Data & Knowledge Engineering 85 (2013) 15–39

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition

Sriparna Saha [1], Asif Ekbal *,[1]

Department of Computer Science and Engineering, Indian Institute of Technology Patna, Patna, Bihar, India

ARTICLE INFO

Article history:
Received 30 December 2010
Received in revised form 20 July 2011
Accepted 25 June 2012
Available online 20 July 2012

Keywords:
Named Entity Recognition (NER)
Weighted Vote based Ensemble
Single Objective Optimization
Multiobjective Optimization (MOO)
Genetic Algorithm (GA)
Naive Bayes
Decision Tree (DT)
Memory Based Learner (MBL)
Hidden Markov Model (HMM)
Maximum Entropy(ME)
Conditional Random Field (CRF)
Support Vector Machine (SVM)

ABSTRACT

In this paper, we pose the classifier ensemble problem under single and multiobjective optimization frameworks, and evaluate it for Named Entity Recognition (NER), an important step in almost all Natural Language Processing (NLP) application areas. We propose the solutions to two different versions of the ensemble problem for each of the optimization frameworks.

We hypothesize that the reliability of predictions of each classifier differs among the various output classes. Thus, in an ensemble system it is necessary to find out either the eligible classes for which a classifier is most suitable to vote (i.e., binary vote based ensemble) or to quantify the amount of voting for each class in a particular classifier (i.e., real vote based ensemble). We use seven diverse classifiers, namely Naive Bayes, Decision Tree (DT), Memory Based Learner (MBL), Hidden Markov Model (HMM), Maximum Entropy (ME), Conditional Random Field (CRF) and Support Vector Machine (SVM) to build a number of models depending upon the various representations of the available features that are identified and selected mostly without using any domain knowledge and/or language specific resources. The proposed technique is evaluated for three resource-constrained languages, namely Bengali, Hindi and Telugu. Results using multiobjective optimization (MOO) based technique yield the overall recall, precision and F-measure values of 94.21%, 94.72% and 94.74%, respectively for Bengali, 99.07%, 90.63% and 94.66%, respectively for Hindi and 82.79%, 95.18% and 88.55%, respectively for Telugu. Results for all the languages show that the proposed MOO based classifier ensemble with real voting attains the performance level which is superior to all the individual classifiers, three *baseline* ensembles and the corresponding single objective based ensemble.

1. Introduction

Named Entity Recognition (NER) is a well-established task that has immense importance in many Natural Language Processing (NLP) application areas such as Information Retrieval [1], Information Extraction [2], Machine Translation [3], Question Answering [4], Automatic Summarization, [5] etc. The objective of NER is to identify and classify every word/term in a document into some predefined categories like person name, location name, organization name, miscellaneous name (date, time, percentage and monetary expressions, etc.) and "none-of-the-above".

The main approaches to NER can be grouped into three main categories, namely rule-based, machine learning based and hybrid approach. Rule based approaches [6–9] focus on extracting names using a number of handcrafted rules. Generally, these systems consist of a set of patterns using grammatical (e.g., part of speech), syntactic (e.g., word precedence) and orthographic features (e.g., capitalization) in combination with dictionaries. These kinds of systems have better results for restricted domains and are capable of detecting complex entities that are difficult with learning models. However, rule based systems lack the ability

* Corresponding author.
E-mail addresses: sriparna.saha@gmail.com, sriparna@iitp.ac.in (S. Saha), asif.ekbal@gmail.com, asif@iitp.ac.in (A. Ekbal).
[1] Authors equally contributed to the paper.

0169-023X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2012.06.003

of portability and robustness, and furthermore the high cost of maintenance of rules increases even when the data is slightly changed. These types of systems are often domain dependent, language specific and do not necessarily adapt well to new domains and languages.

Researchers, nowadays, are popularly using machine learning approaches for NER because these are easily trainable, adaptable to different domains and languages as well as their maintenance being also less expensive. The machine learning techniques can be grouped into the following three categories, namely supervised, semi-supervised and unsupervised. The idea of supervised learning is to study the features of positive and negative examples of named entities (NEs) over a large collection of annotated documents and design rules that capture instances of a given type. Some of the representative supervised machine learning approaches used in NER are Hidden Markov Model (HMM) [10,11], Maximum Entropy (ME) [12,13], Decision Tree [14,15], Conditional Random Field (CRF) [16,17] and Support Vector Machines (SVMs) [18]. The main shortcoming of supervised learning is the requirement of a large annotated corpus, which is very difficult to obtain for many less computerized languages like most of the Indian languages. In hybrid systems [8,19,20], the goal is to combine rule based and machine learning based methods, and develop new methods using strongest points from each one.

Besides the above mentioned works, there are handsome amounts of other existing works too. Majority of the languages covered include English, most of the European languages and some of the Asian languages like Chinese, Japanese and Korean. India is a multilingual country with great linguistic and cultural diversities. However, the works related to NER in Indian languages are still in the nascent stage due to the potential facts such as (i). lack of capitalization information, which is an important clue for NE identification in English and some other languages; (ii). diversity of Indian names and their appearances in the dictionary as common nouns; (iii). free word order nature of the Indian languages; (iv). resource-constrained environment, i.e. non-availability of corpus, annotated corpus, name dictionaries, morphological analyzers, part of speech (POS) taggers, etc. in the required measure. There have been some initiatives [21–25] for NER in Indian languages in the last few years.

Classifier ensemble[2] interchangeably is a popular direction of machine learning. In this paper, we assume that rather than selecting the best-fitting feature set, constructing an ensemble using several NER systems, where each one is based on different feature representations and/or different learning strategies could be more effective in order to achieve better performance. Literature shows the success of classifier combination technique in many language technology tasks including NER [26,22].

The main idea behind classifier ensemble is that this is often much more accurate than the individual classifiers that make them up. The ensemble has greater generalization accuracy that depends on the diversity of each individual classifier as well as on their individual performance. Thus, appropriate classifier selection for constructing an ensemble remains a difficult problem. Moreover, all classifiers are not good to detect all types of output classes. For example, some classifiers are good at detecting *Person names* whereas some are good at detecting *Location names*. Thus, in a voted system, a particular classifier should only be allowed to vote for that output class for which it performs well. Therefore, selection of appropriate votes per classifier is an important research question. In this paper we first pose this classifier ensemble (i.e., the first version[3]) as an optimization problem and provide two different kinds of solutions to it based on single and multiobjective optimization techniques. Here, we determine whether a particular classifier is allowed to vote for a particular class or not.

But, rather than completely disallowing some output classes for voting, it could be more effective if a particular classifier is allowed to vote for all the classes with some confidence values. The underlying assumption is that, in the case of weighted voting, weights of voting should vary among the various output classes in each classifier. The weight should be high for that particular class for which the classifier is more reliable. Otherwise, weight should be low for that class for which the classifier is not very reliable. So, it is very crucial to select the appropriate weights of votes for all the classes in each classifier. In the second version,[4] we make an attempt to quantify the weights of voting for each class in each classifier. For this particular problem, we also present two different solutions based on single and multiobjective optimization techniques.

The single objective optimization (SOO) technique can only optimize a single classification quality measure, e.g. recall, precision or F-measure at a time. But, sometimes a single measure cannot capture the quality of a good ensemble reliably. A good weighted vote based ensemble should have all of its parameters optimized simultaneously. In order to achieve this, we use multiobjective optimization (MOO) [27] that is capable of simultaneously optimizing more than one classification quality measure.

The proposed approaches are evaluated for NER in three resource-constrained Indian languages such as Bengali, Hindi and Telugu. In terms of native speakers, Bengali is the *fifth* most popular language in the world, *second* in India and the *national* language in Bangladesh. Hindi is the *third* most spoken language in the world and the *national* language of India. Telugu is the other popular language and predominantly spoken in the *southern* part of India. The base classifiers in our proposed ensemble system are based on the variety of classification methodologies like Naive Bayes [28], Decision Tree (DT) [29], Memory Based Learner (MBL) [30], Hidden Markov Model (HMM) [11], Maximum Entropy (ME) [12], Conditional Random Field (CRF) [17] and Support Vector Machine (SVM) [31]. A number of models based on these classifiers are generated depending on the different combinations of available features and/or feature templates. One most important characteristic of the features is that these are *identified and selected largely without using any domain dependent knowledge and/or language specific resources*. These features are language independent in nature, and can be easily obtained for almost all the languages with very little effort. Thereafter, the single and multiobjective based methods are applied to construct an ensemble system. Instead of searching for the best performing individual classifiers, our main focus is either to investigate the eligible set of classes for which a classifier is most

[2] We use 'ensemble classifier' and 'classifier ensemble'.
[3] First version is named as 'binary vote based classifier ensemble'.
[4] The second version is named the 'real vote based classifier ensemble' or 'weighted voted based ensemble'.

suited to vote or to determine appropriate weights of voting for all the classes in each classifier. The SOO techniques are based on genetic algorithm (GA), and the MOO based approaches are based on a popular multiobjective technique named Non-dominated Sorting Genetic Algorithm (NSGA)-II [32]. For voting combination determination (i.e., first problem), binary encoding is used. But in the case of weighted voting combination determination (i.e., second problem), real encoding is used. Weights of voting for the output classes in each classifier are encoded in a chromosome. Mutation and crossover operators are modified accordingly. In the case of single objective optimization, a new mutation operator is used to handle real encoding. Adaptive mutation and crossover operators are used to accelerate the convergence of GA. For both the approaches, we also use elitism.

We use the NE-annotated 250 K word forms of the Bengali news corpus [33]. This was annotated with a coarse-grained NE tagset of four tags namely, PER (*Person name*), LOC (*Location name*), ORG (*Organization name*) and MISC (*Miscellaneous name*). We also use the IJCNLP-08 NER on South and South East Asian Languages (NERSSEAL)[5] shared task data of around 100 K word forms that were originally annotated with a fine-grained tagset of twelve tags. For Hindi and Telugu, we use the datasets obtained from the NERSSEAL shared task. Evaluation results show that determining voting weights for all the classes per classifier (i.e., real vote based classifier ensemble) is more fruitful compared to finding out the voting combination selection (i.e., binary vote based ensemble). Evaluation results yield the recall, precision and F-measure values of 92.81%, 92.92% and 92.86%, respectively for Bengali, 90.92%, 92.56% and 91.73%, respectively for Hindi and 80.82%, 93.26% and 86.60%, respectively for Telugu under the single objective real vote based framework. Results using MOO under the real vote based framework show the recall, precision and F-measure values of 94.21%, 94.72% and 94.74%, respectively for Bengali, 99.07%, 90.63% and 94.66%, respectively for Hindi and 82.79%, 95.18% and 88.55%, respectively for Telugu. Evaluation also shows that the real vote based classifier ensemble identified by the MOO based approach performs superior to all the individual classifiers, three standard *baseline* ensembles and the single objective optimization based classifier ensemble techniques for all the languages.

The related works on ensembles for NER are reported in [34–37]. In [34], a GA based classifier ensemble selection technique was developed. This approach determines only a subset of classifiers that can form the final classifier ensemble. But, the appropriate voting weights for all classes per classifier were not computed. In [35] a GA based technique has been proposed for weighted vote based classifier ensemble selection. This was able to determine proper weights of voting for all classes. In [36] a multiobjective optimization based ensemble is developed that can identify a subset of classifiers that forms the final classifier ensemble. A simulated annealing based MOO technique for classifier ensemble is proposed in [37]. Several different versions of the objective functions are exploited.

The main contributions of the present paper are listed below:

1. We use Naive Bayes, Decision Tree, Memory Based Learner, HMM, ME, CRF and SVM as the base classifiers. However, the proposed methods will work for any set of classifiers, i.e. either homogeneous or heterogeneous or a combination of both. The proposed technique is a very general approach and its performance may further improve depending upon the choice and/or the number of classifiers as well as the use of more complex features.
2. Single and multiobjective genetic algorithm based techniques are developed for determining either appropriate voting combination per classifier or best weights of votes to form a classifier ensemble. We tried to establish that such ensemble is capable to increase the classification quality by a reasonable margin compared to the conventional ensemble methods. Note that binary vote based classifier ensemble generation is a new contribution. The real-vote based classifier ensemble selection using single objective optimization was developed in [35]. However in the present paper a multiobjective optimization based ensemble is developed.
3. The features are mostly identified without any domain-dependent knowledge and/or language specific resources. Thus, the proposed techniques can be replicated for any resource-poor language very easily. Here, the approaches are evaluated for three relatively resource-poor languages like Bengali, Hindi and Telugu.
4. The proposed frameworks are applicable for any type of classification problems like NER, Part-of-Speech (POS) tagging, question-answering, etc. To the best of our knowledge, the use of single and multiobjective optimizations for solving these types of classifier ensemble problems is a novel contribution. The use of these techniques for solving any kind of problem in the domain of NLP, especially in NER is also new.
5. Note that our work proposes a novel way of combining the available classifiers. Thus, the performance of the existing works (e.g. [22,26], etc.) can be further improved with our proposed frameworks.
6. Another important motivation of MOO based techniques is to provide the users a set of alternative solutions with high precision values or solutions with high recall values or solutions with moderate recall and precision values. Depending upon the nature of problems or the requirement of the users, appropriate solutions can be selected.

The rest of the paper is structured as follows. A brief introduction to the optimization techniques used in our tasks is reported in Section 2. Section 3 formulates the classifier ensemble problem under both single and multiobjective optimization techniques. Section 4 presents our proposed approach. The problems of NER under the various machine learning frameworks are briefly described in Section 5. Section 6 depicts the set of NE features that we use for NER in three leading Indian languages, namely Bengali, Hindi and Telugu. Detailed evaluation results are reported in Section 7. Finally, we conclude with the future work roadmaps in Section 8.

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

2. Optimization techniques

In this section, we briefly describe the optimization techniques used in our NER tasks. The single objective optimization techniques are based on genetic algorithm (GA) [38], and the multiobjective optimization (MOO) based approaches are based on a popular multiobjective technique named Non-dominated Sorting Genetic Algorithm (NSGA)-II [32].

2.1. Overview of genetic algorithm

Genetic algorithms [38] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, having a large amount of implicit parallelism. GAs perform searches in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. In GAs, the parameters of the search space are encoded in the form of strings called *chromosomes*. A collection of such strings is called a *population*. Initially, a random population is created, which represents different points in the search space. An *objective* or *fitness* function is associated with each string that represents the degree of *goodness* of the string. Based on the principle of survival of the fittest, a few of the strings are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like *crossover* and *mutation* are applied on these strings to yield a new generation of strings. The processes of selection, crossover and mutation continue for a fixed number of generations or till a termination condition is satisfied. The basic steps of conventional GA are shown in Fig. 1.

2.2. Multiobjective algorithms

The multiobjective optimization (MOO) can be formally stated as follows [27]. Find the vectors $\overline{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ of decision variables that simultaneously optimize the M objective values $\{f_1(\overline{x}), f_2(\overline{x}), \ldots, f_M(\overline{x})\}$, while satisfying the constraints, if any.

An important concept of MOO is that of domination. In the context of a maximization problem, a solution $\overline{x_i}$ is said to dominate $\overline{x_j}$ if $\forall k \in 1, 2, \ldots, M,\ f_k(\overline{x_i}) \geq f_k(\overline{x_j})$ and $\exists k \in 1, 2, \ldots, M,$ such that $f_k(\overline{x_i}) > f_k(\overline{x_j})$.

Among a set of solutions P, the nondominated set of solutions P' are those that are not dominated by any member of the set P. The nondominated set of the entire search space S is the globally Pareto-optimal set. In general, a MOO algorithm usually admits a set of solutions that are not dominated by any solution encountered by it.

2.2.1. Nondominated Sorting Genetic Algorithm-II (NSGA-II)

Genetic algorithms (GAs) are known to be more effective than classical methods such as weighted metrics, goal programming [27], for solving multiobjective problems primarily because of their population-based nature. NSGA-II [32] is widely used in this regard, where initially a random parent population P_0 is created and the population is sorted based on the *partial order* defined by the non-domination relation-a sequence of nondominated fronts is obtained. Each solution of the population is assigned a fitness which is equal to its non-domination level in the partial order. A child population Q_0 of size N is created from the parent population P_0 by using binary tournament selection, recombination, and mutation operators. According to this algorithm, in the t^{th} iteration, a combined population $R_t = P_t + Q_t$ is formed. The size of R_t is $2N$. All the solutions of R_t are sorted according to non-domination. If the total number of solutions belonging to the best nondominated set F_1 is smaller than N, then F_1 is totally included in $P_{(t+1)}$. The remaining members of the population $P_{(t+1)}$ are chosen from subsequent nondominated fronts in the order of their ranking. To choose exactly N solutions, the solutions of the last included front are sorted using the crowded comparison operator [32] and the best among them (i.e., those with lower crowding distance) are selected to fill in the available slots in $P_{(t+1)}$. The new population $P_{(t+1)}$ is then used for selection, crossover and mutation to create a population $Q_{(t+1)}$ of size N. The pseudocode of NSGA-II is provided in Fig. 2.

```
Begin
  1. t = 0
  2. initialize population P(t)  /* Popsize = |P| */
  3. for i = 1 to Popsize
       compute fitness P(t)
  4. t = t + 1
  5. if termination criterion achieved go to step 10
  6. select (P)
  7. crossover (P)
  8. mutate (P)
  9. go to step 3
  10. output best chromosome and stop
End
```

Fig. 1. Basic steps of GA.

NSGA-II

- Step 1: Combine parent and offspring populations and create $R_t = P_t \cup Q_t$. Perform a nondominated sort on R_t and identify different fronts: F_i, $i = 1, 2 \ldots$,etc.
- Step 2: Set new population $P_{t+1} = \emptyset$. Set a counter $i = 1$.
- Step 3: Perform the *Crowding-sort* procedure and include the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance values in the sorted F_i to P_{t+1}.
- Step 4: Create offspring population Q_{t+1} from P_{t+1} by using the crowded tournament selection, crossover and mutation operators.

Fig. 2. Main steps of NSGA-II.

3. Problem formulation

In this section, we formulate the problems of binary and real (or weighted) vote based classifier ensembles under the single and multiobjective optimization frameworks.

3.1. Single objective formulation of classifier ensemble problem

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

Find the combination of votes V per classifier C_i which will optimize a function $F(V)$. Here, V can be either a Boolean array (binary vote based ensemble) of size $N \times M$ or a real array (real/weighted vote based ensemble) of size $N \times M$.

In the case of Boolean array: $V(i,j)$ denotes the decision whether i^{th} classifier is allowed to vote for the j^{th} class. $V(i,j) = true/1$ denotes that the i^{th} classifier is allowed to vote for the j^{th} class; else $V(i,j) = false/0$ denotes that the i^{th} classifier is not allowed to vote for the j^{th} class.

In the case of real array: $V(i,j)$ 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 the classifiers using weighted voting. Here, F is a classification quality measure of the combined classifiers.

The particular type of problem like NER has mainly three different kinds of classification quality measures, namely recall, precision and F-measure. Thus, $F \in$ {recall, precision, F-measure}. Here, we choose $F =$ F-measure, which is a combination (harmonic mean) of both recall and precision.

3.2. Multiobjective formulation of classifier ensemble problem

The multiobjective formulation of classifier ensemble problem is as follows:

Find the combination of votes per classifier V such that:

maximize $[F_1(B),\ F_2(B)]$ *where* $F_1, F_2 \in$ {recall, precision, F-measure} and $F_1 \neq F_2$. Here, V is either a Boolean array (binary vote based ensemble) or a real array (real vote based ensemble) of size $N \times M$. We choose $F_1 =$ recall and $F_2 =$ precision.

3.2.1. Selection of objectives

Performance of MOO largely depends on the choice of the objective functions which should be as much contradictory as possible. In this work, we choose recall and precision as two objective functions. The definitions of recall and precision are given below:

$$\text{recall} = \frac{\text{number of NEs correctly identified by the system}}{\text{number of NEs in the gold standard test data}} \tag{1}$$

$$\text{precision} = \frac{\text{number of NEs correctly identified by the system}}{\text{number of NEs identified by the system}} \tag{2}$$

From the definitions, it is clear that while recall tries to increase the number of tagged entries as much as possible, precision tries to increase the number of correctly tagged entries. These two capture two different classification qualities. Often, there is an inverse relationship between recall and precision, where it is possible to increase one at the cost of reducing the other. For example, an information retrieval system (such as a search engine) can often increase its recall by retrieving more documents, at the cost of increasing the number of irrelevant documents retrieved (i.e., decreasing precision). This is the underlying motivation

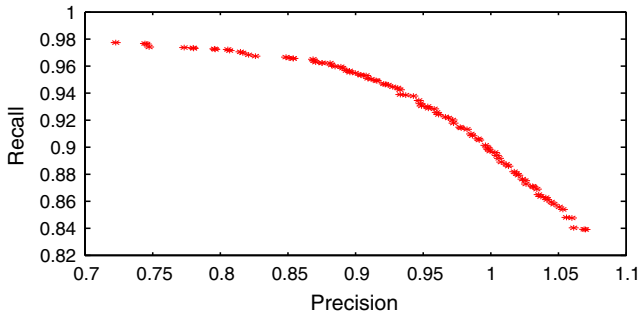

Fig. 3. Pareto optimal front of the proposed MOO based ensemble for Hindi.

of simultaneously optimizing these two objectives. Fig. 3 shows, for example, the Pareto optimal front identified by the proposed real vote based MOO approach. This again supports the contradictory nature of these two objective functions.

Note that F-measure is the harmonic mean (i.e., weighted average) of recall and precision. The equation of F-measure is given below:

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{3}$$

But, it has been thoroughly discussed in the initial chapters (Chapter 2) of Ref. [27] that weighted sum approach cannot identify all non-dominated solutions. Only solutions located on the convex part of the Pareto front can be found. But our other important motivation for this work is to provide the user a set of alternative solutions. Thus, MOO is indeed the best candidate to solve this problem. Here, no weight is required to combine the objectives (i.e., recall and precision) and thus no *a priori* information on the problem is needed.

4. Proposed approach

In this section, we present the solutions to the classifier ensemble problem under both single and multiobjective optimization frameworks. These two solutions are based on GA and NSGA-II, respectively. The GA based weighted voted ensemble technique is proposed in [35].

4.1. String representation and population initialization

If the total number of available classifiers is M and total number of output tags (i.e., number of classes) is O, then the length of the chromosome is $M \times O$. Each chromosome encodes the weights of votes for possible O classes for each classifier.

1. Binary vote based classifier ensemble: As an example, the encoding of a particular chromosome is represented in Fig. 4. Here, $M = 3$ and $O = 4$ (i.e., a total of 12 votes can be possible). The chromosome represents the following voting combination:
   Classifier 1 is allowed to vote for classes 1 and 4;
   Classifier 2 is allowed to vote for classes 1 and 2;

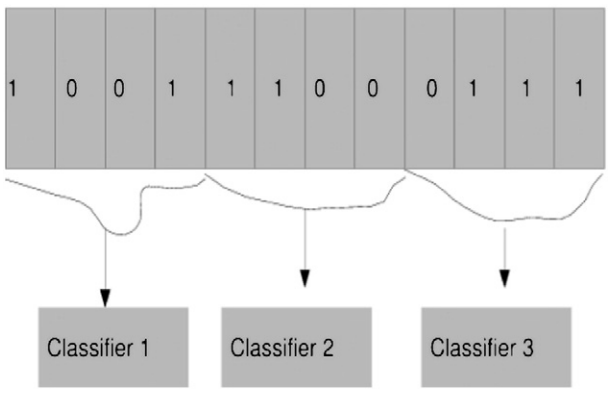

Fig. 4. Chromosome representation for binary voting.

Classifier 3 is allowed to vote for classes 2, 3 and 4.
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 $(i/4+1)^{th}$ classifier is not allowed to vote for the $(i \bmod 4)^{th}$ class. Else, if it is 1 then it means that $(i/4+1)^{th}$ classifier is allowed to vote for the $(i \bmod 4)^{th}$ class. If the population size is P then all the P number of chromosomes of this population are initialized as discussed above.

2. Real (or weighted) vote based classifier ensemble: As an example, the encoding of a particular chromosome is represented in Fig. 5. Here, $M=3$ and $O=3$ (i.e., a total of 9 votes can be possible). The chromosome represents the following voting combination:
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.
Here, we use real encoding, i.e. 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 as discussed above.

4.2. Fitness computation

Initially, all the classifiers are trained using the available training data and evaluated with the development data. The performance of each classifier is measured in terms of the evaluation metrics, namely recall, precision and F-measure. Then, 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 for the development set be F_i, $i=1 \ldots M$, respectively.
2. The ensemble is constructed by combining all the classifiers. Now, for the ensemble classifier the output label for each word in the development data is determined using the weighted voting of these M classifiers' outputs. The binary or real weight of the class provided by the i^{th} classifier is equal to $I(m,i)$. Here, $I(m,i)$ is the entry of the chromosome corresponding to m^{th} classifier and i^{th} class. The combined score of a particular class for a particular word w is:

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

Here, $op(w,m)$ denotes the output class provided by the m^{th} classifier for the word w. The class receiving the maximum combined score is selected as the joint decision.
3. The overall recall, precision and F-measure values of the ensemble classifier are computed on the development set. For single objective approach, we use F-measure value as the objective function, i.e. f_0 = F-measure. In the case of MOO based approach, the objective functions corresponding to a particular chromosome are f_1 = recall and f_2 = precision. The main goal is to maximize these two objective functions using the search capability of NSGA-II.

4.3. Genetic operators used for single objective optimization based approach

In this section we will describe in detail the genetic operators used for solving the single objective optimization based classifier ensemble.

4.3.1. Selection

During each successive generation, a proportion of the existing population is selected to generate a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions.

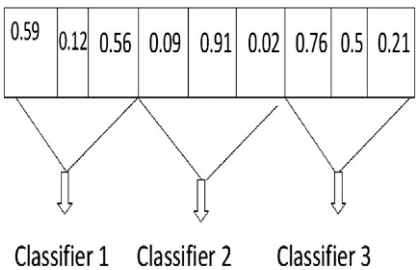

Fig. 5. Chromosome representation for real voting.

In this paper, we use roulette wheel selection. Here, the fitness function associated with each chromosome is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of individual i in the population, its probability of being selected is

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j},$$

where, N is the number of individuals in the population.

This selection process has resemblance to a roulette wheel in a casino. Usually, a proportion of the wheel is assigned to each of the possible selections based on their fitness value. This could be achieved by dividing the fitness of a selection by the total fitness of all the selections, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated. Thus, in the case of roulette wheel selection, chromosomes with higher fitness values are less likely to be eliminated but there is still a chance that they may be.

4.3.2. Crossover

Here, we use the normal single point crossover [39]. As an example, let the two chromosomes be:

P1: 0.24 0.16 0.54 0.87 0.66 0.76 0.01 0.88 0.21
P2: 0.12 0.09 0.89 0.71 0.65 0.82 0.69 0.43 0.15

At first a crossover point has to be selected randomly between 1 and 9 (length of the chromosome) by generating a random number between 1 and 9. Let the crossover point, here, be 4. Then after crossover, the two new offsprings are:

O1: 0.24 0.16 0.54 0.87 0.65 0.82 0.69 0.43 0.15 (taking the first 4 positions from P1 and rest from P2)
O2: 0.12 0.09 0.89 0.71 0.66 0.76 0.01 0.88 0.21 (taking the first 4 positions from P2 and rest from P1)

Similar types of operations can be done with binary encoding. Crossover probability is selected adaptively as in [40]. The probabilities are computed as follows. Let f_{max} be the maximum fitness value of the current population, $\bar{f}$ be the average fitness value of the population and f' be the larger of the fitness values of the solutions to be crossed. Then the probability of crossover, μ_c, is calculated as:

$$\mu_c = \begin{cases} k_1 \times \dfrac{(f_{max} - f')}{(f_{max} - \bar{f})} & \text{if } f' > \bar{f} \\ k_3 & \text{otherwise} \end{cases}$$

Here, as in [40], the values of k_1 and k_3 are kept equal to 1.0. Note that, when $f_{max} = \bar{f}$, then $f' = f_{max}$ and μ_c will be equal to k_3. The aim behind this adaptation is to achieve a trade-off between exploration and exploitation in a different manner. The value of μ_c is increased when the better of the two chromosomes to be crossed is itself quite poor. In contrast when it is a good solution, μ_c is low so as to reduce the likelihood of disrupting a good solution by crossover.

4.3.3. Mutation

Each chromosome undergoes mutation with a probability μ_m. The mutation probability is also selected adaptively for each chromosome as in [40]. The expression for mutation probability, μ_m, is given below:

$$\mu_m = \begin{cases} k_2 \times \dfrac{(f_{max} - f)}{(f_{max} - \bar{f})} & \text{if } f > \bar{f} \\ k_4 & \text{otherwise} \end{cases}$$

Here, values of k_2 and k_4 are kept equal to 0.5. This adaptive mutation helps GA to come out of local optimum. When GA converges to a local optimum, i.e. when $f_{max} - \bar{f}$ decreases, μ_c and μ_m both will be increased. As a result GA will come out of the local optimum. It will also happen for the global optimum and may result in disruption of the near-optimal solutions. This may distract GA to converge to the global optimum. The μ_c and μ_m will get *lower* and *higher* values for *high* and *low* fitness solutions, respectively. While the *high* fitness solutions aid in the convergence of GA, the *low* fitness solutions prevent the GA from getting stuck at a local optimum. The use of elitism will also keep the best solution intact. For a solution with the maximum fitness value, μ_c and μ_m are both zero. The best solution in a population is transferred undisrupted into the next generation. Together with the selection mechanism, this may lead to an exponential growth of the solution in the population and may cause premature convergence. To overcome the above stated problem, a default mutation rate (of 0.02) is kept for every solution in the proposed approach.

In the case of binary encoding, we apply mutation operator to each entry of the chromosome where the entry is randomly replaced by either 0 or 1. But in the case of real encoding, each position in a chromosome is mutated with probability μ_m in the following way. The value is replaced with a random variable drawn from a *Laplacian distribution*, $p(\epsilon) \propto e^{-\frac{|\epsilon - \mu|}{\sigma}}$, where the scaling

factor δ sets the magnitude of perturbation. Here, μ is the value at the position which is to be perturbed. The scaling factor δ is chosen equal to 0.1. The old value at the position is replaced with the newly generated value. By generating a random variable using *Laplacian distribution*, there is a non-zero probability of generating any valid position from any other valid position while probability of generating a value near the old value is more.

4.3.4. Termination condition

In this approach, the processes of fitness computation, selection, crossover and mutation are executed for a maximum number of generations. The best string seen up to the last generation provides the solution to the above classifier ensemble problem. Elitism is implemented at each generation by preserving the best string seen up to that generation in a location outside the population. Thus on termination, this location contains the best classifier ensemble.

4.4. Genetic operators used for MOO based approach

We use crowded binary tournament selection as in NSGA-II, followed by conventional crossover and mutation. Here, mutation operation is same as in single objective optimization. The most characteristic part of NSGA-II is its elitism operation, where the non-dominated solutions [27] among the parent and child populations are propagated to the next generation. The near-Pareto-optimal strings of the last generation provide the different solutions to the ensemble problem.

4.5. Selection of a solution from the final Pareto optimal front in MOO based approach

In MOO, the algorithms produce a large number of non-dominated solutions [27] on the final Pareto optimal front. Each of these solutions provides a vote based classifier ensemble. All the solutions are equally important from the algorithmic point of view. But, sometimes the user may require only a single solution. Consequently, in this paper a method of selecting a single solution from a set of solutions is now developed.

For every solution on the final Pareto optimal front, overall F-measure value of the vote based classifier ensemble is calculated for the development set. The best solution is selected as the one having the highest F-measure value. Final results on the test data are reported using the classifier ensemble corresponding to this best solution. There can be many other different approaches of selecting a solution from the final Pareto optimal front.

5. Base classifiers for NER

We use Naive Bayes, Decision Tree, Memory Based Learner, ME, CRF and SVM as the base classifiers to construct the ensemble system based on weighted voting. Brief descriptions of these classifiers are presented below.

The Naive Bayes classifier [41] is one of the simplest and oldest classifiers. It has been widely used in solving problems in several domains including pattern recognition, NLP, information retrieval, etc. [42–45].

Let $\vec{x}$ be a vector to be classified, and c_k be a possible class. Here the final aim is to find the probability that the vector $\vec{x}$ belongs to the class c_k [41]. Here at first the probability $P\left(c_k\middle|\vec{x}\right)$ is transformed using Bayes' rule,

$$P\left(c_k\middle|\vec{x}\right) = P(c_k) \times \frac{P\left(\vec{x}|c_k\right)}{P\left(\vec{x}\right)} \tag{4}$$

One can estimate the class probability $P(c_k)$ from the training data. But due to the sparseness in the training data, direct estimation of $P\left(c_k\middle|\vec{x}\right)$ is impossible in most cases.

$P\left(\vec{x}\middle|c_k\right)$ can be decomposed by assuming the conditional independence of the elements of a vector as follows,

$$P\left(\vec{x}\right) = \prod_{j=1}^{d} P\left(x_j|c_k\right) \tag{5}$$

where x_j is the j^{th} element of vector $\vec{x}$. Then the equation becomes:

$$P\left(c_k\middle|\vec{x}\right) = P(c_k) \times \frac{\prod_{j=1}^{d} P\left(x_j|c_k\right)}{P\left(\vec{x}\right)} \tag{6}$$

There are some assumptions behind the Naive Bayes classifier like the conditional independence of features which does not hold in many practical cases. However due to its simplicity it has been applied in many tasks including NLP [44].

The experiments were conducted using the WEKA toolkit [46]. This provides implementations of several machine learning algorithms, along with the data structures and code needed to perform data input and output, data filtering, and the evaluation and presentation of results.

5.1. Decision Tree

Decision tree is commonly used in data mining and pattern recognition. The aim is to develop a model that helps in prediction of a target variable based on several input variables. A tree-like structure is generated for prediction, where each internal node corresponds to one of the input variables. There are edges to children for each of the possible values of that input variable. The value of the target variable is represented by a leaf; thus the values of the input variables are represented by the path from the root to the leaf.

We use C4.5 [29] as a decision tree algorithm, which is an extension of ID3 tree learning algorithm. Using the concepts of information entropy, decision trees are built from a set of training data in C4.5. Let the training set be $S = s_1, s_2, \ldots, s_n$, consisting of some already classified samples. Each sample $s_i = x_1, x_2, \ldots, x_n$ is a vector where $x_1, x_2, \ldots, x_n$ represent features of the sample. For the training data another vector is available $C = c_1, c_2, \ldots, c_n$, where $c_1, c_2, \ldots, c_n$ represent the class to which each training sample belongs. In order to generate a sub-tree at a particular node, C4.5 determines a feature of the data that is most useful for splitting its set of samples into subsets enriched in one class or the other. The feature selection is done by optimizing the normalized information gain (difference in entropy) that results from choosing a feature for splitting the data. The feature with the highest normalized information gain is selected for decision making. The C4.5 algorithm then recurs in the smaller sublists.

We use J4.8, which is an implementation of the decision tree algorithm C4.5, publicly available with the WEKA toolkit [46]. For the experiments we use all the default parameters of decision tree. In future we would like to experiment with the detailed parameter tuning.

5.2. Memory Based Learning

Memory based learning (MBL) [30] is a simple and robust machine-learning method that has been applied to a variety of NLP tasks.

These classifiers are descendants of the k-nearest neighbor algorithm [47]. The learning component of MBL is memory-based as it adds new training samples to memory. This memory is known as 'instance base' or 'case base'. It is also referred to as lazy learning as storage is implemented without any abstraction or restructuring. Every instance is a fixed-length vector, consists of two main components. The first component has n feature-value pairs, whereas the second component is known as the information field containing the classification of that particular feature-value vector. The performance component of a MBL system maps input to output based on the product of the learning component. This is the usual form of performing classification. For classification, an unseen instance is presented to the system. Thereafter the similarity between the new instance X and all stored examples Y is computed using some distance metric $\Delta(X, Y)$. All the nearest neighbors of this instance are determined, and subsequently it is assigned the most frequent category. Tie breaking resolution is deployed in case of ties.

We use a memory based tagger (MBT) [48] which makes use of TiMBL, an open source implementation of memory based learning. It uses the IGTREE algorithm for training and testing of the known words. For unknown words IB1 with the overlap metric with gain ratio feature weighting is used. Number of nearest neighbors (i.e. k) is set to 1.

5.3. Hidden Markov Model based named entity tagging

The goal of NER is to find a stochastic optimal tag sequence $T = t_1, t_2, t_3, \ldots, t_n$ for a given word sequence $W = w_1, w_2, w_3 \ldots, w_n$. Generally, the most probable tag sequence is assigned to each sentence following the Viterbi algorithm [49]. The tagging problem becomes equivalent to searching for $argmax_T P(T) * P(W|T)$, by the application of Bayes' law ($P(W)$ is constant).

The probability of the NE tag, i.e., $P(T)$ can be calculated by Markov assumption which states that the probability of a tag is dependent only on a small, fixed number of previous NE tags. Here, in this work, a trigram model has been used. So, the probability of a NE tag depends on two previous tags, and then we have,

$$P(T) = P(t_1) \times P(t_2|t_1) \times P(t_3|t_1, t_2) \times P(t_4|t_2, t_3) \times \ldots \times P(t_n|t_{n-2}, t_{n-1})$$

An additional tag '$' (dummy tag) is introduced in this work to represent the beginning of a sentence. Thus, the modified equation looks as:

$$P(T) = P(t_1|) \times P(t_2|, t_1) \times P(t_3|t_1, t_2) \times P(t_4|t_2, t_3) \times \ldots \times P(t_n|t_{n-2}, t_{n-1})$$

Due to sparse data problem, the linear interpolation method is used to smooth the trigram probabilities as follows: $P'(t_n|t_{n-2}, t_{n-1}) = \lambda_1 P(t_n) + \lambda_2 P(t_n|t_{n-1}) + \lambda_3 P(t_n|t_{n-2}, t_{n-1})$ such that the λs sum to 1. The values of λs have been calculated by the method reported in [50]:

1. set $\lambda_1 = \lambda_2 = \lambda_3 = 0$
2. for each trigram (t_1, t_2, t_3) with $freq(t_1, t_2, t_3) > 0$ depending on the maximum of the following three values:
   - case: $\frac{(freq(t_1, t_2, t_3) - 1)}{(freq(t_1, t_2) - 1)}$: increment λ_3 by $freq(t_1, t_2, t_3)$
   - case: $\frac{(freq(t_2, t_3) - 1)}{freq(t_2) - 1}$: increment λ_2 by $freq(t_1, t_2, t_3)$
   - case: $\frac{(freq(t_3) - 1)}{(N - 1)}$: increment λ_1 by $freq(t_1, t_2, t_3)$

3. normalize λ_1, λ_2, λ_3.

Here, N is the corpus size, i.e., the number of tokens present in the training corpus. If the denominator in one of the expression is 0, then the result of that expression is defined to be 0. The -1 in both the numerator and denominator has been considered for taking unseen data into account.

By making the simplifying assumption that the relation between a word and its tag is independent of context, $P(W|T)$ can be calculated as:$P(W|T) \approx P(w_1|t_1) \times P(w_2|t_2) \times \ldots \times P(w_n|t_n)$

The emission probabilities in the above equation can be calculated from the training set as, $P(w_i|t_i) = \frac{freq(w_i|t_i)}{freq(t_i)}$.

5.3.1. Context dependency

To make the Markov model more powerful, additional context dependent features were introduced to the emission probability. This specifies that the probability of the current word depends on the tag of the previous word and the tag to be assigned to the current word. Now, $P(W|T)$ is calculated by the equation:

$$P(W|T) \approx P(w_1|, t_1) \times P(w_2|t_1, t_2) \times \ldots \times P(w_n|t_{n-1}, t_n)$$

So, the emission probability can be calculated as:

$$P\left(w_i|t_{i-1}, t_i\right) = \frac{freq(t_{i-1}, t_i, w_i)}{freq(t_{i-1}, t_i)}.$$

Here, the smoothing technique is also applied rather than using the emission probability directly. The emission probability is calculated as:

$$P'(w_i|t_{i-1}, t_i) = \theta_1 P(w_i|t_i) + \theta_2 P(w_i|t_{i-1}, t_i), \text{ where } \theta_1, \ \theta_2 \text{ are two constants such that all } \theta s \text{ sum to } 1.$$

The values of θs should be different for different words. But the calculation of θs for every word takes a considerable time and hence θs are calculated for the entire training corpus. In general, the values of θs can be calculated by the same method that is adopted in calculating λs.

5.3.2. Viterbi algorithm

The Viterbi algorithm [49] allows us to find the best T in linear time. The idea behind the algorithm is that of all the state sequences, only the most probable of these sequences need to be considered. The trigram model has been used in the present work. The pseudo code of the algorithm is shown bellow.

```
for i = 1 to Number_of_Words_in_Sentence
    for each state c ∈ Tag_Set
        for each state b ∈ Tag_Set
            for each state a ∈ Tag_Set
                for the best state sequence ending in state a at time (i-2), b at time (i-1), compute the probability of that state
sequence going to state c at time i.
                end
            end
        end
    end
    Determine the most-probable state sequence ending in state c at time i.
end
```

So if every word can have S possible tags, then the Viterbi algorithm runs in $O(S^3 \times |W|)$ time, or linear time with respect to the length of the sentence.

5.3.3. Handling the unknown words

Handling of unknown words is an important issue in NE tagging. Viterbi algorithm [49] attempts to assign a NE tag to the unknown words. Specifically, suffix features of the words and a lexicon are used to handle the unknown words in Bengali.

For words which have not been seen in the training set, $P(w_i|t_i)$ is estimated based on features of the unknown words, such as whether the word contains a particular suffix. The probability distribution of a particular suffix with respect to specific tag is generated from all words in the training set that share the same suffix. Two additional features that cover the numbers and symbols are also considered.

For Bengali, to handle the unknown words further, a lexicon [51], which was developed in an unsupervised way from the tagged Bengali news corpus, is used. Lexicon contains the Bengali root words and their basic part of speech information such as:

noun, verb, adjective, adverb, pronoun and indeclinable, excluding NEs. The lexicon has around 100,000 word entries. The heuristic is that '*if an unknown word is found to appear in the lexicon, then most likely it is not a named entity*'.

5.4. Maximum entropy framework for NER

The ME framework estimates probabilities based on the principle of making as few assumptions as possible, other than the constraints imposed. Such constraints are derived from the training data, expressing some relationships between features and outcome. The probability distribution that satisfies the above property is the one with the highest entropy. It is unique, agrees with the maximum likelihood distribution, and has the exponential form

$$P\left(t|h\right) = \frac{1}{Z(h)} exp\left(\sum_{j=1}^{n} \lambda_j f_j(h,t)\right) \tag{7}$$

where, t is the NE tag, h is the context (or history), $f_j(h,t)$ are the features with associated weight λ_j and $Z(h)$ is a normalization function.

The problem of NER can be formally stated as follows. Given a sequence of words $w_1,\ldots,w_n$, we want to find the corresponding sequence of NE tags $t_1,\ldots,t_n$, drawn from a set of tags T, which satisfies:

$$P\left(t_1,\ldots,t_n|w_1,\ldots,w_n\right) = \prod_{i=1,2\ldots,n} P(t_i|h_i) \tag{8}$$

where, h_i is the context for the word w_i.

The features are, in general, binary valued functions, which associate a NE tag with various elements of the context. For example:

$$f_j(h,t) = 1 \text{if word(h)} = \text{sachIn and t} = \text{I—PER}$$
$$= 0 \quad otherwise$$

We use the OpenNLP Java based MaxEnt package[6] for the computation of the values of the parameters λ_j. This allows to concentrate on selecting the features, which best characterize the problem instead of worrying about assigning the relative weights to the features. We use the Generalized Iterative Scaling [52] algorithm to estimate the MaxEnt parameters.

5.5. Conditional Random Field framework for NER

Conditional Random Fields (CRFs) [17] are undirected graphical models, a special case of which corresponds to conditionally trained probabilistic finite state automata. Being conditionally trained, these CRFs can easily incorporate a large number of arbitrary, non-independent features while still having efficient procedures for non-greedy finite-state inference and training.

CRF is used to calculate the conditional probability of values on designated output nodes given values on other designated input nodes. The conditional probability of a state sequence $s = <s_1, s_2, \ldots, s_T>$ given an observation sequence $o = <o_1, o_2, \ldots, o_T>$ is calculated as:

$$P_{\wedge}\left(s|o\right) = \frac{1}{Z_o} exp\left(\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \times f_k(s_{t-1}, s_t, o, t)\right),$$

where, $f_k(s_{t-1}, s_t, o, t)$ is a feature function whose weight λ_k, is to be learned via training. The values of the feature functions may range between $-\infty, \ldots +\infty$, but typically they are binary. To make all conditional probabilities sum up to 1, we must calculate the normalization factor,

$$Z_o = \sum_{s} exp\left(\sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \times f_k(s_{t-1}, s_t, o, t)\right),$$

which as in HMMs, can be obtained efficiently by dynamic programming.

To train a CRF, the objective function to be maximized is the penalized log-likelihood of the state sequences given the observation sequences:

$$L_{\wedge} = \sum_{i=1}^{N} log\left(P_{\wedge}\left(s^{(i)}\middle|o^{(i)}\right)\right) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2},$$

where $\{<o^{(i)}, s^{(i)}>\}$ is the labeled training data. The second sum corresponds to a zero-mean, σ^2 -variance Gaussian prior over parameters, which facilitates optimization by making the likelihood surface strictly convex. Here, we set parameters λ to

[6] http://maxent.sourceforge.net/.

maximize the penalized log-likelihood using Limited-memory BFGS [53], a quasi-Newton method that is significantly more efficient, and which results in only minor changes in accuracy due to changes in λ.

When applying CRFs to the NER problem, an observation sequence is a token of a sentence or document of text and the state sequence is its corresponding label sequence.

A feature function $f_k(s_{t-1}, s_t, o, t)$ has a value of 0 for most cases and is only set to be 1, when s_{t-1}, s_t are certain states and the observation has certain properties. We have used the C^{++} based CRF^{++} package,[7] a simple, customizable, and open source implementation of CRF for segmenting or labeling sequential data.

5.6. Support Vector Machine framework for NER

In the field of NLP, Support Vector Machines (SVMs) [31] are applied to text categorization, and are reported to have achieved high accuracy without falling into over-fitting even though with a large number of words taken as the features [54,55]. Suppose, we have a set of training data for a two-class problem: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in R^D$ is a feature vector of the i-th sample in the training data and $y \in \{+1, -1\}$ is the class to which $\mathbf{x}_i$ belongs. In their basic form, a SVM learns a linear hyperplane that separates the set of positive examples from the set of negative examples with *maximal margin* (the margin is defined as the distance of the hyperplane to the nearest of the positive and negative examples). In basic SVMs framework, we try to separate the positive and negative examples by the hyperplane written as:

$$(\mathbf{w}.\mathbf{x}) + b = 0 \quad \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}.$$

SVMs find the "optimal" hyperplane (optimal parameter $\overline{w}$, b) which separates the training data into two classes precisely.

The linear separator is defined by two elements: a weight vector $\mathbf{w}$ (with one component for each feature), and a bias b which stands for the distance of the hyperplane to the origin. The classification rule of a SVM is:

$$sgn(f(\mathbf{x}, \mathbf{w}, b)) \tag{9}$$

$$f(\mathbf{x}, \mathbf{w}, b) = <\mathbf{w}.\mathbf{x} > +b \tag{10}$$

being $\mathbf{x}$ the example to be classified. In the linearly separable case, learning the maximal margin hyperplane $(\mathbf{w}, b)$ can be stated as a convex quadratic optimization problem with a unique solution: *minimize* $||\mathbf{w}||$, *subject to the constraints* (one for each training example):

$$y_i(<\mathbf{w}.x_i > +b) \geq 1. \tag{11}$$

The SVM model has an equivalent dual formulation, characterized by a weight vector α and a bias b. In this case, α contains one weight for each training vector, indicating the importance of this vector in the solution. Vectors with non null weights are called *support vectors*. The dual classification rule is:

$$f(\mathbf{x}, \alpha, b) = \sum_{i=1}^{N} y_i \alpha_i <\mathbf{x}_i.\mathbf{x} > +b. \tag{12}$$

The α vector can be calculated also as a quadratic optimization problem. Given the optimal α^* vector of the dual quadratic optimization problem, the weight vector $\mathbf{w}^*$ that realizes the maximal margin hyperplane is calculated as:

$$\mathbf{w}^* = \sum_{i=1}^{N} y_i \alpha_i^* \mathbf{x}_i. \tag{13}$$

The b^* has also a simple expression in terms of $\mathbf{w}^*$ and the training examples $(\mathbf{x}_i, y_i)_{i=1}^N$.

The advantage of the dual formulation is that efficient learning of non-linear SVM separators, by introducing *kernel functions*. Technically, a *kernel function* calculates a dot product between two vectors that have been (non linearly) mapped into a high dimensional feature space. Since there is no need to perform this mapping explicitly, the training is still feasible although the dimension of the real feature space can be very high or even infinite.

By simply substituting every dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in dual form with any *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j)$, SVMs can handle non-linear hypotheses. Among the many kinds of *kernel functions* available, we will focus on the d-th *polynomial kernel*:

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\mathbf{x}_i.\mathbf{x}_j + 1\right)^d.$$

Use of d-th polynomial kernel function allows us to build an optimal separating hyperplane which takes into account all combination of features up to d.

[7] http://crfpp.sourceforge.net.

We have used YamCha[8] toolkit, an SVM based tool for detecting classes in documents and formulating the NER task as a sequential labeling problem. Here, the *pairwise multi-class decision* method and the *polynomial kernel function* are used. We use TinySVM-0.07[9] classifier.

6. Named entity features

The main features for the NER task are identified and selected mostly without using any deep domain knowledge and/or language specific resources. The following features are used for constructing the various classifiers based on the Naive Bayes, Decision Tree, Memory based Learner, Hidden Markov Model, Maximum Entropy, Conditional Random Field and Support Vector Machine frameworks. These features are language independent in nature, and can be easily obtained for almost all the languages.

1. Context words: These are the preceding and following words 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: Fixed length (say, n) word suffixes and prefixes are very effective in identifying NEs and work well for the highly inflective Indian languages. Actually, these are the fixed length character sequences stripped either from the rightmost (for suffix) or from the leftmost (for prefix) positions of the words. For example, the suffixes of length up to 3 characters of the word "*ObAmA*" [Obama] are "*A*", "*mA*" and "*AmA*". Henceforth, all the Bengali glosses are written in ITRANS notation.[10] The prefixes of length up to 3 characters of the word "*ObAmA*" [Obama] are "*O*", "*Ob*" and "*ObA*". If the length of the corresponding word is less than or equal to $n-1$ then the feature values are not defined and denoted by ND. The feature value is also not defined (ND) if the token itself is a punctuation symbol or contains any special symbol or digit. This feature is included with the observation that NEs share some common suffixes and/or prefixes.
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 sentences 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 at 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. We compile a list of most frequently occurring words from the training set by defining an appropriate threshold value. This threshold value does vary depending upon the size of the training set. In the present work, we set the threshold values at 10, 15 and 7 for Bengali, Hindi and Telugu, respectively. A binary valued feature is 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. Last word of sentence: This feature checks whether the current word is the last word of a sentence or not. In Indian languages, verbs generally appear in the last position of the sentence. Indian languages follow *subject-object-verb* structure. This feature distinguishes NEs from the verbs.
7. 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).
8. 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. This feature is used for ME and SVM models. For CRF, we consider the bigram template that computes the combination of the output labels of the current and previous tokens.
9. Content words in surrounding contexts: We consider 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. 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 at 1 if and only if the context w_{i-3}^{i+3} of w_i contains t.

7. Experimental setup, datasets and evaluation results

In this section, we report the details of experimental setup, datasets of experiments and the evaluation results.

7.1. Experimental setup

For GA, the following parameter values are used: population size = 50, number of generations = 40. The mutation and crossover probabilities are selected adaptively. For NSGA-II, the following parameter values are used: population size = 100,

[8] http://chasen-org/taku/software/yamcha/.
[9] http://cl.aist-nara.ac.jp/taku-ku/software/TinySVM.
[10] Bengali glosses are written using ITRANS notation (http://www.aczone.com/itrans/).

Table 1
NE 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

number of generations = 50, probability of mutation = 0.2 and probability of crossover = 0.9. Note that these values are selected after a thorough sensitivity analysis of the parameter values on the performance of the proposed system for the development set.

The proposed approaches are compared with three different *baseline* ensemble systems, which are defined as below:

- *Baseline 1*: In this *baseline* model, all the individual classifiers are combined together into a final system based on the majority voting of the output class labels. If all the outputs differ then any one is selected randomly.
- *Baseline 2*: All the individual classifiers are combined with the help of a weighted voting approach. In each classifier, weight is calculated based on the F-measure value on the development data. The final output label is selected based on the highest weighted vote.
- *Baseline 3*: For each classifier, the average F-measure value of each output class is computed from the development data. The weight of any classifier is set at the average F-measure value of the corresponding class, assigned by it.

7.2. Datasets for NER

Indian languages are resource-constrained in nature. For NER, we use a named entity (NE) annotated dataset of 250 K word forms of the Bengali news corpus [33], developed from the web-archive of a leading Bengali newspaper. This was annotated 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. This annotation was carried out by one of the authors and verified by an expert. We also use the IJCNLP-08 NER on South and South East Asian Languages (NERSSEAL)[11] Shared Task data of around 100 K word forms 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 the datasets obtained from the NERSSEAL shared task. The underlying reason to adapt the finer NE tagset in the shared task was 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 constituent parts of a larger NE. But, the training data were provided with the type of the maximal NE only. For example, *mahatmA gAndhi roDa* (Mahatma Gandhi Road) was annotated as location and assigned the tag 'NEL' even if *mahatmA* (Mahatma) and *gAndhi* (Gandhi) are NE title person (NETP) and person name (NEP), respectively. The task was to identify *mahatmA gAndhi roDa* as a NE and classify it as NEL. In addition, *mahatmA* and *gAndhi* had to be recognized as NEs of the categories NETP (Title person) and NEP (Person name), respectively.

In the present work, we consider only the classes 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 MISC class that denotes miscellaneous entities. Other classes of the shared task are mapped to the 'other-than-NE' category, denoted by 'O'. Hence, the tagset mapping now becomes as shown in Table 2.

In order to properly denote the boundaries of NEs, four basic NE tags are further divided into the format I-TYPE (TYPE → PER/LOC/ORG/MISC) which means that the word is inside a NE of type TYPE. Only if two NEs of the same type immediately follow each other, the first word of the second NE will have tag B-TYPE to show that it starts a new NE. For example, the name *mahatmA gAndhi*[Mahatma Gandhi] is tagged as *mahatmA*[Mahatma]/I-PER *gAndhi*[Gandhi]/I-PER. But, the names *mahatmA gAndhi* [Mahatma Gandhi] *ect rabIndrAnAth thAkur* [Rabindranath Tagore] are to be tagged as: *mahatmA*[Mahatma]/I-PER *gAndhi* [Gandhi]/I-PER *rabIndrAnAth*[Rabindranath]/B-PER *thAkur*[Tagore]/I-PER, if they appear sequentially in the text. This is the standard IOB format that was followed in the CoNLL-2003 shared task [56].

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

Table 2
Tagset mapping table.

IJCNLP-08 shared task tag	Coarse-grained tag	Meaning
NEP	PER	Person name
NEL	LOC	Location name
NEO	ORG	Organization name
NEN, NEM, NETI	MISC	Miscellaneous name
NED, NEA, NEB, NETP, NETE	O	Other than NEs

The available datasets are randomly partitioned into training and development sets. For each of the languages, the system is tuned on the development set and final evaluation results are reported with the gold standard test set. Some statistics of training and test data are presented in Table 3.

7.3. Evaluation results and discussion

We build two HMM models (bigram and trigram) using local features like current word, previous one or two output tags only. A number of different Naive Bayes, Decision Tree (DT), MBL, ME, CRF and SVM models are also generated by considering the various combinations of the available NE features and/or feature templates. In this particular work, we construct the classifiers from the following set of features:

Various context window within the previous three and next three words, i.e. $w_{i-3}^{i+3} = w_{i-3} \ldots w_{i+3}$ of w_i, word suffixes and prefixes of length up to three ($3+3$ different features) or four ($4+4$ different features) characters, first word, length, infrequent word, last word in the sentence, several digit features, content words in surrounding context, and dynamic NE information.

For Bengali, we generate two HMM based classifiers using the approach mentioned in Section 5.3. Overall results of these classifiers are reported in Table 4. Based on ME, we generated 152 different models, out of which only 21 are shown in Table 4. We also construct several Naive Bayes, Decision Tree, Memory Based, CRF and SVM based classifiers, out of which only 1 Naive Bayes-based, 1 Decision Tree based, 3 MBL based, 9 CRF-based and 8 SVM-based classifiers are shown in Table 4. Please note that in CRF, the 'bigram feature template' represents the combination of current and preceding output labels. The CRF-based model exhibits the best performance with the overall recall, precision and F-measure values of 89.42%, 90.55% and 89.98%, respectively. Thereafter, we apply our proposed single objective GA and multiobjective NSGA-II based approaches to determine the appropriate classifier ensemble. Overall evaluation results of these ensemble techniques along with the best individual classifier and three different *baseline* ensembles are reported in Table 5. Results show that MOO with real vote based approach (or, weighted vote based approach) performs superiorly to all the other methods. It is also evident from the results that the proposed real vote based approaches perform better than the binary vote based approaches for both the single and multiobjective optimization frameworks. The single objective optimization with binary vote based approach attains the overall recall, precision and F-measure values of 91.08%, 92.10% and 91.59%, respectively while MOO with binary vote based approach yields recall, precision and F-measure values of 91.78%, 92.90 and 92.34%, respectively. Results show that the single objective optimization based approach with real coding performs better than the best performing individual classifier with the increments in recall, precision and F-measure values by 3.39, 2.37 and 2.88 percentage points, respectively. The real coded ensemble technique also performs reasonably better than three *baseline* models. It shows the overall performance improvements of 7.50, 6.76 and 5.71 percentage F-measure points over *Baseline 1*, *Baseline 2* and *Baseline 3*, respectively. The real coded single objective GA also attains an improvement of 1.27 percentage F-measure points compared to the binary coded GA.

The MOO with real vote based approach shows the highest performance with the recall, precision and F-measure values of 94.21%, 94.72% and 94.47%, respectively. This is actually the increments of 9.11, 8.37 and 7.32 percentage F-measure points over *Baseline 1*, *Baseline 2* and *Baseline 3*, respectively. The relatively lower performance in the *baselines* suggest that rather than blindly combining all the classifiers, it is better to quantify the voting weights for each class in each classifier. It is interesting to note that even the *baselines* perform lower compared to the best individual classifier. Evaluation also shows the superiority of the MOO based method over the single objective approach with the increments of 1.40, 1.80 and 1.61percentage points in recall, precision and F-measure, respectively. The real coded MOO based approach attains 2.13 more F-measure points than the binary coded MOO.

Table 3
Statistics of the datasets.

Language	# Tokens in Training	#NEs in Training	#Tokens in Test	#NEs in Test	Unknown NEs (in %)
Bengali	312,947	37,009	37,053	4413	35.10
Hindi	503,179	26,432	32,796	2022	33.03
Telugu	64,026	8178	8006	3153	34.17

Table 4
Evaluation results with various feature combinations for Naive Bayes, Decision Tree (DT), Memory Based Learning (MBL), HMM, ME, 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, 'PW': position of the word, 'FW': first word, DI: 'Digit-Information', 'NE/FT': Dynamic NE information/feature template, 'CT': Content word feature, $-i,j$: words spanning from the i^{th} left to the j^{th} right position, '$-i$': previous i no. of tokens, current token is at 0^{th} position, 'P': previous token, 'C': current token, 'N': next token, 'B': bigram feature template. *MBL1*: MBL trained with default parameter settings (known words: ddfa, unknown words:dFapsss), *MBL2*: MBL trained with feature pattern 'ddfa' for known words and features pattern 'dFapsss' for unknown words, *MBL3*: MBL trained with FW, WL, IW, PW, DI and Sem features along with default feature patterns for known and unknown words. Here 'd': left context output tag, 'f': ambiguous tag for known word, 'a': right ambiguous tag of both known and unknown words, 'p': character at the start of the word, 's': character at the end of the word, 'F': position of unknown word, 'w': left or right context word(s) of both known and unknown, 'r': recall, 'p': precision, 'F': F-measure, X: denotes the presence of the corresponding feature (we report percentages).

Classifier	Context	FW	PS	SS	WL	IW	PW	DI	NE/FT	CT	r	p	F
Naive Bayes	−3,+3	X	3	3	X	X	X	X		X	63.00	56.94	59.82
DT	−3,+3	X	3	3	X	X	X	X		X	82.52	83.32	82.92
MBL1											69.45	72.22	70.81
MBL2											70.11	71.42	70.76
MBL3		X			X	X	X	X		X	74.76	74.98	74.87
HMM (bigram)	−1										77.04	75.6	76.31
HMM (trigram)	−2										78.02	76.1	77.05
ME_1	−2,2	X	3					X	−2	X	83.69	87.92	85.75
ME_2	−2,1	X	3					X	−2	X	83.92	87.87	85.85
ME_3	−1,1	X	3					X	−2	X	84.05	86.96	85.48
ME_4	−1,2	X	3					X	−2	X	86.03	89.70	85.83
ME_5	−2,2	X	3	3				X	−2	X	84.87	88.09	86.45
ME_6	−2,1	X	3	3				X	−2	X	86.82	88.28	86.52
ME_7	−2,0	X	3	3				X	−2	X	83.92	87.36	85.60
ME_8	−1,1	X	3	3				X	−2	X	84.48	86.63	85.54
ME_9	−1,2	X	3	3				X	−2	X	85.12	87.52	86.30
ME_{10}	0,2	X	3	3				X	−2	X	84.76	86.69	85.71
ME_{11}	−3,3	X	3	3				X	−2	X	84.12	88.10	86.07
ME_{12}	−2,2	X	4	3				X	−2	X	83.44	88.23	85.77
ME_{13}	−2,1	X	4	3				X	−2	X	83.62	88.15	85.83
ME_{14}	−1,1	X	4	3				X	−2	X	83.71	87.09	85.37
ME_{15}	−1,2	X	4	3				X	−2	X	83.71	87.84	85.73
ME_{16}	−2,2	X	3	4				X	−2	X	83.80	87.89	85.80
ME_{17}	−2,1	X	3	4				X	−2	X	84.21	87.87	86.00
ME_{18}	−2,0	X	3	4				X	−2	X	83.46	87.05	85.22
ME_{19}	−1,1	X	3	4				X	−2	X	83.78	86.56	85.15
ME_{20}	−1,2	X	3	4				X	−2	X	84.17	87.52	85.81
ME_{21}	−3,3	X	3	4				X	−2	X	83.19	87.74	85.41
CRF_1	−2,2	X	4	4	X	X	X	X	B	X	88.67	89.91	89.29
CRF_2	−3,3	X	4	4	X	X	X	X	B	X	88.49	89.71	89.09
CRF_3	−3,2	X	4	4	X	X	X	X	B	X	88.51	89.79	89.15
CRF_4	−1,1	X	4	4	X	X	X	X	B	X	88.49	89.26	88.87
CRF_5	−2,2	X	4	4	X	X	X	X	B		74.85	83.03	78.73
CRF_6	−2,2	X	3 (P and C)	3 (P and C)	X	X	X	X	B	X	89.42	90.55	89.98
CRF_7	−2,2	X	3	3	X	X	X	X	B	X	89.06	90.10	89.57
CRF_8	−2,2	X	3 (C and N)	3 (C and N)	X	X	X	X	B	X	88.46	89.75	89.10
CRF_9	−1,1	X	3	3	X	X	X	X	B	X	87.85	89.25	88.55
SVM_1	−1,1	X	4	4	X	X	X	X	−1	X	87.58	87.31	87.44
SVM_2	−2,2	X	4	4	X	X	X	X	−1	X	87.49	87.59	87.54
SVM_3	−2,2	X	4	4	X	X	X	X	−2	X	87.13	87.19	87.16
SVM_4	−2,2	X	3	3	X	X	X	X	−2	X	87.29	87.39	87.34
SVM_5	−2,2	X	2	2	X	X	X	X	−2	X	86.52	86.95	86.73
SVM_6	−2,2	X	2	2	X	X	X		−2	X	86.63	86.95	86.79
SVM_7	−2,2	X	4	4	X	X	X	X	−2		75.93	83.32	79.45
SVM_8	−2,2	X	3	3					−2	X	65.65	76.71	70.75

Table 5
Overall results for Bengali (we report percentages).

Classification scheme	Recall	Precision	F-measure
Best individual classifier	89.42	90.55	89.98
Baseline 1	84.83	85.90	85.36
Baseline 2	85.25	86.97	86.10
Baseline 3	86.97	87.34	87.15
Single objective with binary vote based classifier ensemble [35]	91.08	92.10	91.59
MOO with binary vote based classifier ensemble	91.78	92.90	92.34
Single objective with real vote based classifier ensemble	92.81	92.92	92.86
MOO with real voted based classifier ensemble	94.21	94.72	94.47

Table 6
Estimated marginal means and pairwise comparison between the proposed MOO with binary vote based classifier ensemble approach (MOO_{binary}) and several other ensembles.

Evaluation criterion	Technique (I)	Comp.	Mean diff. (I − J)	Significance value
F-measure	MOO_{binary}	Individual classifier	2.36 ± 0.013	$1.1623e-009$
F-measure	MOO_{binary}	*Baseline 1*	6.98 ± 0.014	$3.3990e-009$
F-measure	MOO_{binary}	*Baseline 2*	6.24 ± 0.011	$8.6386e-010$
F-measure	MOO_{binary}	*Baseline 3*	5.19 ± 0.014	$5.5376e-010$
F-measure	MOO_{binary}	GA_{binary}	0.75 ± 0.012	$2.2356e-010$

Table 7
Estimated marginal means and pairwise comparison between the proposed MOO with real vote based approach (MOO_{real}) and several other ensembles.

Evaluation criterion	Technique (I)	Comp.	Mean diff. (I − J)	Significance value
F-measure	MOO_{real}	Individual classifier	4.49 ± 0.011	$2.3323e-009$
F-measure	MOO_{real}	*Baseline 1*	9.11 ± 0.0189	$4.8990e-009$
F-measure	MOO_{real}	*Baseline 2*	8.37 ± 0.0101	$6.6176e-010$
F-measure	MOO_{real}	*Baseline 3*	7.32 ± 0.0178	$7.9786e-010$
F-measure	MOO_{real}	MOO_{binary}	2.13 ± 0.0178	$7.9786e-010$
F-measure	MOO_{real}	GA_{real}	1.61 ± 0.0109	$5.5216e-010$

Statistical analysis of variance, (ANOVA) [57], is performed in order to examine whether the MOO with binary or real vote based classifier ensemble techniques really outperform the best individual classifier, three *baseline* ensembles and the corresponding single objective GA based approaches. Here, all the classifiers and the proposed ensemble techniques are executed 10 times. Thereafter, ANOVA analysis is carried out on these outputs. Evaluation results of the ANOVA analyses are shown in Tables 6 and 7 for MOO with binary and real vote based approaches, respectively. ANOVA tests show that the differences in mean recall, precision and F-measure are statistically significant as the p value is less than 0.05 in each of the cases. Results also reveal that MOO based techniques truly perform better than the corresponding single objective GA based techniques.

For the purpose of illustration the boxplots of recall, precision and F-measure values of the final Pareto optimal front for Bengali obtained after the application of the proposed MOO with real vote based approach are shown in Figs. 7, 8 and 9, respectively. The corresponding final Pareto optimal front is also shown in Fig. 6.

Thereafter, the proposed system is evaluated on Hindi data. Two HMM based NER systems are developed using the bigram and trigram probabilistic models. Additional context dependent information is introduced during emission probabilities. Unlike Bengali, we have not used any technique for handling the unknown words due to the unavailability of NE suffixes in Hindi. Evaluation results of these two classifiers are shown in Table 8.

For Naive Bayes, DT, MBL, ME, CRF and SVM we use the same set of features as Bengali. Several different versions of these classifiers are built by varying the available features and/or feature templates. Based on the performance, 44 classifiers (Naive Bayes:1, DT:2, MBL: 3, ME:22, CRF:9, SVM:8), which are selected to construct the ensemble, are shown in Table 8. Like Bengali, CRF-based approach yields the best individual performance with the overall recall, precision and F-measure values of 88.72%,

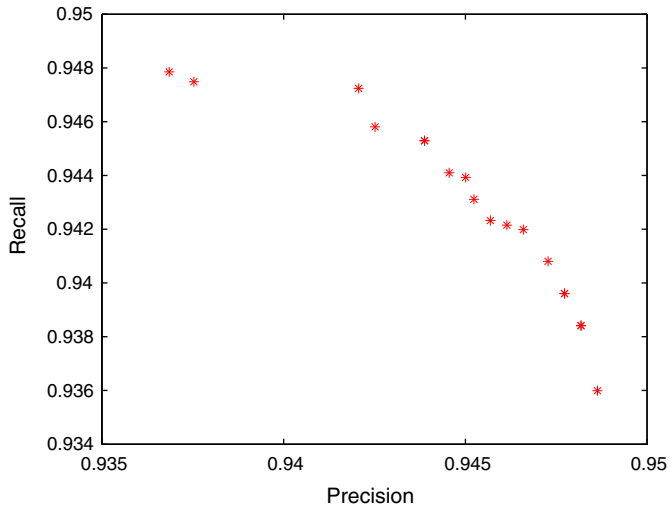

Fig. 6. Pareto optimal front obtained by the MOO with real vote based approach for Bengali.

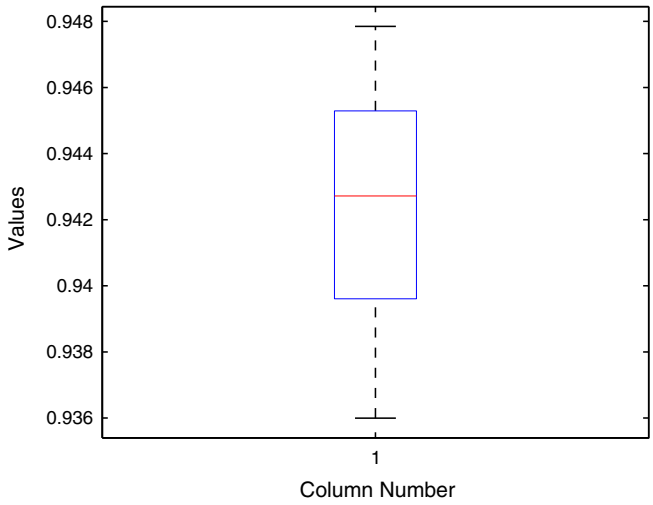

Fig. 7. Boxplot of the recall values obtained by the proposed MOO with real vote based technique for Bengali.

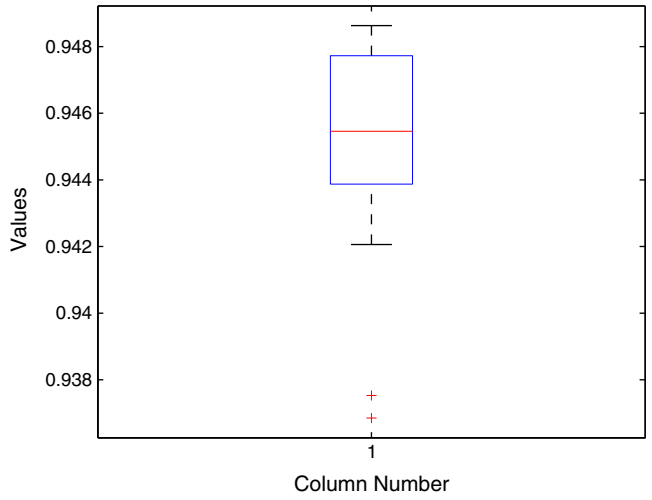

Fig. 8. Boxplot of the precision values obtained by the proposed MOO with real vote based technique for Bengali.

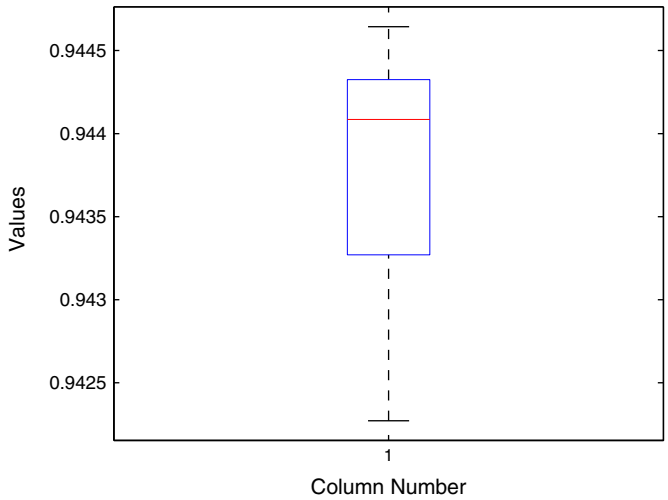

Fig. 9. Boxplot of the F-measure values obtained by the proposed MOO with real vote based technique for Bengali.

Table 8
Evaluation results with various feature combinations for Naive Bayes, Decision Tree (DT), Memory Based Learning (MBL), HMM, ME, CRF and SVM based classifiers for Hindi. Here, the abbreviations are the same as for Bengali (we report in percentages).

Classifier	Context	FW	PS	SS	WL	IW	PW	DI	NE/FT	CT	r	p	F
Naive Bayes	−3,+3	X	3	3	X	X	X	X		X	65.31	48.63	55.75
DT	−3,+3	X	3	3	X	X	X	X		X	81.02	85.32	83.11
MBL1											72.42	71.14	71.77
MBL2											69.68	74.71	72.11
MBL3		X			X	X	X	X		X	70.43	70.24	70.33
HMM(*bigram*)	−1										74.05	73.39	73.72
HMM(*trigram*)	−2										75.01	74.29	74.65
ME_1	−2,2	X	3					X	−1	X	81.73	89.09	85.25
ME_2	−2,1	X	3					X	−1	X	82.13	88.71	85.29
ME_3	−1,1	X	3					X	−1	X	82.99	89.02	85.90
ME_4	−1,2	X	3					X	−1	X	82.33	89.52	85.78
ME_5	0,2	X	3					X	−1	X	82.59	89.34	85.84
ME_6	0,1	X	3					X	−1	X	85.19	90.67	87.85
ME_7	−2,2	X	3	3				X	−1	X	84.16	92.21	86.00
ME_8	−2,1	X	3	3				X	−1	X	83.03	90.00	86.37
ME_9	−2,0	X	3	3				X	−1	X	82.03	86.79	85.28
ME_{10}	−1,1	X	3	3				X	−1	X	82.63	89.66	86.00
ME_{11}	−1,2	X	3	3				X	−1	X	82.53	90.03	86.12
ME_{12}	0,2	X	3	3				X	−1	X	82.43	89.73	85.93
ME_{13}	0,1	X	3	3				X	−1	X	83.72	89.52	86.53
ME_{14}	−3,3	X	3	3				X	−1	X	81.23	90.91	85.80
ME_{15}	−2,2	X	3	4				X	−1	X	81.99	90.35	85.97
ME_{16}	−2,1	X	3	4				X	−1	X	82.69	89.73	86.07
ME_{17}	−2,0	X	3	4				X	−1	X	82.03	88.91	85.33
ME_{18}	−1,1	X	3	4				X	−1	X	83.03	89.65	86.21
ME_{19}	−1,2	X	3	4				X	−1	X	82.63	90.27	86.28
ME_{20}	0,2	X	3	4				X	−1	X	82.36	89.56	85.81
ME_{21}	0,1	X	3	4				X	−1	X	83.29	89.43	86.26
ME_{22}	−3,3	X	3	4				X	−1	X	81.19	90.74	85.71
CRF_1	−2,2	X	4	4	X	X	X	X	B	X	88.02	89.36	88.68
CRF_2	−3,3	X	4	4	X	X	X	X	B	X	87.42	88.90	88.15
CRF_3	−3,2	X	4	4	X	X	X	X	B	X	87.92	89.35	88.63
CRF_4	−1,1	X	4	4	X	X	X	X	B	X	88.62	89.78	89.20
CRF_5	−2,2	X	4	4	X	X	X	X	B		66.82	80.32	72.95
CRF_6	−2,2	X	3 (P and C)	3 (P and C)	X	X	X	X	B	X	87.72	89.17	88.44
CRF_7	−2,2	X	3	3	X	X	X	X	B	X	88.72	90.10	89.40
CRF_8	−2,2	X	3 (C and N)	3 (C and N)	X	X	X	X	B	X	87.59	89.07	88.32
CRF_9	−1,1	X	3	3	X	X	X	X	B	X	87.45	89.08	88.26
SVM_1	−1,1	X	4	4	X	X	X	X	−1	X	87.95	88.33	88.14
SVM_2	−2,2	X	4	4	X	X	X	X	−1	X	88.38	89.24	88.81
SVM_3	−2,2	X	4	4	X	X	X	X	−2	X	85.55	86.27	85.91
SVM_4	−2,2	X	3	3	X	X	X	X	−2	X	84.89	85.63	85.26
SVM_5	−2,2	X	2	2	X	X	X	X	−2	X	83.72	84.65	84.18
SVM_6	−2,2	X	2	2	X	X	X		−2	X	83.22	84.09	83.65
SVM_7	−2,2	X	4	4	X	X	X	X	−2		83.79	84.46	84.12
SVM_8	−2,2	X	3	3					−2	X	64.65	75.71	69.74

90.10% and 89.40%, respectively. Thereafter, *baseline*, single objective GA and MOO based ensemble techniques are executed on these available classifiers. Overall evaluation results are presented in Table 9.

The binary coded GA based approach attains the recall, precision and F-measure values of 89.72%, 91.25% and 90.48%, respectively whereas the binary coded MOO based approach attains the recall, precision and F-measure values of 94.27%, 89.45%

Table 9
Overall results for Hindi (we report percentages).

Classification scheme	Recall	Precision	F-measure
Best individual classifier	88.72	90.10	89.40
Baseline 1	63.32	90.99	74.69
Baseline 2	74.67	94.73	83.64
Baseline 3	75.52	96.13	84.59
Single objective with binary vote based ensemble	89.72	91.25	90.48
MOO with binary vote based ensemble	94.27	89.45	91.80
Single objective with real vote based ensemble [35]	90.92	92.56	91.73
MOO with real vote based ensemble	99.07	90.63	94.66

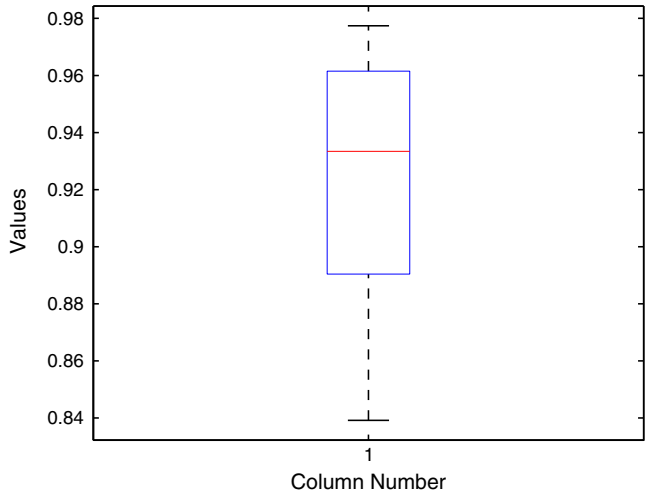

Fig. 10. Boxplot of the recall values obtained by the proposed MOO with real vote based technique for Hindi.

and 91.80%, respectively. Results show that real vote based single objective and MOO based approaches perform better than the corresponding binary vote based approaches. The single objective GA based classifier ensemble with real voting [35] shows the overall recall, precision and F-measure values as 90.92%, 92.56% and 91.73%, respectively. These are the increments of 2.33, 7.04, 8.09, 7.14 and 1.25 percentage F-measures compared to the best individual classifier, *Baseline 1*, *Baseline 2*, *Baseline 3* and single objective with binary vote based classifier ensemble techniques, respectively.

The MOO based technique with real weight yields the overall recall, precision and F-measure values of 99.07%, 90.63% and 94.66%, respectively. It performs better with more than 5.26, 19.97, 11.02, 10.07 and 2.86 percentage F-measure points over the best individual classifier, *Baseline 1*, *Baseline 2*, *Baseline 3* and MOO with binary vote based techniques, respectively. This MOO based approach also shows superior performance compared to the single objective version [35] with more than 2.93 percentage F-measure points. ANOVA tests showed that the performance improvements of the proposed approaches over the best individual classifier and *baselines* are statistically significant. Results also show that MOO based technique performs significantly better than the corresponding single objective GA based ensembles.

The boxplots of recall, precision and F-measure values on the final Pareto optimal front obtained by the MOO with real vote based classifier ensemble technique for Hindi are shown in Figs. 10, 11 and 12, respectively. The corresponding final Pareto optimal front is also shown in Fig. 3.

Finally, we apply our proposed method for Telugu, another popular language widely spoken in the southern part of India. Like Hindi, we generate two different models of HMM using bigram and trigram probabilities. Again additional context dependent information is considered during emission probabilities. But, no specific technique for handling the unknown words is used due to the unavailability of NE suffixes in Telugu. Evaluation results of these classifiers are reported in Table 10. Initially, several different

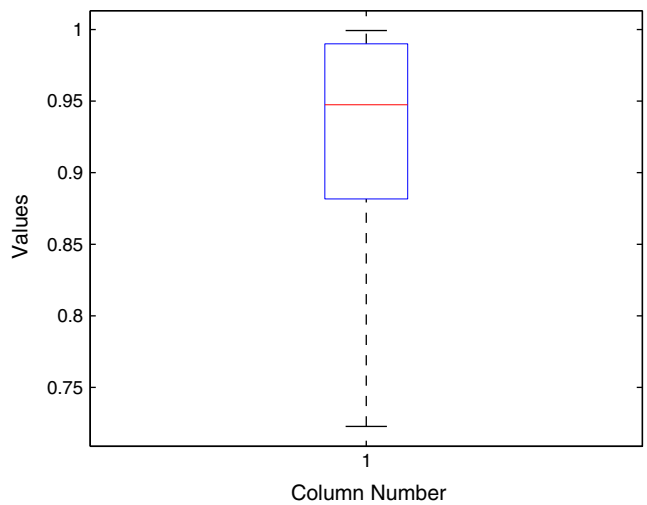

Fig. 11. Boxplot of the precision values obtained by the proposed MOO with real vote based technique for Hindi.

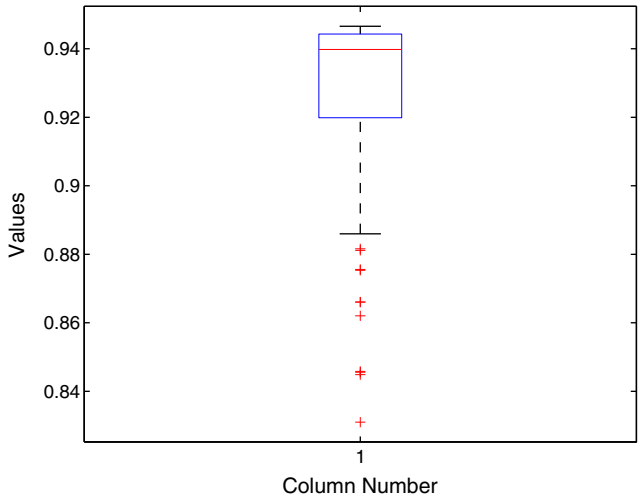

Fig. 12. Boxplot of the F-measure values obtained by the proposed MOO with real vote based technique for Hindi.

versions of Naive Bayes, DT, MBL, ME, CRF and SVM based classifiers are developed using the same set of features as of Bengali and Hindi. Among them, based on the performance, 1 Naive Bayes, 1 DT, 3 MBL, 29 ME, 9 CRF and 8 SVM based classifiers are selected for constructing the ensemble. These classifier combinations are reported in Table 10. The best individual model corresponds to a SVM-based classifier which yields recall, precision and F-measure values of 77.42%, 77.99% and 77.70%, respectively. Evaluation results of the proposed ensemble techniques along with three *baselines* are presented in Table 11. The binary vote based approaches attain recall, precision and F-measure values of 78.71%, 89.10% and 83.58%, respectively for the single objective and 79.53%, 91.71% and 85.19%, respectively for the MOO based approaches. Evaluation results show that single objective with real vote based technique [35] yields the overall recall, precision and F-measure values of 80.82%, 93.26% and 86.60%, respectively. The real voted single objective based technique [35] attains the performance which is better than the best individual classifier by 3.40, 15.27 and 8.90 percentage points in recall, precision and F-measure values, respectively. It also shows 15.37, 5.77, 5.26 and 3.02 percentage higher F-measure points compared to *Baseline 1*, *Baseline 2*, *Baseline 3* and corresponding binary voted ensemble, respectively. Three *baseline* ensembles perform poorly compared to the proposed single objective and MOO based methods. This behavior supports our underlying hypothesis that rather than combing all the available classifiers blindly it is better either to find the eligible classes for which a classifier is allowed to vote or to find the appropriate weights of voting for each class in each classifier. The MOO with real vote based classifier ensemble technique provides the overall recall, precision and F-measure values of 82.79%, 95.18% and 88.55%, respectively. It shows the increment of 10.85, 17.32, 7.72 and 7.21 percentage F-measure points over the best individual classifier and three *baseline* approaches, respectively. This performs superiorly to binary vote based approach with more than 3.36 percentage F-measure point. The MOO based approach also shows better performance over the corresponding single objective version.

7.4. Summary of results

Evaluation results on all the languages show that, in general, the proposed real vote based approaches perform superiorly to the binary vote based approaches. This proves the necessity of determining the proper weights of votes for all the classes in each classifier. It is also evident from the results that MOO based approaches perform better than the corresponding single objective based approaches. This is because in a single objective optimization based technique, a single classification quality measure (i.e., F-measure) is optimized and cannot always capture the quality of a good ensemble reliably. In contrast, an MOO based technique is more efficient to construct the classifier ensemble as it optimizes more than one classification quality measure like recall and precision simultaneously. Thus, it can be resolved that MOO based approaches are more robust compared to the single objective based approaches. All the proposed techniques also perform better than the three standard *baseline* techniques. This proves the need for either looking for the eligible classes for which a classifier is allowed to vote or for searching the proper weights of all the classes in each classifier. Results for all the languages show that real voted approach under single objective optimization framework [35] can be more effective in comparison with the binary vote based approach under MOO framework.

8. Conclusion and future works

In this paper, we have posed the classifier ensemble problem under single and multiobjective optimization frameworks, and evaluated it for NER. We assumed and experimentally shown that instead of eliminating some classifiers completely, it is better to quantify the amount of voting for each class in each classifier. We also hypothesized that reliability of predictions of each classifier

Table 10
Evaluation results with various feature combinations for Naive Bayes, Decision Tree (DT), Memory Based Learning (MBL), HMM, ME, CRF and SVM based classifiers for Telugu. Here, the abbreviations are the same as for Bengali (we report in percentages).

Classifier	Context	FW	PS	SS	WL	IW	PW	DI	NE/FT	CT	r	p	F
Naive Bayes	−3,+3	X	3	3	X	X	X	X		X	60.41	53.72	56.87
DT	−3,+3	X	3	3	X	X	X	X		X	60.85	62.59	61.71
MBL1											46.33	77.64	58.03
MBL2											44.87	74.63	56.04
MBL3		X			X	X	X	X		X	64.08	64.45	64.26
HMM(bigram)	−1										71.17	67.04	69.04
HMM(trigram)	−2										72.07	68.00	69.97
ME_1	−2,2	X	3	–				X	−1	X	67.40	80.94	73.55
ME_2	−2,1	X	3	–				X	−1	X	67.40	80.52	73.38
ME_3	−1,1	X	3	–				X	−1	X	69.01	80.52	74.32
ME_4	−1,2	X	3	–				X	−1	X	67.40	80.11	73.20
ME_5	0,2	X	3	–				X	−1	X	67.54	80.84	73.59
ME_6	0,1	X	3	–				X	−1	X	69.16	79.50	73.97
ME_7	−2,2	X	3	3				X	−1	X	69.01	84.79	76.09
ME_8	−2,1	X	3	3				X	−1	X	69.01	84.79	76.09
ME_9	−2,0	X	3	3				X	−1	X	69.45	84.73	76.33
ME_{10}	−1,1	X	3	3				X	−1	X	71.35	84.98	77.57
ME_{11}	−1,2	X	3	3				X	−1	X	70.48	85.24	77.16
ME_{12}	0,2	X	3	3				X	−1	X	70.33	85.07	77.00
ME_{13}	0,1	X	3	3				X	−1	X	70.97	83.30	76.64
ME_{14}	−3,3	X	3	3				X	−1	X	66.52	83.94	74.22
ME_{15}	−2,2	X	4	3				X	−1	X	69.01	83.61	75.61
ME_{16}	−2,1	X	4	3				X	−1	X	70.33	84.04	76.58
ME_{17}	−2,0	X	4	3				X	−1	X	69.30	83.10	75.57
ME_{18}	−1,1	X	4	3				X	−1	X	71.79	83.37	77.15
ME_{19}	−1,2	X	4	3				X	−1	X	70.92	83.46	76.68
ME_{20}	0,2	X	4	3				X	−1	X	70.92	84.02	76.92
ME_{21}	0,1	X	4	3				X	−1	X	71.53	82.94	76.81
ME_{22}	−3,3	X	4	3				X	−1	X	67.10	83.02	74.22
ME_{23}	−1,1	X	3	4				X	−1	X	69.74	82.07	75.41
ME_{24}	−1,2	X	3	4				X	−1	X	68.72	81.96	74.76
ME_{25}	0,2	X	3	4				X	−1	X	68.13	83.12	74.88
ME_{26}	0,1	X	3	4				X	−1	X	70.48	82.10	75.85
ME_{27}	−1,1	X	4	4				X	−1	X	69.45	79.97	74.34
ME_{28}	0,2	X	4	4				X	−1	X	68.28	80.75	73.99
ME_{29}	0,1	X	4	4				X	−1	X	70.33	80.58	75.11
CRF_1	−2,2	X	4	4	X	X	X	X	B	X	74.63	75.18	74.91
CRF_2	−3,3	X	4	4	X	X	X	X	B	X	73.75	74.30	74.02
CRF_3	−3,2	X	4	4	X	X	X	X	B	X	74.34	74.89	74.61
CRF_4	−1,1	X	4	4	X	X	X	X	B	X	76.39	76.96	76.67
CRF_5	−2,2	X	4	4	X	X	X	X	B		37.98	94.19	54.13
CRF_6	−2,2	X	3 (P and C)	3 (P and C)	X	X	X	X	B	X	73.31	75.08	74.18
CRF_7	−2,2	X	3	3	X	X	X	X	B	X	75.66	77.36	76.50
CRF_8	−2,2	X	3 (C and N)	3 (C and N)	X	X	X	X	B	X	73.61	75.04	74.32
CRF_9	−1,1	X	3	3	X	X	X	X	B	X	73.17	74.70	73.93
SVM_1	−1,1	X	4	4	X	X	X	X	−1	X	77.42	77.99	77.70
SVM_2	−2,2	X	4	4	X	X	X	X	−1	X	77.42	77.99	77.70
SVM_3	−2,2	X	4	4	X	X	X	X	−2	X	74.34	74.89	74.61
SVM_4	−2,2	X	3	3	X	X	X	X	−2	X	73.46	74.00	73.73
SVM_5	−2,2	X	2	2	X	X	X	X	−2	X	71.99	72.53	72.26
SVM_6	−2,2	X	2	2	X	X	X		−2	X	73.61	74.15	73.88
SVM_7	−2,2	X	4	4	X	X	X	X	−2		71.55	76.85	74.12
SVM_8	−2,2	X	3	3					−2	X	73.90	74.45	74.17

Table 11
Overall results for Telugu (we report percentages).

Classification scheme	Recall	Precision	F-measure
Best individual classifier	77.42	77.99	77.70
Baseline 1	60.12	87.39	71.23
Baseline 2	71.87	92.33	80.83
Baseline 3	72.22	93.10	81.34
Single objective with binary vote based ensemble	78.71	89.10	83.58
MOO with binary vote based ensemble	79.53	91.71	85.19
Single objective with real vote based ensemble [35]	80.82	93.26	86.60
MOO with real vote based ensemble	82.79	95.18	88.55

differs among the various output classes. Thus, in an ensemble system it is necessary to find out either the set of classes for which a classifier is most suitable to vote or to search for the appropriate weights of votes for all the classes in each classifier.

We have presented two different versions of the classifier ensemble problem under each of the single and multiobjective optimization frameworks. In the first version, we have investigated appropriate voting combination per classifier (i.e., binary vote based ensemble) and in the second version we have quantified the amount of votes (i.e., real vote based ensemble) for all the classes in each classifier. Thereafter, we present solutions to these two types of problems using single and multiobjective optimization techniques that made use of GA and NSGA-II, respectively. Based on the diverse classification methodologies such as Naive Bayes, DT, MBL, HMM, ME, CRF and SVM, a number of different classifiers were built by selecting different feature and/or feature template combinations. One most interesting characteristic of the features is that these are identified and selected largely without using any domain knowledge and/or language specific resources. The algorithms have been evaluated for three resource-constrained languages, namely Bengali, Hindi and Telugu. The proposed techniques achieve the state-of-the-art performance for all the languages. Evaluation results show that real vote based single and multiobjective approaches perform better than the binary vote based approaches. Overall performance attained by the proposed techniques show the effectiveness by exhibiting better performance compared to the individual classifiers and three different *baseline* ensembles for both kinds of ensemble problems under single and multiobjective optimization frameworks. Results also show that the proposed MOO based techniques perform superiorly to the single objective based methods.

In future we would like to investigate some more language independent features to generate more classifiers. In the current work, we use the default parameter settings of the classifiers. In future we would like to determine the appropriate values of the parameters using some single/multi objective optimization techniques. Future works also include performing NER into different steps, i.e. identification of NEs at the first step and classification of them into the predefined categories in the second step. We will evaluate the proposed techniques for some other Indian languages, English and biomedical domain.

References

[1] T. Mandl, C. Womser-Hacker, The effect of named entities on effectiveness in cross-language information retrieval evaluation, In: Proceedings of the 2005 ACM Symposium on Applied Computing (SAC 2005), 2005, pp. 1059–1064.
[2] M. Pasca, D. Lin, J. Bigham, A. Lifchits, A. Jain, Organizing and searching the world wide web of facts—step one: the one-million fact extraction challenge, In: Proceedings of National Conference on Artificial Intelligence (AAAI-06), 2006.
[3] B. Babych, A. Hartley, Improving machine translation quality with automatic named entity recognition, In: Proceedings of EAMT/EACL 2003 Workshop on MT and Other Language Technology Tools, 2003, pp. 1–8.
[4] L.A. Pizzato, D. Molla, C. Paris, Pseudo relevance feedback using named entities for question answering, In: Proceedings of the 2006 Australian Language Technology Workshop (ALTW-2006), 2006, pp. 89–90.
[5] C. Nobata, S. Sekine, H. Isahara, R. Grishman, Summarization system integrated with named entity tagging and IE pattern discovery, In: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002), Spain, 2002.
[6] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunnigham, Y. Wilks, Univ. of Sheffield: description of the LaSIE-II system as used for MUC-7, MUC-7, Fairfax, Virginia, 1998.
[7] C. Aone, L. Halverson, T. Hampton, M. Ramos-Santacruz, SRA: description of the IE2 system used for MUC-7, In: MUC-7, Fairfax, Virginia, 1998.
[8] A. Mikheev, C. Grover, M. Moens, Description of the LTG System used for MUC-7, In: MUC-7, Fairfax, Virginia, 1998.
[9] A. Mikheev, C. Grover, M. Moens, Named entity recognition without gazetteers, In: Proceedings of EACL, Bergen, Norway, 1999, pp. 1–8.
[10] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, R. Weischedel, the Annotation Group, BBN: description of the SIFT system as used for MUC-7, In: MUC-7, Fairfax, Virginia, 1998.
[11] D.M. Bikel, R.L. Schwartz, R.M. Weischedel, An algorithm that learns what's in a name, Machine Learning 34 (1–3) (1999) 211–231.
[12] A. Borthwick, Maximum entropy approach to named entity recognition, Ph.D. thesis, New York University (1999).
[13] A. Borthwick, J. Sterling, E. Agichtein, R. Grishman, NYU: description of the MENE named entity system as used in MUC-7, In: MUC-7, Fairfax, 1998.
[14] S. Sekine, Description of the Japanese NE system used for MET-2, In: MUC-7, Fairfax, Virginia, 1998.
[15] S.W. Bennet, C. Aone, C. Lovell, Learning to tag multilingual texts through observation, In: Proceedings of Empirical Methods of Natural Language Processing, Providence, Rhode Island, 1997, pp. 109–116.
[16] A. McCallum, W. Li, Early results for named entity recognition with conditional random fields, feature induction and Web-enhanced lexicons, In: Proceedings of CoNLL, Canada, 2003, pp. 188–191.
[17] J.D. Lafferty, A. McCallum, F.C.N. Pereira, Conditional random fields: probabilistic models for segmenting and labeling sequence data, In: ICML, 2001, pp. 282–289.
[18] H. Yamada, T. Kudo, Y. Matsumoto, Japanese named entity extraction using support vector machine, Transactions of IPSJ 43 (1) (2001) 44–53.
[19] R. Srihari, C. Niu, W. Li, A hybrid approach for named entity and sub-type tagging, In: Proceedings of Sixth Conference on Applied Natural Language Processing (ANLP), 2002, pp. 247–254.
[20] X. YU, Chinese named entity recognition with cascaded hybrid model, In: Proceedings of NAACL HLT 2007, Prague, 2007, pp. 197–200.
[21] A. Ekbal, S. Naskar, S. Bandyopadhyay, Named entity recognition and transliteration in Bengali, Named Entities: Recognition, Classification and UseSpecial Issue of Lingvisticae Investigationes Journal 30 (1) (2007) 95–114.
[22] A. Ekbal, S. Bandyopadhyay, Voted NER system using appropriate unlabeled data, In: Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), ACL-IJCNLP 2009, 2009, pp. 202–210.
[23] A. Ekbal, R. Haque, S. Bandyopadhyay, Named entity recognition in Bengali: a conditional random field approach, In: Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP 2008), 2008, pp. 589–594.
[24] W. Li, A. McCallum, Rapid development of Hindi named entity recognition using conditional random fields and feature induction, ACM Transactions on Asian Languages Information Processing 2 (3) (2004) 290–294, http://dx.doi.org/10.1145/979872.979879.
[25] P.M. Shishtla, P. Pingali, V. Varma, 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, 2008, pp. 101–108.
[26] R. Florian, A. Ittycheriah, H. Jing, T. Zhang, Named entity recognition through classifier combination, In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, 2003.
[27] K. Deb, Multi-objective optimization using evolutionary algorithms, John Wiley and Sons Ltd., England, 2001.
[28] A. Patel, G. Ramakrishnan, P. Bhattacharya, An empirical study of the naive Bayes classifier, In: Proceedings of IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, 2001.
[29] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
[30] W. Daelemans, A.V. den Bosch, Memory-Based Language Processing, Cambridge University Press, Cambridge, UK, 2005.

[31] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
[32] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 181–197.
[33] A. Ekbal, S. Bandyopadhyay, A Web-based Bengali news corpus for named entity recognition, Language Resources and Evaluation Journal 42 (2) (2008) 173–182.
[34] A. Ekbal, S. Saha, Classifier ensemble selection using genetic algorithm for named entity recognition, Research on Language and Computation 8 (2010) 73–99.
[35] A. Ekbal, S. Saha, Weighted vote-based classifier ensemble for named entity recognition: a genetic algorithm-based approach, ACM Transactions on Asian Language Information Processing 10 (2) (2011) 9.
[36] A. Ekbal, S. Saha, Multiobjective optimization for classifier ensemble and feature selection: an application to named entity recognition, International Journal on Document Analysis and Recognition 15 (2) (2012) 143–166.
[37] A. Ekbal, S. Saha, A multiobjective simulated annealing approach for classifier ensemble: named entity recognition in Indian languages as case studies, Expert Systems with Applications 38 (12) (2011) 14760–14772.
[38] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[39] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.
[40] M. Srinivas, L.M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics 24 (4) (1994) 656–667.
[41] Y. Tsuruoka, J. Tsujii, Training a naive Bayes classifier via the EM algorithm with a class distribution constraint, In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 — Volume 4, CONLL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003, pp. 127–134, http://dx.doi.org/10.3115/1119176.1119193, (10.3115/1119176.1119193).
[42] G. Escudero, L. Arquez, G. Rigau, Naive Bayes and exemplar-based approaches to word sense disambiguation revisited, In: Proceedings of the 14th European Conference on Artificial Intelligence, 2000.
[43] D.D. Lewis, Naive Bayes at forty: the independence assumption in information retrieval, In: Proceedings of ECML-98, 10th European Conference on Machine Learning, 1998.
[44] A. McCallum, K. Nigam, A comparison of event models for naive Bayes text classification, In: AAAI-98 Workshop on Learning for Text Categorization, 1998.
[45] T. Pedersen, A simple approach to building ensembles of naive Bayesian classifiers for word sense disambiguation, In: Proceedings of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics, Seattle, WA, 2000.
[46] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, San Francisco, 2005.
[47] D.W. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Machine Learning 6 (1) (Jan. 1991).
[48] W. Daelemans, J. Zavrel, A. van den Bosch, K. van der Sloot, MBT: memory-based tagger, In: Version 3.2, Reference Guide. ILK Technical Report 10–04, 2010, http://ilk.uvt.nl/downloads/pub/papers/ilk.1004.pdf.
[49] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory 13 (2) (1967) 260–267.
[50] T. Brants, TnT: a statistical parts-of-speech tagger, In: Proceedings of the Sixth Conference on Applied Natural Language Processing ANLP-2000, 2000, pp. 224–231.
[51] A. Ekbal, S. Bandyopadhyay, Lexicon development and POS tagging using a tagged Bengali news corpus, In: Proceedings of the 20th International Florida AI Research Society Conference (FLAIRS-2007), Florida, 2007, pp. 261–263.
[52] J. Darroch, D. Ratcliff, Generalized iterative scaling for log-linear models, The Annals of Mathematical Statistics 43 (1972) 1470–1480.
[53] F. Sha, F. Pereira, Shallow parsing with conditional random fields, In: Proceedings of NAACL '03, Canada, 2003, pp. 134–141.
[54] T. Joachims, In: Making Large Scale SVM Learning Practical, MIT Press, Cambridge, MA, USA, 1999, pp. 169–184.
[55] H. Taira, M. Haruno, Feature selection in SVM text categorization, In: Proceedings of AAAI-99, 1999.
[56] E.F. Tjong Kim Sang, F. De Meulder, Introduction to the Conll-2003 shared task: language independent named entity recognition, In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, 2003, pp. 142–147.
[57] T.W. Anderson, S. Scolve, Introduction to the Statistical Analysis of Data, Houghton Mifflin, 1978.