A Character n-gram Based Approach for Improved Recall in Indian Language NER

Praneeth M Shishtla
praneethms
@students.iiit.ac.in

Prasad Pingali
pvvpr
@iiit.ac.in

Vasudeva Varma
vv@iiit.ac.in

Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India

Abstract

Named Entity Recognition (NER) is the task of identifying and classifying all proper nouns in a document as person names, organization names, location names, date & time expressions and miscellaneous. Previous work (Cucerzan and Yarowsky, 1999) was done using the complete words as features which suffers from a low recall problem. Character n-gram based approach (Klein et al., 2003) using generative models, was experimented on English language and it proved to be useful over the word based models. Applying the same technique on Indian Languages, we experimented with Conditional Random Fields (CRFs), a discriminative model, and evaluated our system on two Indian Languages Telugu and Hindi. The character n-gram based models showed considerable improvement over the word based models. This paper describes the features used and experiments to increase the recall of Named Entity Recognition Systems which is also language independent.

1 Introduction

The objective of NER is to classify all tokens in a text document into predefined classes such as person, organization, location, miscellaneous. NER is a precursor to many language processing tasks. The creation of a subtask for NER in Message Understanding Conference (MUC) (Chinchor, 1997) reflects the importance of NER in Information Extraction (IE). NER also finds aplication in question answering systems (Toral et al., 2005; Molla et al., 2006), and machine translation (Babych and Hartley, 2003). NER is an essential subtask in organizing and retrieving biomedical information (Tsai, 2006). NER can be treated as a two step process

- identification of proper nouns.
- classification of these identified proper nouns.

Challenges in named entity recognition.

Many named entities (NEs) occur rarely in corpus if at all.

Ambiguity of NEs. Ex *Washington* can be a person's name or location.

There are many ways of mentioning the same NE. *Ex: Mahatma Gandhi, M.K.Gandhi, Mohandas Karamchand Gandhi, Gandhi* all refer to the same person. *New Jersey, NJ* both refer to the same location.

In English, the problem of identifying NEs is solved to some extent by using the capitalization feature. Most of the named entities begin with a capital letter which is a discriminating feature for classifying a token as named entity. In addition to the above challenges, the complexity of Indian Languages pose few more problems. In case of Indian languages there is no concept of capitalization. Ex: The person name *Y.S.R (in english)* is represented as *ysr* in the Indian Languages.

Agglutinative property of the Indian Languages makes the identification more difficult. *For example: hyderabad, hyderabad ki, hyderabadki, hyderabadlo, hyderabad ni, hyderabad ko* etc .. all refer to the place Hyderabad. where *lo, ki, ni* are all postpostion markers in Telugu and *ko* is a postposition marker in Hindi.

There are many ways of representing acronyms. The letters in acronyms could be the English alphabet or the native alphabet. Ex: *B.J.P* and *BaJaPa* both are acronyms of *Bharatiya Janata Party*. Indian Languages lack particular standard for forming acronyms.

Due to these wide variations and the agglutinative nature of Indian languages, probabilistic graphical models result in very less recall. If we are able to identify the presence of a named entity with a fairly good amount of accuracy, classification then can be done efficiently. But, when the machine fails to identify the presence of named entities, there is no chance of entity classification because we miss many of the named entities (less recall which results in less F-measure,$F_{\beta=1}$). So we focus mainly on the ways to improve the recall of the system. Also, Indian Languages have a relatively free word order, i.e. the words (named entities) can occupy any place in the sentence. This change in the word position is compensated using case markers.

2 Related Work & Our Contributions

The state-of-art techniques for Indic languages(Telugu and Hindi) use word based models which suffer from low recall, use gazetteers and are language dependent. As such there is no NER system for Telugu. Previously (Klein et al., 2003) experimented with character-level models for English using character based HMM which is a generative model. We experimented using the discriminative model for English, Hindi and Telugu.

- We propose an approach that increases the recall of Indic languages (even the agglutinative languages).
- The model is language independent as none of the language resources is needed.

3 Problem Statement

3.1 NER as sequence labelling task

Named entity recognition (NER) can be modelled as a sequence labelling task (Lafferty et al., 2001). Given an input sequence of words $W_1^n = w_1 w_2 w_3 ... w_n$, the NER task is to construct a label sequence $L_1^n = l_1 l_2 l_3 ... l_n$, where label l_i either belongs to the set of predefined classes for named entities or is none (representing words which are not proper nouns). The general label sequence l_1^n has the highest probability of occuring for the word sequence W_1^n among all possible label sequences, that is

$$\hat{L}_1^n = \text{argmax } \{\Pr(L_1^n \mid W_1^n) \}$$

3.2 Tagging Scheme

We followed the IOB tagging scheme (Ramshaw and Marcus, 1995) for all the three languages (English, Hindi and Telugu). In this scheme each line contains a word at the beginning followed by its tag. The tag encodes the type of named entity and whether the word is in the beginning or inside the NE. Empty lines represent sentence (document) boundaries. An example of the IOB tagging scheme is given in Table 1.

Words tagged with O are outside of named entities

Token	Named Entity Tag
Dr.	B-PER
Talcott	I-PER
led	O
a	O
team	O
of	O
researchers	O
from	O
the	O
National	B-ORG
Cancer	I-ORG
Institute	I-ORG

Table 1: IOB tagging scheme.

and the I-XXX tag is used for words inside a named entity of type XXX. Whenever two entities of type XXX are immediately next to each other, the first word of the second entity will be tagged B-XXX in order to show that it starts another entity. This tagging scheme is the IOB scheme originally put forward by Ramshaw and Marcus (Ramshaw and Marcus, 1995).

4 Conditional Random Fields

Conditional Random Fields (CRFs) (Wallach, 2004) are undirected graphical models used to calculate the conditional probability of values on designated output nodes given the 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 = $\langle O_1, O_2, ... O_T \rangle$ 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 Finite State Machine (FSM) states, each of which is associated with a label, l $\in \mathscr{L}$.
Let **s** = $\langle s_1, s_2, ...\ s_T, \rangle$ 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

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

where Z_o is a normalization factor over all state sequences, is an arbitrary feature function over its arguments, and λ_k 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 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, Z_o, (also known in statistical physics as the partition function) is the sum of the scores of all possible state sequences,

$$Z_o = \sum_{s \in S^T} * exp(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t))$$

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.

5 Features

There are many types of features used in NER systems.

Many systems use binary features i.e. the word-internal features, which indicate the presence or absence of particular property in the word. (Mikheev, 1997; Wacholder et al., 1997; Bikel et al., 1997). Following are examples of commonly used binary features: All-Caps (IBM), internal capitalization (eBay), initial capital (Abdul Kalam), uncapitalized word (can), 2-digit number (83, 73), 4-digit number (1983, 2007), all digits (8, 28, 1273) etc. The features that correspond to the capitalization are not applicable to Indian languages. Also, we have not used any of the binary features in any of our models.

Dictionaries: Dictionaries are used to check if a part of the named entity is present in the dictionary. These dictionaries are called as gazetteers. The problem with the Indian languages is that there are no proper gazetteers in Indian languages.

Lexical features like a sliding window $[w_{-2}, w_{-1}, w_o, w_1, w_2]$ are used to create a lexical history view. Prefix and suffix tries were also used previously (Cucerzan and Yarowsky, 1999).

Linguistics features like Part Of Speech, Chunk, etc are also used. In our approach we don't use any of these language specific (linguistic) information.

5.1 Our Features

In our experiments, we considered and character n-grams (ASCII characters) as tokens.
For example for the word *Vivekananda*, the 4-gram model would result in 8 tokens namely *Vive, ivek, veka, ekan, kana, anan, nand* and *anda*. If our current token (w_0) is *kana*

Feature	Example
current token: w_0	*kana*
previous 3 tokens: w_{-3}, w_{-2}, w_{-1}	*ivek,veka,ekan*
next 3 tokens: w_1, w_2, w_3	*anan,nand,anda*
compound feature: $w_0\ w_1$	*kanaanan*
compound feature: $w_{-1}\ w_0$	*ekankana*

In Indian Languages suffixes and other inflections get attached to the words increasing the length of the word and reducing the number of occurences of that word in the entire corpus. The character n-grams can capture these variations. The compound features also help in capturing such variations. The sliding window feature helps in guessing the class of the entity using the context. In total 9 features were used in training and testing. All the features are languge independent and no binary features are used.

6 Experimental Setup

6.1 Corpus

We conducted the experiments on three languages namely Telugu, Hindi and English. We collected the Telugu corpus from Eenadu, a telugu daily newspaper. The topics included politics, health and medicine, sports, education, general issues etc. The annotated corpus had 45714 tokens, out of which 4709 were named entities. We collected the English corpus from the Wall Street Journal (WSJ) news articles. The corpus had 45870 tokens out of which 4287 were named entities. And we collected the hindi corpus from various sources. The topics in the corpus included social sciences, biological sciences, financial articles, religion, etc. The hindi corpus is not a news corpus. The corpus had 45380 tokens out of which 3140 were named entities. We evaluated the hand-annotated corpus once to check for any errors.

6.2 Experiments

We conducted various experiments on Telugu and Hindi. Also, to verify the correctness of our model for other languages, we have conducted some experiments on English data also. In this section we describe the various experiments conducted on the Telugu, Hindi and English data sets.

We show the average performance of the system in terms of precision, recall and F-measure for Telugu, Hindi and English in Table 6 and then for the impact of training data size on performance of the system in Table 7 (Telugu), Table 8 (English) and Table 9 (Hindi). Here, precision measures the number of correct Named Entities (NEs) in the machine tagged file over the total number of NEs in the machine tagged file and the recall measures the number of correct NEs in the machine tagged file over the total number of NEs in the golden standard file while F-measure is the weighted harmonic mean of precision and recall:

$$F = \frac{\left(\beta^2+1\right)\ RP}{\beta^2 R+P}$$

with

$$\beta^2 = 1$$

where P is Precision, R is Recall and F is F-measure.

	Precision	Recall	$F_{\beta=1}$
words	89.66%	29.21%	44.07
n=2	77.36%	46.07%	57.75
n=3	85.45%	**52.81%**	**65.28**
n=4	79.63%	48.31%	60.14
n=5	74.47%	39.33%	51.47
n=6	76.32%	32.58%	45.67

Table 2: Precision,Recall and $F_{\beta=1}$ measure for Date & Time expressions in Telugu.

	Precision	Recall	$F_{\beta=1}$
words	83.65%	28.71%	42.75
n=2	80.29%	**36.30%**	**50**
n=3	78.26%	35.64%	48.98
n=4	81.03%	31.02%	44.87
n=5	75.42%	29.37%	42.28
n=6	53.21%	27.39%	36.17

Table 3: Precision,Recall & $F_{\beta=1}$ measure values for location names in Telugu.

	Precision	Recall	$F_{\beta=1}$
words	51.11%	18.70%	27.38
n=2	53.41%	38.21%	44.55
n=3	**69.35%**	**34.96%**	**46.49**
n=4	**69.35%**	**34.96%**	**46.49**
n=5	55.00%	26.83%	36.07
n=6	50.98%	21.14%	29.89

Table 4: Precision,Recall and $F_{\beta=1}$ measure values for organisation names in Telugu.

Table:6 shows the average precison(P),recall(R) and F-measure(F) values for NEs in Telugu.

Tables 2 to 5 show the P,R,F values for the individual categories of NEs in Telugu. Interestingly,

	Precision	Recall	$F_{\beta=1}$
words	57.32%	18.65%	28.14
n=2	55.77%	34.52%	42.65
n=3	**61.04%**	**37.30%**	**46.31**
n=4	56.92%	29.37%	38.74
n=5	60.50%	28.57%	38.81
n=6	54.21%	23.02%	32.31

Table 5: Precision,Recall and $F_{\beta=1}$ measure values for Person names in Telugu.

though we have not used any of the features pertaining to years and numbers we have acheived an appreciable F-measure of 65.28 for date & time expressions.

In each table the model with the highest F-measure is higlighted in bold. And, the tri-gram model performed best in most of the cases except with locations where bi-gram model performed well. But, even the tri-gram model ($F_{\beta=1}$=48.98) performed close to the bi-gram model (($F_{\beta=1}$=50).

For Hindi, the recall of the n-gram models(Table 6) is more than the word based models but the amount of increase in recall and F-measure is less. On examining, we found that the average number of named entities in the Hindi data were quite less. This is because the articles for hindi were taken from general articles. Whereas in case of English and Telugu, the corpus was collected from news articles, which had more probability of having new and more named entities, which can occur in a similar repeating pattern.

The character n-gram approach showed considerable improvement in recall and F-measure (with a drop in precision) in Telugu and Hindi, which are agglutinative in nature. In Telugu, there is a difference of 14.19 and 14.02 in recall and F-measure respectively between the word based model and the best performing n-gram model (n=3) of size 3. In Hindi, there is a difference of 2.34 and 2.33 in recall and F-measure respectively between the word based model and the best performing n-gram model (n=5). Even in case of non-agglutinative language like English there is a considerable improvement of 1.48 and 1.91 in recall and F-measure respectively between the word based model and best performing n-gram model (n=2) of size 2.

In almost all the cases the character based models performed better in terms of recall and F-measure than the word based models.

We also experimented changing the training data size keeping the testing data size unchanged for Telugu(Table 7) and English(Table 8) and Hindi(Table 9). From Table 7:All the models (words,character n-gram models) are able to learn as we increase the training data size. And the recall of the character n-gram models is considerably more than recall of the word based model. Also the 3-gram model performed well in almost all the runs. The rate of learning is more in case of 30K.
From Table 8, in all the runs, the bi-gram character model constantly performed the best. Also interestingly the model is able to achieve a least F-measure of 44.75 with just 10K words of training data. But, in case of Telugu,(Table 7) an F-measure of 44+ was reached with training data of size 35K i.e the learning rate for english is more for less amount of data. This is due to the reason that Telugu (Entropy=15.625 bits per character) (Bharati et al., 1998) is comparitively a high entropy language than English (Brown and Pietra, 1992). However for Hindi, the relative jump in the performance (compared to Telugu and English)is less. Even the entropy of Hindi (Entorpy=11.088) (Bharati et al., 1998) is more than English. This is also observed from the table (Table 10). The numbers in the second, third and fourth columns are the number of features for English,Telugu and Hindi respectively.

	English	Telugu	Hindi
words	29145	320260	685032
n=2	27707	267340	647109
n=3	45580	680720	1403352
n=4	64284	1162320	1830438
n=5	65248	1359980	1735614
n=6	57297	1278790	1433322

Table 10: Number of features calculated in the word based model for English,Telugu and Hindi.

7 Conclusion & Future Work

The character based n-gram approach worked better than the word based approach even with agglutinative languages. A considerably good NER for

Language	English			Telugu			Hindi		
	Precision	Recall	$F_{\beta=1}$	Precision	Recall	$F_{\beta=1}$	Precision	Recall	$F_{\beta=1}$
Words	92.42%	47.29%	62.56	70.38%	23.83%	35.6	51.66%	36.45%	42.74
n=2	81.21%	68.77%	74.47	65.67%	37.11%	47.42	37.30%	36.06%	36.67
n=3	88.37%	62.45%	73.18	71.39%	38.02%	49.62	54.89%	37.23%	44.37
n=4	93.17%	59.19%	72.39	70.17%	33.07%	44.96	54.67%	37.62%	44.57
n=5	90.71%	58.30%	70.98	66.57%	29.82%	41.19	53.78%	38.79%	45.07
n=6	91.03%	56.14%	69.45	55.68%	25.52%	35	51.79%	36.65%	42.92

Table 6: Average Precision, Recall and $F_{\beta=1}$ measure for English, Telugu and Hindi 'n' indicates the number of n-gram characters

Size	10K			20K			30K			35K		
Model	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$
words	58.04	8.46	14.77	56.54	14.06	22.52	67.90	21.48	32.64	71.03	23.31	35.1
n=2	53.81	13.80	21.97	60.31	**25.52**	**35.86**	63.68	31.51	42.16	65.16	35.55	46
n=3	68.07	**14.71**	**24.2**	64.71	24.35	35.38	70.22	**32.55**	**44.48**	71.79	**37.11**	**48.93**
n=4	71.23	13.54	22.76	63.42	21.22	31.8	68.14	28.12	39.82	68.16	31.77	43.34
n=4	69.92	11.20	19.3	61.20	19.92	30.06	63.90	26.04	37	66.96	29.30	40.76
n=6	52.38	8.59	14.77	52.70	16.54	25.17	56.13	22.66	32.28	55.16	24.35	33.79

Table 7: Effect of training data size on Average Precision,Recall and $F_{\beta=1}$ measure for Telugu.

Size	10K			20K			30K			35K		
Model	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$
words	81.84	30.79	44.75	86.54	40.93	55.57	89.04	45.95	60.62	89.80	46.35	61.14
n=2	71.49	**42.00**	**52.92**	74.80	**58.40**	**65.59**	75.46	**61.03**	**67.49**	76.63	**61.87**	**68.46**
n=3	76.09	28.85	41.84	81.15	50.03	61.9	81.31	54.28	65.11	82.18	56.84	67.2
n=4	83.42	25.75	39.36	83.35	42.93	56.67	88.01	48.70	62.7	87.40	50.25	63.81
n=5	81.95	25.64	39.06	84.48	41.00	55.21	86.81	44.47	58.81	88.07	47.43	61.66
n=6	79.24	26.89	40.16	83.31	38.18	52.36	89.34	42.88	57.95	87.71	44.32	58.88

Table 8: Effect of training data size on Average Precision,Recall and $F_{\beta=1}$ measure for English.

Size	10K			20K			30K			35K		
Model	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$	$P_{(\%)}$	$R_{(\%)}$	$F_{\beta=1}$
words	43.13	30.60	35.80	47.97	34.50	40.14	48.67	35.67	41.17	51.92	36.84	43.10
n=2	39.29	30.41	34.29	40.73	34.70	37.47	37.58	36.26	36.90	37.91	36.06	36.96
n=3	48.17	33.33	39.40	50.56	35.28	41.56	47.72	36.65	41.46	50.68	36.06	42.14
n=4	49.18	**35.09**	**40.96**	49.21	**36.26**	**41.75**	52.14	35.67	**42.36**	54.87	38.40	45.18
n=5	41.08	34.11	37.27	41.93	33.92	37.50	48.72	37.23	42.21	53.12	**39.77**	**45.48**
n=6	41.43	31.58	35.84	44.59	33.72	38.40	46.35	35.87	40.44	50.67	36.84	42.66

Table 9: Effect of training data size on Average Precision,Recall and $F_{\beta=1}$ measure for Hindi.

English can be built with less amount of data when we use character based models and for high entropy languages large amount of training data is necessary to build a considerably good NER. We are able to achieve an F-measure (49.62 for Telugu and 45.07 for Hindi) even without any extra features like regular expressions and gazetteer information. The character based n-gram models have worked well even with the discriminative models. A total of 9 features were used in training and testing. We have not used any of the language dependent resources and any binary features. To improve the efficiency of the system we plan to experiment with language specific resources like Part Of Speech (POS) Taggers, Chunkers, Morphological analyzers.. etc and also include some regular expressions and gazetteer information.

References

Bogdan Babych and Anthony Hartley. 2003. Improving machine translation quality with automatic named entity recognition.

Akshar Bharati, Prakash Rao K, Rajeev Sangal, and S.M.Bendre. 1998. Basic statistical analysis of corpus and cross comparison among corpora. Technical report, International Institute of Information Technology-Hyderabad(IIIT-H).

D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning namefinder.

Peter E Brown and Vincent J. Della Pietra. 1992. An estimate of an upper bound for the entropy of english.

Nancy Chinchor. 1997. Muc-7 named entity task definition. Technical Report Version 3.5, Science Applications International Corporation, Fairfax, Virginia.

S. Cucerzan and D. Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence.

D. Klein, J. Smarr, H. Nguyen, and C. Manning. 2003. Named entity recognition with character-level models.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Comput. Linguist.*, 23(3):405–423.

Diego Molla, Menno van Zaanen, and Daniel Smith. 2006. Named entity recognition for question answering. In *Proceedings of Australasian Language Technology Workshop 2006*, Sydney, Australia.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

Antonio Toral, Elisa Noguera, Fernando Llopis, and Rafael Muñoz. 2005. Improving question answering using named entity recognition. In *Proceedings of the 10th NLDB congress*, Lecture notes in Computer Science, Alicante, Spain. Springer-Verlag.

Richard Tzong-Han Tsai. 2006. A hybrid approach to biomedical named entity recognition and semantic role labeling. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 243–246, Morristown, NJ, USA. Association for Computational Linguistics.

N. Wacholder, Y. Ravin, and M. Choi. 1997. Disambiguation of proper names in text.

Hanna M. Wallach. 2004. Conditional random fields: An introduction. Technical Report MS-CIS-04-21, University of Pennsylvania, Department of Computer and Information Science, University of Pennsylvania.