.24	0.35	0.82	0.24	0.37

7 Error Analysis

As the results show the f-value measure is highest for CRFs using a refined data while the SVMs give the lowest f-value measure. These results are obtained against a base line of 25.68%. We can easily infer that the sparseness of the named entities plays a major role in deciding the final classification. Also, as a finite state machine derived from HMMs, CRFs can naturally consider state-to-state dependences and feature-to-state dependences. On the other hand, SVMs do not consider such dependencies. SVMs separate the data into categories via a kernel function. They implement this by mapping the data points onto an optimal linear separating hyperplane. Finally, SVMs do not behave well for large number of feature values. For large number of feature values, it would be more difficult to find discriminative lines to categorize the labels.

In recognition task, NETO and NETE are giving worst results since they are difficult to discriminate even manually. The low results of the whole process are due to the use of ambiguous and a vast variety of named entity tags.

8 Conclusion

Since the Indian sub-continent has a large variety of regional languages which inherently have the same number of vowels and consonants and also the semantic of construction of words, the results give future researchers a direction regarding the suitability of the algorithms and also to focus their research in the areas of shortcomings viz. on the ambigous tags like NETO and NETE as also the usage of gazetteers and other pre-processing and post-processing steps to refine the results. Nevertheless, we feel that our results are far better since we have achieved higher accuracy without any pre-processing or post-processing steps and it is expected to increase the efficiency of the system with these aids.

References

1. Borthwick, A., Sterling, J., Agichtein, E., Grishman, R.: Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In: Proceedings of the Sixth Workshop on Very Large Corpora, pp. 152–160 (1998)
2. Singh, A.K., Surana, H.: Can Corpus Based Measures be Used for Comparative Study of Languages? In: Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology, ACL (2007)
3. McCallum, A., Li, W.: Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Seventh Conference on Natural Language Learning (CoNLL) (2003)
4. Cortes, C., Vapnik, V.: Support-vector network. Machine Learning 20, 273–297 (1995)
5. Sang, E.F.T.K., De Meulder, F.: Language Independent Named Entity Recognition. In: Introduction to the CoNLL 2003 Shared Task. Development, vol. 922, p. 1341 (2003)
6. Erik, F., Kim Sang, T.: Introduction to the conll 2002 shared task: Language-independentnamed entity recognition. In: Proceedings of CoNLL 2002, Taipei, Taiwan, pp. 155–158 (2002)
7. Zhou, G.D., Su, J.: Named entity recognition using an HMM-based chunk tagger. In: Proceedings of the 40th AnnualMeeting on Association for Computational Linguistics, pp. 473–480 (2001)
8. Ralph, G.: The New York University System MUC-6 orWhere's the syntax? In: Proceedings of the Sixth Message Understanding Conference (1995)
9. Isozaki, H.: Japanese named entity recognition based on a simple rule generator and decision tree learning. In: Meeting of the Association for Computational Linguistics, India, pp. 306–313 (January 2001)
10. John, L., Andrew, M., Fernando, P.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001) (2001)
11. Takeuchi, K., Collier, N.: Use of support vector machines in extended named entity recognition. In: Proceedings of the sixth Conference on Natural Language Learning (CoNLL 2002) (2002)
12. Settles, B.: Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. log, 1:1
13. Karthik, G., Harshit, S., Ashwini, V., Praneeth, S., Misra, S.D.: Aggregating machine learning and rule based heuristics for named entity recognition. In: Proceedings of the IJCNLP 2008 Workshop on NER for South and South East Asian Languages, Hyderabad, India, pp. 25–32 (January 2008)
14. Kumar, N., Pushpak, B.: Named Entity Recognition in Hindi using MEMM. Technical Report, IIT Bombay, India (2006)
15. Wei, L., Andrew, M.: Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. ACM Transactions on Computational Logic (2004)
16. Sassano, M., Utsuro, T.: Named entity chunking techniques in supervised learning for japanese named entity recognition. In: Proceedings of the 18th conference on Computational linguistics, Morristown, NJ, USA, pp. 705–711. Association for Computational Linguistics (2000)

17. McDonald, D.: Internal and external evidence in the identification and semantic categorization of proper names. In: Boguraev, B., Pustejovsky, J. (eds.) Corpus Processing for Lexical Acquisition, pp. 21–39 (1996)
18. Cucerzan, S., Yarowsky, D.: Language independent named entity recognition combining morphological and contextual evidence. In: Proceedings of the Joint SIGDAT Conference on EMNLP and VLC 1999, pp. 90–99 (1999)
19. Srihari, R., Niu, C., Li, W.: A Hybrid Approach for Named Entity and Sub-Type Tagging. In: Proceedings of the sixth conference on Applied natural language processing (2000)