Differential Evolution based Feature Selection and Classifier Ensemble for Named Entity Recognition

Utpal Kumar Sikdar Asif Ekbal* Sriparna Saha*
Department of Computer Science and Engineering
Indian Institute of Technology Patna, Patna, India
{utpal.sikdar,asif,sriparna}@iitp.ac.in

ABSTRACT
In this paper, we propose a differential evolution (DE) based two-stage evolutionary approach for named entity recognition (NER). The first stage concerns with the problem of relevant feature selection for NER within the frameworks of two popular machine learning algorithms, namely Conditional Random Field (CRF) and Support Vector Machine (SVM). The solutions of the final best population provides different diverse set of classifiers; some are effective with respect to recall whereas some are effective with respect to precision. In the second stage we propose a novel technique for classifier ensemble for combining these classifiers. The approach is very general and can be applied for any classification problem. Currently we evaluate the proposed algorithm for NER in three popular Indian languages, namely Bengali, Hindi and Telugu. In order to maintain the domain-independence property the *features are selected and developed mostly without using any deep domain knowledge and/or language dependent resources*. Experimental results show that the proposed two stage technique attains the final F-measure values of 88.89%, 88.09% and 76.63% for Bengali, Hindi and Telugu, respectively. The key contributions of this work are two-fold, *viz.* (i). proposal of differential evolution (DE) based feature selection and classifier ensemble methods that can be applied to any classification problem; and (ii). scope of the development of language independent NER systems in a resource-poor scenario.

KEYWORDS: Named Entity Recognition, Differential Evolution, Feature Selection, Classifier Ensemble.

Authors equaly contributed for this paper. * corresponding authors

Proceedings of COLING 2012: Technical Papers, pages 2475–2490,
COLING 2012, Mumbai, December 2012.

1 Introduction

Named Entity Recognition (NER) aims to identify and classify each word of a document into some predefined target categories such as person, location, organization, miscellaneous (e.g., date, time, number, percentage, monetary expressions etc.) and none-of-the-above. Over the decades, it has shown success in almost all application areas of Natural Language Processing (NLP) that includes information retrieval, information extraction, machine translation, question-answering and automatic summarization etc. Proper identification and classification of NEs are very crucial and pose a big challenge to the NLP researchers. The main challenge is due to the fact that named entity (NE) expressions are hard to analyze using traditional NLP because they belong to the open class of expressions, i.e., there is an infinite variety and new expressions are constantly being invented.

The problem of NER was actually formulated in Message Understanding Conferences (MUCs) [MUC6; MUC7] (Chinchor, 1995, 1998). The issues of correct identification of NEs were specifically addressed and benchmarked by the developers of information extraction system, such as the GATE system (Cunningham, 2002). The existing approaches for NER can be grouped into three main categories, namely rule-based, machine learning based and hybrid approach. Majority of the research focussed on machine learning (ML) approaches (Bikel et al., 1999; Borthwick, 1999; Sekine, 1998; Lafferty et al., 2001a; Yamada et al., 2001) because these are easily trainable, adaptable to different domains and languages as well as their maintenance are also being less expensive. In contrast, rule-based approaches lack the ability of dealing with the problems of robustness and portability. Each new source of text requires significant updates to the rules to maintain optimal performance and the maintenance costs could be quite steep.

Literature shows that most of the works carried out in this direction cover mostly English and European languages. There are also significant amount of works in some of the Asian languages like Chinese, Japanese and Korean. India is a multilingual country with great linguistic and cultural diversities. People speak in 22 different official languages that are derived from almost all the dominant linguistic families. Some works on NER for Indian languages can be found in (Ekbal and Saha, 2011; Ekbal and Bandyopadhyay, 2008; Ekbal et al., 2007; Ekbal and Bandyopadhyay, 2009b, 2007; Ekbal et al., 2008; Li and McCallum, 2004; Patel et al., 2009; Srikanth and Murthy, 2008; Shishtla et al., 2008; Vijayakrishna and Sobha, 2008). However, the works related to NER in Indian languages are still in the nascent stages due to the potential facts such as (Ekbal and Saha, 2011):

- Unlike English and most of the European languages, Indian languages lack capitalization information, which plays a very important role in NE identification.
- Indian person names are more diverse compared to the other languages and a lot of these words can be found in the dictionary with specific meanings.
- Indian languages are highly inflectional language providing one of the richest and most challenging sets of linguistic and statistical features resulting in long and complex word-forms.
- Indian languages do not conform to any fixed word-order.
- Indian languages are resource poor in nature- annotated corpora, name dictionaries, good morphological analyzers, Part-of-Speech (PoS) taggers etc. are not yet available in the required measure.
- Although Indian languages have a very old and rich literary history, technological devel-

opments are of recent origin.

- Web sources for name lists are available in English, but such lists are not available in Bengali, Hindi and Telugu forcing the use of transliteration.

In this paper we propose a two-stage approach for NER in Indian languages. The first step solves the problems of feature selection for NER within the frameworks of two robust machine learning algorithms, namely Conditional Random Field (CRF)(Lafferty et al., 2001a) and Support Vector Machine (SVM) (Vapnik, 1995). The performance of any classification technique depends on the features of training and test datasets. Feature selection, also known as variable selection, feature reduction, attribute selection or variable subset selection, is the technique, commonly used in machine learning, of selecting a subset of relevant features for building robust learning models. In a machine learning approach, feature selection is an optimization problem that involves choosing an appropriate feature subset. In CRF or SVM based models, relevant feature selection is a very crucial problem and also a key issue to improve the classifier's performance. Usually, heuristics are used to find the appropriate set of features in these classification models. In this paper we propose an evolutionary algorithm for automatic feature selection. The second stage deals with the problem of classifier ensemble. Classifier ensemble [1] is relatively a new direction of machine learning. The main idea behind classifier ensemble is that ensembles are often much more accurate than the individual classifiers that make them up. It is to be noted that all the existing ensemble techniques should have a way of combining the decisions of a set of classifiers. Existing approaches (e.g., stacking, AdaBoost, bagging etc.) combine the outputs of all classifiers by using either majority voting or weighted voting. The weights of votes depend on the error rate/performance of the individual classifiers. However, in reality, in an ensemble system all the classifiers are not equally efficient in detecting all types of output classes. Thus, while combining the classifiers using weighted voting, weights of voting should vary among the different output classes in each classifier. The weight should be high for that particular output class for which the classifier performs good. Otherwise, weight should be low for the output class for which its output is not very reliable. So, it is a crucial issue to select the appropriate weights of votes for all the classes in each classifier. The proposed algorithms for feature selection and classifier ensemble are based on an evolutionary algorithm, differential evolution (DE) (Storn and Price, 1997). To train and test the classifiers we use a set of features that are *mostly selected and developed without using any deep domain knowledge and/or language dependent resources*. The first stage of the algorithm produces a population that contains a set of solutions, some of them are effective with respect to recall whereas some are effective with respect to precision. In the second stage we combine the decisions of these solutions using the proposed DE based classifier ensemble technique. The proposed approach is evaluated for three resource-poor Indian languages, namely Bengali, Hindi and Telugu. Bengali is the seventh most spoken language in the world, second in India and the national language of Bangladesh. Hindi is the third most spoken language in the world and the national language of India. Results show that this two stage DE based technique attains the final F-measure values of 88.89%, 88.09% and 76.63% for Bengali, Hindi and Telugu, respectively. The proposed approach is compared with the conventional majority and weighted voting techniques, popular ensemble methods like stacking, AdaBoost and Error Correcting Output Codes. We also compare our proposed method with two ensemble techniques (Ekbal and Saha, 2010, 2011) based on evolutionary genetic algorithm. Our analysis shows that the proposed approach can attain superior performance in comparison to the exist-

[1] We use 'ensemble classifier' and 'classifier ensemble' interchangeably

ing methods. The key contributions of this work are two-fold, *viz.* (i). proposal of differential evolution (DE) based feature selection and classifier ensemble methods that can be applied to any classification problem; and (ii). scope of the development of language independent models for NER in a resource-poor scenario.

2 Overview of Differential Evolution

Differential Evolution (DE) (Storn and Price, 1997) is a parallel direct search method which performs search in complex, large and multi-modal landscapes, and provides near-optimal solutions for objective or fitness function of an optimization problem. In DE, the parameters of the search space are encoded in the form of strings called chromosomes. A collection of such strings is called a population denoted by P. It is a collection of NP number of d-dimensional parameter vectors $x_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}]$ $, i = 1, 2, \ldots, NP$ for each generation G. The value of D represents the number of real parameters on which optimization or fitness function depends. The value of NP does not change during the minimization process. The initial vector population is chosen randomly which represents different points in the search space and should cover the entire parameter space. An objective or a fitness function is associated with each string that represents the degree of goodness of the string. Differential evolution generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. This operation is called mutation. The mutated vector's parameters are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector. Parameter mixing is often referred to as "crossover". If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. This last operation is called selection. The process of selection, crossover and mutation continues for a fixed number of generations or till a termination condition is satisfied.

3 Proposed Method for Feature Selection

In this section we firstly formulate the problem of relevant feature selection within the framework of differential evolution, and then present the proposed approach.

3.1 Problem Formulation for Feature Selection

Suppose, the M number of available features for a given classifier are denoted by $F_1, \ldots, F_M$. Let, $\mathscr{A} = \{F_i : i = 1; M\}$. The feature selection problem is then stated as follows:

Find the appropriate subset of features $\mathscr{A}' \subseteq \mathscr{A}$ such that the classifier trained using these features should have optimized some metrics. In this work we optimize the F-measure value which is the combination of both recall and precision.

3.2 Chromosome Representation and Population Initialization

If the total number of features is F, then the length of the chromosome is F. As an example, the encoding of a particular chromosome is represented in Figure 1. Here, $\mathscr{F} = 12$ (i.e., total 12 different features are available). The chromosome represents the use of 7 features for constructing a classifier (first, third, fourth, seventh, tenth, eleventh and twelfth features). The entries of each chromosome are randomly initialized to either 0 or 1. Here, if the i^{th} position of a chromosome is 0 then it represents that i^{th} feature does not participate in constructing the classifier. Else, if it is 1 then the i^{th} feature participates in constructing the classifier.

If the population size is P then all the P number of chromosomes of this population are initialized in the above way.

3.3 Fitness Computation

For the fitness computation, the following steps are executed.

1. Suppose, there are N number of features present in a particular chromosome (i.e., there are total N number of 1's in that chromosome).
2. Construct a classifier with only these N features.
3. Here, initially the training data is divided into 3 parts. The above classifier is trained using 2/3 parts of the training data with the features encoded in that chromosome and evaluated with the remaining 1/3 part.
4. Now, the overall recall, precision and F-measure values of this classifier for the 1/3 training data are calculated.
5. Steps 3 and 4 are repeated 3 times to perform 3-fold cross validation. The average F-measure value is used as the objective function. Thus, the objective function corresponding to a particular chromosome is $f_1 = \textit{F-measure}_{avg}$. The objective is to maximize this objective function using the search capability of DE.

3.4 Mutation

For each target vector $x_{i,G}$; $i = 1,2,3,\ldots,NP$, a mutant vector/donor vector is generated according to

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}), \tag{1}$$

where $r1$, $r2$, $r3$ are the random indices and belong to $\{1,2,\ldots,NP\}$. These are some integer values, mutually different and $F > 0$. The randomly chosen integers $r1$, $r2$ and $r3$ are also chosen to be different from the running index i, so that NP must be greater or equal to four to allow for this condition. F is a real and constant factor 0.5 $[0,1]$ which controls the amplification of the differential variation $(x_{r2,G} - x_{r3,G})$.

3.5 Crossover

In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. This is well-known as the recombination. To this end, the trial vector:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}) \tag{2}$$

is formed, where

$$u_{j,i,G+1} = v_{j,i,G+1} \quad \text{if } (randb(j) \leq CR) \text{ or } j = rnbr(i) \tag{3}$$

$$= x_{j,i,G} \quad \text{if } (randb(j) > CR) \text{ and } j \neq rnbr(i) \tag{4}$$

for $j = 1,2,\ldots,D$,

In Equation 3, $randb(j)$ is the jth evaluation of an uniform random number generator with outcome belongs to $[0,1]$. CR is the crossover constant belongs to $[0,1]$ which has to be determined by the user. Here the value of CR is 0.5. $rnbr(i)$ is a randomly chosen index x belongs to $\{1,2,\ldots,D\}$ which ensures that $u_{i,G+1}$ gets at least one parameter from $v_{i,G+1}$.

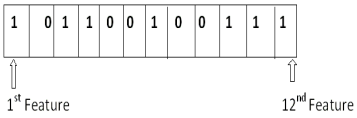

Figure 1: Chromosome representation for DE based feature selection

3.6 Selection

To decide whether or not it should become a member of generation G+1, the trial vector $u_{i,G+1}$ is compared to the target vector $x_{i,G}$ using the greedy criterion. If vector $u_{i,G+1}$ yields a smaller cost function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G+1}$, otherwise, the old value $x_{i,G}$ is retained.

3.7 Termination Condition

In this approach, the processes of mutation, crossover (or, recombination), fitness computation and selection are executed for a maximum number of generations. The best string seen up to the last generation provides the best subset of features. Here the best string contains a set of features. This set is the optimal subset of features for NER problem for a specific language.

4 Proposed Method for Classifier Ensemble

The first step yields a set of solutions on the final best population. There is a single best solution, and along with that the population also contains many other solutions. Some of these solutions are good with respect to recall whereas some are good with respect to precision. All these solutions are equally important from the algorithmic point of view. We generate several different classifiers using these feature combinations. These are then combined using the proposed classifier ensemble creation technique in the second step. The proposed technique determines the best weights of NE classes for classifier combination.

4.1 Problem Formulation

Suppose, the N number of available classifiers be denoted by $C_1, \ldots, C_N$. Let, $\mathscr{A} = \{C_i : i = 1; N\}$ and there are M number of output classes. The proposed classifier ensemble is then stated as follows:

Find the weights of votes V per classifier which will optimize a function $F(V)$. Here, V is a real array of size $N \times M$. $V(i, j)$ denotes the weight of vote of the i^{th} classifier for the j^{th} class. More weight is assigned for that particular class for which the classifier is more confident; whereas the output class for which the classifier is less confident is given less weight. $V(i, j) \in [0, 1]$ denotes the degree of confidence of the i^{th} classifier for the j^{th} class. These weights are used while combining the outputs of classifiers using weighted voting. Here, F is a classification quality measure of the combined classifier. We choose F-measure as the objective function to optimize.

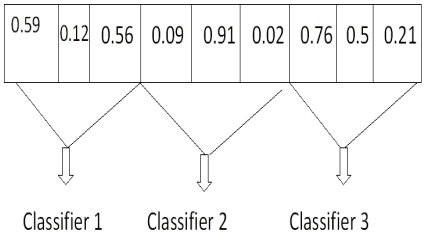

Figure 2: Chromosome Representation for Weighted Vote Based Classifier Ensemble Selection

4.1.1 Chromosome Representation and Population Initialization

If the total number of available classifiers is M and total number of output classes is O, then the length of the chromosome is $M \times O$. Each chromosome encodes the voting weights for possible O classes in each classifier. As an example, the encoding of a particular chromosome is represented in Figure 2. Here, $M = 3$ and $O = 3$ (i.e., total 9 votes can be possible). The chromosome represents the following ensemble:

The weights of votes for 3 different output classes for classifier 1 are 0.59, 0.12 and 0.56, respectively. Similarly, weights of votes for 3 different output classes are 0.09, 0.91 and 0.02, respectively for classifier 2 and 0.76, 0.5 and 0.21, respectively for classifier 3.

We use real encoding, and the entries of each chromosome are randomly initialized to a real value (r) between 0 and 1. Here, $r = \frac{rand()}{RAND\_MAX+1}$. If the population size is P then all the P number of chromosomes of this population are initialized in the above way.

4.1.2 Objective Functions Computation

Initially, the F-measure values of all the available classifiers for each of the output classes are calculated based on the development data. Thereafter, we execute the following steps to compute the objective values.

1. Suppose, there are total M number of classifiers. Let, the overall F-measure values of these M classifiers on the development data be F_i, $i = 1 \ldots M$.

2. We have M classes (each from a different classifier) for each token in the development data. Now for the ensemble classifier, the output class label for each token is determined using the weighted voting of these M classifiers' outputs. The weight of the output class provided by the i^{th} classifier is equal to F_i. The combined score of a particular class for a particular token t is:

$$f(c_i) = \sum F_m \times I(m,i),$$
$$\forall m = 1 \text{ to } M \text{ and } op(t,m) = c_i$$

   Here, $I(m,i)$ is the entry of the chromosome corresponding to the m^{th} classifier and i^{th} class; and $op(t,m)$ denotes the output class provided by the classifier m for the token t. The class receiving the maximum combined score is selected as the joint decision.

Examples: Let us consider the chromosome in Figure 2. Let the three classes be 'PER' (class 1), 'LOC' (class 2) and 'ORG' (class 3). Suppose the final F-measure values of 3 classifiers are 0.8, 0.7 and 0.85, respectively. Then let for a token *'kolkAtA* (Kolkata)' the 3 classifiers predict outputs as follows: Classifier 1: 'PER'; Classifier 2:'LOC'; Classifier 3:'LOC'. Then f('PER')=0.8*0.59=0.472; and f('LOC')=0.91*0.7+0.5*0.85=1.062. Henceforth, all the Bengali glosses are written in ITRANS notation [2]. Thus final class label selected for this particular token is 'LOC' as f('LOC')>f('PER').

3. We use the following objective function: F-measure$_{avg}$. This is maximized using the search capability of DE.

4.1.3 Operators

Other operators of this DE based ensemble approach are similar to those of the DE based feature selection technique described in the previous section.

5 Language Independent Features for NER

The main features for the NER task are identified based on the different possible combinations of available word and tag contexts. We use the following features for constructing the various models of CRF and SVM classifiers. The most important characteristics of our system is that it is both language as well as domain independent in nature. In order to maintain this property the features are *identified and generated without using any deep domain knowledge and/or language specific resources*. Due to this language independent behavior, the features can be easily obtained for almost all the languages.

1. **Context words**: These are the preceding and following tokens surrounding the current token. This is based on the observation that surrounding words carry effective information for the identification of NEs.

2. **Word suffix and prefix**: We use fixed length (say, *n*) word suffixes and prefixes as the features. Actually, these are the character strings stripped either from the rightmost (for suffix) or from the leftmost (for prefix) positions of the words. Morphological analyzers or stemmers could be more effective to extract the meaningful affixes of the wordforms. But there are no such freely available high quality tools for the Indian languages. We also wanted to build our system without using any language dependent resource or tool.

3. **First word**: This is a binary valued feature that checks whether the current token is the first word of the sentence or not. We consider this feature with the observation that the first word of the sentence is most likely a NE. This is the most useful feature for Bengali as NEs generally appear in the first position of the sentence in news-wire data.

4. **Length of the word**: This binary valued feature checks whether the number of characters in a token is less than a predetermined threshold value (here, set to 5). This feature is defined with the observation that very short words are most probably not the NEs.

5. **Infrequent word**: This is a binary valued feature that checks whether the current word appears in the training set very frequently or not. For each of the languages, we compile

[2] http://www.aczone.com/itrans/

the lists of most frequently occurring words for all the three languages. The threshold frequencies depend on the sizes of the datasets. In the present work, we consider the words to be infrequent if they have less than 10 occurrences in Bengali and Hindi datasets, and less than 5 occurrences in the Telugu dataset. A binary valued feature is then defined that fires if and only if the word does not appear in this list. We include this feature as the frequently occurring words are most likely not the NEs.

6. **Digit features**: Several digit features are defined depending upon the presence and/or the number of digits and/or symbols in a token. These features are digitComma (token contains digit and comma), digitPercentage (token contains digit and percentage), digitPeriod (token contains digit and period), digitSlash (token contains digit and slash), digitHyphen (token contains digit and hyphen) and digitFour (token consists of four digits only).

7. **Dynamic NE information**: This is the output label(s) of the previous token(s). The value of this feature is determined dynamically at run time. For CRF we use the bigram feature template that computes all the feature combinations of current and previous tokens.

8. **Content words in surrounding contexts**: At first we extract all unigrams in contexts $w_{i-3}^{i+3} = w_{i-3} \ldots w_{i+3}$ of w_i (crossing sentence boundaries) for the entire training data. Thereafter, we convert tokens to lower case, remove stopwords, numbers and punctuation symbols. We define a feature vector of length 10 using the 10 most frequent content words. Given a classification instance, the feature corresponding to token t is set to 1 iff the context w_{i-3}^{i+3} of w_i contains t. In order to compute this feature for the test set we first pass it through a NER system (Ekbal and Bandyopadhyay, 2009a) to extract the NE information of each token.

6 Datasets, Experimental Setup and Evaluation Results

In this section, we report the datasets used for the experiment, experimental setup and evaluation results with necessary discussions.

6.1 Datasets for NER

Indian languages are resource-constrained in nature. For NER, we use a Bengali news corpus (Ekbal and Bandyopadhyay, 2008), developed from the archive of a leading Bengali newspaper available in the web. One of the authors manually annotated a portion of this corpus containing approximately 250K wordforms with a coarse-grained NE tagset of four tags namely, PER (*Person name*), LOC (*Location name*), ORG (*Organization name*) and MISC (*Miscellaneous name*). The *Miscellaneous name* includes date, time, number, percentages, monetary expressions and measurement expressions. The data is collected mostly from the *National*, *States*, *Sports* domains and the various sub-domains of *District* of the particular newspaper. We also use the IJCNLP-08 NER on South and South East Asian Languages (NERSSEAL)[3] Shared Task data of around 100K wordforms that were originally annotated with a fine-grained tagset of twelve tags. This data is mostly from the agriculture and scientific domains. For Hindi and Telugu, we use approximately 502,913 and 64,026 tokens obtained from the NERSSEAL shared

[3] http://ltrc.iiit.ac.in/ner-ssea-08

Table 1: Named entity tagset for Indian languages (IJCNLP-08 NERSSEAL Shared Task Tagset)

NE Tag	Meaning	Example
NEP	Person name	*sachIna*/NEP, *sachIna ramesha tenDUlkara* / NEP
NEL	Location name	*kolkAtA*/NEL, *mahatmA gAndhi roDa* / NEL
NEO	Organization name	*yadabpUra bishVbidyAlYa*/NEO, *bhAbA eytOmika risArcha sentAra* / NEO
NED	Designation	*cheYArmAn*/NED, *sA.msada*/NED
NEA	Abbreviation	*bi e*/NEA, *ci em di a*/NEA, *bi je pi*/NEA, *Ai.bi.em*/ NEA
NEB	Brand	*fYAntA*/NEB
NETP	Title-person	*shrImAna*/NED, *shrI*/NED, *shrImati*/NED
NETO	Title-object	*AmericAn biUti*/NETO
NEN	Number	*10*/NEN, *dasha*/NEN
NEM	Measure	*tina dina*/NEM, *p.NAch keji*/NEM
NETE	Terms	*hidena markbha madela*/NETE, *kemikYAla riYYAkchYAna*/NETE
NETI	Time	*10 i mAgha 1402* / NETI, *10 ema*/NETI

Table 2: Statistics of the datasets

Language	# tokens in training	#NEs in training	#tokens in test	#NEs in test
Bengali	312,947	37,009	37,053	4,413
Hindi	444,231	43,021	58,682	3,005
Telugu	61,684	5,004	2,342	128

task. The underlying reason to adopt the finer NE tagset in the shared task is to use the NER system in various NLP applications, particularly in machine translation. The IJCNLP-08 NERSSEAL shared task tagset is shown in Table 1. One important aspect of the shared task was to identify and classify the maximal NEs as well as the nested NEs, i.e. the constituents of a larger NE. In the present work, we consider only the tags that denote person names (NEP), location names (NEL), organization names (NEO), number expressions (NEN), time expressions (NETI) and measurement expressions (NEM). The NEN, NETI and NEM tags are mapped to the *Miscellaneous name* tag that denotes miscellaneous entities. Other tags of the shared task are mapped to the 'other-than-NE' category denoted by 'O'. Statistics of the data sets in terms of the number of tokens in the training set, number of tokens in test set and number of NEs in training and test sets are given in Table 2.

6.2 Experimental Setup

The parameters of the proposed algorithm are selected by conducting a thorough sensitivity analysis on the development data. A part of the training dataset is used as the development set.

The parameters of DE are determined based on the development sets. The parameters of DE technique are as follows: population size = 100, CR (probability of crossover)=0.5, number of generations = 50 and F (mutation factor) = 0.5. We define the following *baseline* ensemble techniques:

- *Baseline 1*- This baseline is constructed by considering the following feature combination: Context of previous two and next two tokens along with all the features listed in Section 5.
- *Baseline 2*- This baseline is trained using the following feature combination: Context of previous two and next two tokens along with all the features listed in Section 5.
- *Baseline 3*- In this *baseline* model, all the individual classifiers selected from the first stage of the algorithm are combined together into a final system based on the majority voting of the output class labels. If all the outputs differ then anyone is selected randomly.
- *Baseline 4*- All the classifiers selected from the first stage of the algorithm are combined together with the help of a weighted voting approach. In each classifier, weight is calculated based on the F-measure value of the 3-fold cross validation on the training data. The final output label is selected based on the highest weighted vote.

In this work, we use CRF and SVM as the base classifiers. CRF (Lafferty et al., 2001b) considers a global exponential model. It has the freedom to include arbitrary features and the ability of feature induction to automatically construct the most useful feature combinations. For constructing CRF based classifiers, we use the C^{++} based CRF^{++} package [4], a simple, customizable, and open source implementation of CRF for segmenting or labeling sequential data. The SVM technique (Joachims, 1999; Vapnik, 1995) takes a strategy that maximizes the margin between the critical samples and the separating hyperplane. In particular, SVMs achieve high generalization even with training data of a very high dimension. Moreover, with the use of *kernel function*, SVMs can handle non-linear feature spaces, and carry out the training considering combinations of more than one feature. For constructing SVM based classifiers, we use YamCha[5] toolkit, an SVM based tool for detecting classes in documents and formulating the NER task as a sequential labeling problem. Here, we use both the *one-vs-rest* and *pairwise multi-class decision* methods, and the *polynomial kernel function* of degree 2.

At first DE based feature selection technique is used to determine the most relevant set of features for CRF and SVM. It produces a set of solutions that also includes the best solution. Based on the F-measure values we sort these solutions in the descending order. We select in total 14 solutions, 7 each for CRF and SVM. These effective classifiers are then used to construct the ensemble in the second stage. The performance of these (7+7=14) classifiers (with their feature combinations) are reported in Table 3 for Bengali. The CRF-based model exhibits the best performance (with respect to the overall F-measure value) yielding the recall, precision and F-measure values of 88.05%, 86.58% and 87.31%, respectively. Our first baseline which is constructed by including all the features in CRF model yields the recall, precision and F-measure values of 87.75%, 85.27% and 86.49%, respectively. Thus the DE based feature selection technique improves the F-measure value by 0.82 points. With SVM, the feature selection approach

[4] http://crfpp.sourceforge.net
[5] http://chasen-org/ taku/software/yamcha/

yields the recall, precision and F-measure values of 86.25%, 85.33% and 85.79%, respectively. This is an increment of 0.89 points over the second baseline where SVM is trained with all the available features. Overall evaluation results of our proposed two stage approach along with the best individual classifier and four different *baseline* ensembles are reported in Table 4. The proposed two-stage algorithm shows the recall, precision and F-measure values of 89.79%, 88.01% and 88.89%, respectively. This is an improvement of 1.58 points over the first stage, i.e. feature selection technique. It also demonstrates the overall performance increments of 1.62 and 0.78 percentage F-measure points over the third and fourth baselines, respectively.

Thereafter the proposed techniques are applied to Hindi and Telugu data sets. For both of these languages, we select 14 solutions, 7 each from CRF and SVM from the first stage, i.e. DE based feature selection approach. For Hindi, the CRF-based feature selection model exhibits the best performance with the recall, precision and F-measure values of 87.79%, 86.05% and 86.91%, respectively. Overall evaluation results of our proposed two stage approach along with the best individual classifier and four different *baseline* ensembles are reported in Table 4. The DE based feature selection approach shows an improvement of 0.59 points over the first baseline which is constructed by considering all the features into the model. The ensemble based approach attains the recall, precision and F-measure values of 88.88%, 87.35% and 88.09%, respectively. This is an improvement of 1.18 points over the feature selection approach. We clearly observe the increments of 1.28 and 1.25 points over the majority voted (i.e. Baseline-3) and weighted voted ensemble (i.e. Baseline-4) methods, respectively. For Telugu, the feature selection approach shows the recall, precision and F-measure values of 76.05%, 72.05% and 74.00%, respectively. These are the increments of 2.43 and 4.34 F-measure points over the first and second baseline, respectively. Overall evaluation results are reported in Table 4. Finally, the ensemble approach yields the overall recall, precision and F-measure values of 78.58%, 74.78% and 76.63%, respectively. This is superior to the feature selection approach as well as the other two ensemble methods. In order to show that the proposed two stage approach really outperforms the best individual classifier (i.e. feature selection approach) and four *baseline* ensembles, statistical analysis of variance (ANOVA) (Anderson and Scolve, 1978) is performed, when each is executed ten times. ANOVA tests show that for all the languages differences in mean recall, precision and F-measure are statistically significant as p value is less than 0.05 in each of the cases. We also compare the performance of our proposed approach with three state-of-the-art ensemble methods, namely stacking (Wolpert, 1992), AdaBoost (Freund and Schapire, 1995), ECOC (Error correcting Output Codes) (Dietterich and Bakiri, 1995) and two genetic algorithm (GA) based ensemble methods (Ekbal and Saha, 2010, 2011). In stacking, we used CRF as a meta-classifier. For ECOC, initially we generate binary classifiers for each NE class. Binary classifiers are generated using CRF and SVM. Thereafter, ECOC (Dietterich and Bakiri, 1995) method is applied to solve the multi-class problem. The code matrix is generated exhaustively and minimum Hamming distance based method is applied to determine the appropriate class label of each test instance. In case of AdaBoost we used CRF as a weak classifier. The GA based ensemble methods are executed on the classifiers obtained from the first stage of our proposed technique. We used the same parameter settings as used in (Ekbal and Saha, 2011) and (Ekbal and Saha, 2010). The techniques proposed in these papers were evaluated with set of features presented in this paper. Comparative evaluation results are shown in Table 5 for Bengali. In (Ekbal and Saha, 2010) authors have proposed a GA based simple classifier ensemble technique. The proper weights of votes were not determined like (Ekbal and Saha, 2011). Comparison shows that our proposed algorithm achieves superior performance compared to the previous two approaches. But approach proposed in (Ekbal and Saha, 2010) obtains better

Table 3: Evaluation results with various feature combinations for the CRF and SVM based classifiers for Bengali. Here, the following abbreviations are used: 'CW':Context words, 'PS': Size of the prefix, 'SS': Size of the suffix, 'WL': Word length, 'IW': Infrequent word, 'FW':First word, 'TD': 'Two-Digit', 'FD': 'Four-Digit', 'DH': 'Digit-Hyphen', 'DS': 'Digit-Slash', 'DD': 'Digit-Dot', 'DP': 'Digit-Percentage', 'DC': 'Digit-Comma', 'Sem': Content words, 'P', 'C' and 'N': Previous, current and next tokens, '$-i, j$': Words spanning from the i^{th} left position to the j^{th} right position, Current token is at 0^{th} position, 'X': Denotes the presence of the corresponding feature (we report in percentages), 'r': recall, 'p': precision, 'F': F-measure

Classifier	CW	TD	FD	DH	DS	DD	DP	DC	IW	WL	SS	PS	Sem	FW	p	r	F
CRF_1	-2,2	X	X	X	X	X	-	X	-	X	X	X	X	X	88.05	86.58	87.31
CRF_2	-3,3	X	X	X	X	X	X	X	-	X	X	X	X	X	87.98	86.53	87.25
CRF_3	-3,3	X	X	-	X	X	-	X	-	X	X	X	X	X	87.94	86.51	87.22
CRF_4	-2,2	X	-	X	X	X	X	X	-	X	X	X	X	X	87.98	86.57	87.26
CRF_5	-2,2	X	X	-	X	X	X	-	-	X	X	X	X	-	88.00	86.45	87.24
CRF_6	-2,2	X	-	X	X	X	X	X	-	X	X	X	X	X	87.88	86.56	87.21
CRF_7	-2,2	X	-	X	X	X	X	X	-	X	X	X	X	-	87.86	86.53	87.19
SVM_1	$-1,1$	X	-	X	-	X	-	X	X	-	X	X	X	X	86.25	85.33	85.79
SVM_2	$-2,2$	X	-	X	X	X	X	X	X	X	X	X	X	X	86.17	85.31	85.74
SVM_3	$-2,2$	X	X	X	-	X	X	-	-	X		X	X	X	85.61	84.18	84.89
SVM_4	$-2,2$	X	X	X	-	X	X	-	-	X		X	X	X	85.55	84.02	84.78
SVM_5	$-2,2$	-	-	X	X	X	X	X	-	X		X	X	X	85.10	84.00	84.54
SVM_6	$-2,2$	-	-	-	-	X	X	X	-	X		X	X	X	85.09	83.90	84.49
SVM_7	$-2,2$	X	X	-	X	X	X	X	X	X	X	X	X	-	85.19	83.79	84.49

Table 4: Overall results (we report in percentages) for Bengali, Hindi and Telugu; here 'r': recall, 'p': precision, 'F': F-measure, ME: Method, BIC: Best individual classifier, B1: Baseline-1, B2: Baseline-2, B3: Baseline-3, B4:Baseline- 4, TA: proposed two stage approach.

ME.	Bengali			Hindi			Telugu		
	p	r	F	p	r	F	p	r	F
BIC	88.05	86.58	87.31	87.79	86.05	86.91	76.05	72.05	74.00
B1	87.75	85.27	86.49	87.21	85.45	86.32	72.29	70.87	71.57
B2	86.01	83.82	84.90	87.34	83.79	84.52	70.52	68.82	69.66
B3	88.03	86.53	87.27	87.66	85.99	86.81	74.78	72.19	73.47
B4	88.69	87.54	88.11	87.58	86.12	86.84	75.57	72.05	73.77
TA	89.79	88.01	88.89	88.88	87.35	88.09	78.58	74.78	76.63

F-measure values for some data sets than the proposed method. This is due to the use of many language specific features extracted from Part-of-Speech (PoS) tagger and several gazetteers. Experimental analysis suggests that errors are mostly due to boundary detection and the conflicts between the organization and location classes.

Results show that our proposed two-stage DE based technique performs better than the best individual classifier, four different baseline ensemble techniques and some other existing state-of-the-art ensemble methods (c.f. Table 4 and Table 5). It is already established in machine learning literature that proper ensemble of classifiers should always perform better in comparison to the existing base classifiers. The first stage of our algorithm that concerns with the relevant feature selection performs better than the two baseline models, where classifiers are trained with all the available features. This clearly shows the necessity of determining appropriate feature set for the problem. Our proposed DE based ensemble shows significant performance gains over all the baselines. In the third and fourth baselines, we combined the

Table 5: Comparative Evaluation Results (we report in percentages)

Classification Scheme	recall	precision	F-measure
Stacking	87.88	86.24	87.05
ECOC	87.91	86.23	87.06
AdaBoost	88.53	86.84	87.68
GA based ensemble (Ekbal and Saha, 2010)	88.01	85.89	86.93
GA based ensemble (Ekbal and Saha, 2011)	88.72	87.06	87.88
DE based Approach	89.79	88.01	88.89

classifiers blindly. As a result, in some cases, the combined model even shows inferior performance compared to the best individual classifier(s) that correspond to our feature selection method. The effectiveness of DE in determining proper weights of voting is also the another reason of showing better performance. The nature of performance supports our underlying hypothesis that determining appropriate weights of voting for each class in each classifier is very crucial. Results show that the proposed algorithm performs best for Bengali followed by Hindi and Telugu. The possible reasons could be (i). the ratios of positive (NEs) and negative examples (non-NEs) in the respective training data, i.e. 1:9 (Bengali), 1:9.33 (Hindi). and 1:13 (Telugu), (ii). agglutinative nature of Telugu that may possibly require some language specific rules and (iii). less amount of training data for Telugu compared to others. This may be due to the fact that all the datasets are highly imbalanced, and hence greatly biased towards the negative categories. Thus, there are not enough NE instances that could be more effective for NE identification. This observation leads to the path of investigating appropriate sampling strategy in order to make the ratios of positive and negative examples more balanced.

Conclusion

In this paper we have proposed a two stage differential evolution based technique for NER in three different languages. Here at first DE based technique is used to select relevant sets of features for different classifiers. We used CRF and SVM as the underlying classification methods. Thereafter effective classifiers were selected based on the F-measure scores, and combined using a DE based classifier ensemble technique. Evaluation results show the encouraging performance in all the settings. Comparisons with the conventional baselines and some state-of-the-art ensemble methods show the superiority of our proposed technique. The key contributions of this work are are two-fold, *viz.* (i). proposal of differential evolution (DE) based feature selection and classifier ensemble methods that can be applied to any classification problem; and (ii). scope of the development of language independent NER systems in a resource-poor scenario. In future we will study the effects of various language dependent features that can be extracted from morphological analyzers and gazetteers.

Overall evaluation results suggest that there is still the room for further improvement. In this work, we have considered the problem of feature selection and classifier ensemble as a single objective optimization problem. In future, we will develop some multi-objective DE based techniques to solve these feature selection and classifier ensemble methods.

References

Anderson, T. W. and Scolve, S. (1978). *Introduction to the Statistical Analysis of Data*. Houghton Mifflin.

Bikel, D. M., Schwartz, R. L., and Weischedel, R. M. (1999). An Algorithm that Learns What's in a Name. *Machine Learning*, 34(1-3):211–231.

Borthwick, A. (1999). *Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York University.

Chinchor, N. (1995). MUC-6 Named Entity Task Definition (Version 2.1). In *MUC-6*. Maryland.

Chinchor, N. (1998). MUC-7 Named Entity Task Definition (Version 3.5). In *MUC-7*. Fairfax.

Cunningham, H. (2002). GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254.

Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

Ekbal, A. and Bandyopadhyay, S. (2007). Lexical Pattern Learning from Corpus Data for Named Entity Recognition. In *Proceedings of the 5th International Conference on Natural Language Processing (ICON)*, pages 123–128, India.

Ekbal, A. and Bandyopadhyay, S. (2008). A Web-based Bengali News Corpus for Named Entity Recognition. *Language Resources and Evaluation Journal*, 42(2):173–182.

Ekbal, A. and Bandyopadhyay, S. (2009a). A Conditional Random Field Approach for Named Entity Recognition in Bengali and Hindi. *Linguistic Issues in Language Technology (LiLT)*, 2(1):1–44.

Ekbal, A. and Bandyopadhyay, S. (2009b). Voted NER System using Appropriate Unlabeled Data. *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), ACL-IJCNLP 2009*, pages 202–210.

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

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

Ekbal, A. and Saha, S. (2010). Classifier ensemble selection using genetic algorithm for named entity recognition. *Research on Language and Computation*, 8:73–99.

Ekbal, A. and Saha, S. (2011). Weighted vote-based classifier ensemble for named entity recognition: A genetic algorithm-based approach. *ACM Trans. Asian Lang. Inf. Process.*, 10(2).

Freund, Y. and Schapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, Taipei, Taiwan.

Joachims, T. (1999). *Making Large Scale SVM Learning Practical*, pages 169–184. MIT Press, Cambridge, MA, USA.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001a). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001b). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Li, W. and McCallum, A. (2004). Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. *ACM Transactions on Asian Languages Information Processing*, 2(3):290–294.

Patel, A., Ramakrishnan, G., and Bhattacharya, P. (2009). Relational Learning Assisted Construction of Rule Base for Indian Language NER. In *Proceedings of ICON 2009: 7th International Conference on Natural Language Processing*, India.

Sekine, S. (1998). Description of the Japanese NE System used for MET-2. In *MUC-7*, Fairfax, Virginia.

Shishtla, P. M., Pingali, P., and Varma, V. (2008). A Character n-gram Based Approach for Improved Recall in Indian Language NER. In *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages*, pages 101–108.

Srikanth, P. and Murthy, K. N. (2008). Named Entity Recognition for Telugu. In *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages*, pages 41–50.

Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Vijayakrishna, R. and Sobha, L. (2008). Domain Focused Named Entity Recognizer for Tamil using Conditional Random Fields. In *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages*, pages 93–100.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Yamada, H., Kudo, T., and Matsumoto, Y. (2001). Japanese Named Entity Extraction using Support Vector Machine. *In Transactions of IPSJ*, 43(1):44–53.