Optimizing Genetic Algorithm in Feature Selection for Named Entity Recognition

Huong Thanh LE
Hanoi University of Science and Technology
huongthanh@gmail.com

Luan Van TRAN
Hanoi University of Science and Technology
tranvanluan7@gmail.com

Xuan Hoai NGUYEN
Hanoi University
nxhoai@gmail.com

Thi Hien NGUYEN
Le Quy Don Technical University
nguyenthihienqn@gmail.com

ABSTRACT

This paper proposes some strategies to reduce the running time of genetic algorithms used in a feature selection task for the problem of named entity recognition. They include: (i) reduction of population size during the evolution process of the genetic algorithm; (ii) parallelization of the fitness computation; and (iii) use of progressive sampling for calculating the optimal sample size of the training data. Maximum Entropy algorithm is then used, as a test classifier, to compute the accuracy of the named entity recognition system with the reduced feature sets identified by the genetic algorithm. Experimental results show that our improved genetic algorithm run three time faster than the standard genetic algorithm, while the accuracy of the named entity recognition system (using Maximum Entropy) on the induced feature subset does not decrease. In addition, the feature subset induced by our improved genetic algorithm is much smaller than the original feature set and has helped Maximum Entropy to achieve higher accuracy than the original one.

CCS Concepts

•**Computing methodologies** → **Genetic algorithms;** *Maximum entropy modeling; Feature selection;*

Keywords

Genetic Algorithm; Progressive Sampling; Feature Selection; Named Entity Recognition; Maximum Entropy.

1. INTRODUCTION

Feature selection (FS) is the process of selecting an optimal subset from original features for Machine Learning (ML) algorithms. This is one of the most important factors for the success of ML tasks. The smaller size of the feature subset is, the faster the learning system runs. However, the size of feature subsets is not always proportional with the accuracy of the learning algorithm that works on it. A bad feature set may greatly degrade the performance of the system. Most machine learning approaches determine the features manually, which does not guarantee the optimality of the crafted feature set. The main focus of this paper is to use genetic algorithms for automated feature selection in solving the Named Entity Recognition (NER) task. Although genetic algorithm (GA) has been shown to induce better feature set than other feature selection approaches such as forward selection [10] and backward selection [5] in NER task, its complexity is extremely high. The reason for this is that the genetic algorithm has to compute the fitness of a large number of individuals in the population for every generation. Here, each individual represents a combination of features from the original feature set. For the NER task, the time to compute the fitness of an individual is usually long. To compute the fitness of an individual, the features represented by this individual are fed to the machine learning algorithm for training a NER model, often, on a huge training data set. Then, the NER model is tested with the test set to compute the accuracy of the system. This accuracy is the feedback for computing the fitness of the individual in the genetic algorithm. In this paper, we propose two following strategies to reduce the computation time of the GA: (i) reducing population size of the genetic algorithm after some generations; and (ii) reducing the fitness computation time of individuals in the genetic algorithm. To reduce fitness computation time of each individual, the following techniques are employed: (i) progressive sampling for finding the (near) optimal sample size of the training data; and (ii) parallelization of individual fitness computation in each generation. The rest of this paper is organized as follows. Section 2.1 introduces some backgrounds on genetic algorithms, progressive sampling, and related work of feature selection for NER using genetic algorithms. Section 3 describes our improved genetic algorithm in optimizing the set of features for NER system. Experimental results are given and discussed in Section 5. Finally, Section 6 concludes the paper and highlights some possible extensions of the work in this paper.

SoICT 2015, December 03-04, 2015, Hue City, Viet Nam

DOI: http://dx.doi.org/10.1145/2833258.2833262

2. BACKGROUND

2.1 Genetic algorithm in feature selection

A genetic algorithm [3] is a methodology for solving optimization problems based on natural selection. In a GA, a population of strings, which encodes candidate solutions (called individuals) to an optimization problem, evolves toward better solutions. Each individual is evaluated by a fitness function, which measures the quality of its corresponding solution. The evolution usually starts from a population of randomly generated individuals. In each generation, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has exceeded, or a satisfactory fitness value has been reached. GAs have recently been used in feature selection [8, 6, 11]. Each feature subset is represented by an individual in GA and is encoded in a string of bits. The length of string corresponds to the total number of features considered in the task. Bit's value 1 means the feature is selected and 0 if otherwise. The basic flowchart of the GA system for feature selection is shown in Fig.1 below. One of the main problems of GAs for feature selection in NER task is that its run time complexity is usually extremely high. This is because the individual fitness computation of the GA, in this case, is very time consuming. The fitness computation time for each individual is the total time of training the system (using the feature set induced by the individual), testing it on the test set of data, and computing the system's accuracy. To reduce the time of computing the fitness of individuals, Lanzi [8] introduced inconsistency rate to evaluate the fitness of individuals in the population independently from a learning algorithm. The inconsistency rate specifies to what extent the reduced data still represents the original dataset and can be considered a measure of how much inconsistent the data become when only a subset of attributes is considered. However, such learning algorithm independent approach would miss the chance to discover the feature set that is suitable for the bias of the chosen learning algorithm. Umamaheswari and Radhamani [13] used fuzzy rough set and GA for feature selection. They applied a Rough set method to select an optimal feature subset and then applied a GA to select another one. The union of these two subsets is used for classification. The classification consists of multi linear discriminant analysis and support vector machines. Classification is done on the base of parameter extracted by gray level co-occurrence matrix and histogram texture feature extraction method. Parsi et al. [11] proposed swap training method in a genetic algorithm for feature selection in a face recognition system. K-Nearest Neighbor is used as the classifying algorithm in this approach. In each iteration of the genetic algorithm, the system switches the role of training and test data in order to prevent premature convergence to local minimums. Obtained results from implementing the proposed technique on Yale Face database showed performance improvement of the proposed GA in selecting proper features. Some FS approaches (e.g., [4, 5, 14]) use local search operations for speeding up the convergence of GA. The mutual information between each pair of features is used in [4] as an independent measure for feature ranking in classification tasks. They perform the local search operations in GA by computing the mutual information between each pair of features, which demands excessive computation. Zhu et al. [14] proposed a GA for gene selection, which uses information gain (IG) for local search operations that is sensitive for the real-valued continuous feature set. In this paper, we propose a method to reduce the fitness computation time of an individual by reducing the sample size of the training data set, using progressive sampling [12], which is introduced next.

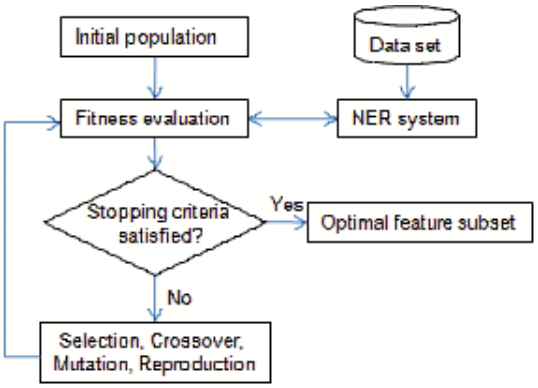

Figure 1: The flowchart of the GA for feature selection.

2.2 Progressive Sampling

Given a large data set and a classification learning algorithm, Progressive Sampling (PS) takes gradually increasing portions of the available data as the sample until the system accuracy no longer improves. At this point, the Optimal Sample Size (OSS) is achieved. It has been shown that the technique is remarkably efficient compared to using the entire data [12]. In progressive sampling, a learning curve (see Fig.2) is used to depicts the relationship between sample size and model accuracy [12]. The horizontal axis in Fig. 2 represents the sample size and the vertical axis represents the accuracy of the model. Most learning curves typically have steeply sloping portion early in the curve, and a plateau late in the curve. In a well-behaved learning curve, progressive sampling will stop at a size equal to or slightly larger than the OSS corresponding to the optimal system accuracy. However, finding the OSS is not simple. Progressive sampling is most efficient when the starting sample size is closed to the OSS. The less the information divergence between the starting sample and the whole data set, the more efficient it is. Gu et al. [1] proposed a method for finding the Statistical Optimal Sample Size by defining an information-based measure of Sample Quality. This sample quality is obtained

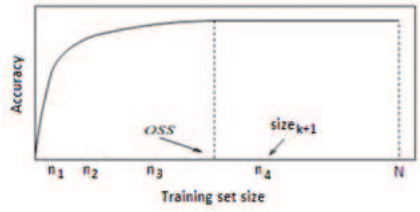

Figure 2: Learning Curves and Progressive Samples.

Table 1: Features for Named Entity Recognition

Index	Feature group	Feature type	Position
1	Context	word	0, 1, -1, 2, -2
2		wordPair	(-1,0), (0,1), (-1,1)
3	Part-of-Speech	pos	0, 1, -1, 2, -2
4		posPair	(-1,0), (0,1), (-1,1)
5	Orthography	Prefix	0, 1, -1, 2, -2
6		Suffix	0, 1, -1, 2, -2
7		LengthOfWord	0, 1, -1, 2, -2
8		Capitalization	0, 1, -1, 2, -2
9		DigitAndSymbol	0, 1, -1, 2, -2
10		IsUrl	0, 1, -1, 2, -2
11		AllLowerCase	0, 1, -1, 2, -2
12		AllCapitalization	0, 1, -1, 2, -2
13		StopWord	0, 1, -1, 2, -2
14	Combination	PrevPosAndCWord	(-1,0)
15	Dictionary	InPerDict	0, 1, -1, 2, -2
16		InLocDict	0, 1, -1, 2, -2
17		InOrgDict	0, 1, -1, 2, -2
18	Statistics	InfrequentWord	0, 1, -1, 2, -2
19		Bigram	(-1,0)
20	Position	FirstWordOfSentence	0, 1, -1

by measuring the information divergence between the sample and the whole data set. The method of Gu et al. [1] will be used in our research to find the optimal sample size of the training data, which will be discussed in details in Section 3.2.

3. THE IMPROVED GENETIC ALGORITHM IN FEATURE SELECTION FOR NER

The NER task considered in this research is to recognize Person, Organization, Location name entities from English texts. By investigating different researches on NER, a set of 20 feature types have been proposed by us. Each feature type is considered in the window size of five. This feature set is depicted in Table 1 above. In Table 1, position 0 means the current word; positions ±1, ±2 mean the word at the position ±1, ±2 relative to the current word. Since some feature types do not need to be considered at some specific positions in the window, the total features to be considered in our NER task is 88. The size of the search space (feature space) of GA is therefore 2^{88} (all possible subset of 88 features).

As mentioned in Section 1, the size of feature's subsets is not proportional to the accuracy of the learning algorithm. Therefore, finding an optimal feature subset is an optimization problem of multiple objectives. To evaluate the fitness of an individual in the population (a feature subset), the following fitness function is used in our GA:

$$Fitness(ind_i) = w_1 * \frac{total(ind_i)}{len(ind_i)} + w_2 * accuracy(ind_i) \quad (1)$$

where ind_i is the i^{th} individual in the population; $total(ind_i)$ is the total number of features in the feature subset (or the total number of values 1 in the individual i); $len(ind_i)$ is the total number of features (or the length of the individual i); $accuracy(ind_i)$ is the accuracy of the NER system when the feature subset represented by the individual i is used. Weights w_1 and w_2 are positive numbers such that $w_1 + w_2 = 1$. In our experiments, w_1 and w_2 are tuned as 0.1 and 0.9, respectively, indicating that the accuracy is more important than the number of features being used. As aforementioned, the main objective of this research is to reduce the computation time of the GA in finding the optimal feature subset for NER. In our proposed approach, this is done by two strategies:

1. To reduce population size of the GA after some generations;
2. To reduce the fitness computation time spent on individuals in GA.

These strategies are described as follows.

3.1 Reducting Population Size of the GA

Reducing population size will reduce the computation time dramatically. Since the population after many generations will converge, removing reasonably some individuals from a generation will not reduce the average fitness of the population. In order to do that, our system needs to consider each individual is bad or not. Let fit_{ij} is the fitness of the individual j at generation i; $\overline{fit_i}$ is the average fitness of the population at generation i; the operator E denotes the average value of fit_{ij}. Then $E(fitij) = \overline{fit_i}$. The standard deviation of a fitness fit_{ij} is calculated by the following formula:

$$\sigma(i) = \sqrt{E[(\overline{fit_i} - fit_{ij})]^2} \quad (2)$$

An individual f_{ij} in generation i is considered as bad if its fitness is smaller than $\overline{fit_i} - \sigma(i)$. In our approach, after N generations (N is determined by testing different values in our experiments), the population at the next generation will be the population at the current generation after removing its bad individuals. Therefore, the population size at the next generation is:

$$pSize(i+1) = pSize(i) - numBadIndividuals(i) \quad (3)$$

In the above formula, $pSize(i)$ is the population size at generation i; $numbadIndividuals(i)$ is the number of bad individuals at generation i.

3.2 Reduction of Individual Fitness Computation Time

To reduce the fitness computation time of individuals, the following techniques are applied: (i) progressive sampling through generations of GA; (ii) parallel computing the fitness of individuals in each generation.

3.2.1 Progressive sampling through generations of GA

In our system, the fitness of an individual depends on the accuracy of the NER system when using the feature subset corresponding to this individual. To compute this accuracy, a training data set is used in a machine learning algorithm with the feature subset to produce a training model. The accuracy is computed based on the recognition result of NER model on the testing data. However, the training time on a large training data set is very time consuming. Our proposed solution to this problem is to use Progressive Sampling [1], a method to reduce the training data without losing accuracy. As mentioned in Section 2.2, Gu et al. [1] developed a method for finding the *Statistical Optimal Sample Size* (SOSS) by defining an information-based measure of Sample Quality. This method is applied to our research to find the optimal sample size from the training data set. Our NER data set contains words with their tags. Let D is the training data set, S is a sample set from the training data set D, $J_k(S, D)$ is the information divergence between D and S based on a feature k. These features are also the ones used in the machine learning algorithm for NER task.

$J_k(S, D)$ is computed by the following formula:

$$J_k(S, D) = \sum_{j=1}^{c} (p_{Sj} - p_{Dj}) log \frac{p_{Sj}}{p_{Dj}} \quad (4)$$

in which p_{ij} is the probability of occurrence of the j^{th} value in population i (here the population i can be the sample set S or the whole training data set D); c is all possible values of the feature k. The average difference in information between S and D is:

$$J = \frac{1}{f} \times \sum_{k=feat_1}^{feat_f} (J_k(S, D)) \quad (5)$$

in which f is the total numbers of features being used. The sample quality of S in the training set D of Gu et al. [1]'s system is measured by e_J. To calculate the SOSS of D, Gu et al. [1] set n sample sizes S_i spanning the range of $[1, N]$ and compute the corresponding qualities Q_i $(i = 1, 2, ..., N)$. They then draw a sample quality curve (relationship between sample size and sample quality) using these (S_i, Q_i) points. The SOSS is estimated using the curve. To calculate samples'quality, upon scanning each sample, a random number r uniformly distributed on [0.0, 1.0) is generated. If r < Si/N, then corresponding statistics (by counting a categorical value or binning a numerical value) are gathered for the i^th sample. In our system, the quality Q_i of a sample set of size S_i is evaluated by measuring the F-score of the NER system when using this sample set. The algorithm to find relationship between sample size and sample quality based on Gu et al. [1]'s algorithm is shown below. In our algorithm, S_1 is the sample size at generation 0; S_n is the size of the entire data set. Support that n_0 is the sample size at generation 0 ($S_1 = n_0$), then the sample size S_i at generation i is $min(N, n_0 * a^i)$, in which N is the size of training data, a is the increasing data rate. n_0 is defined manually in our system. a is calculated such as $n_0 * a^n$ is approximately equal to N. After calculating all pairs (S_i, Q_i), the SOSS is extracted by getting the last sample size returned by the progressive sampling process.

Algorithm 1: The SOSS algorithm to find the starting sample size.

Input: The training data D of size N; n sample sizes $S_1, S_2, ..., S_n$; $S_i \in [1, N]$
Output: n pairs (S_i, Q_i) represent the relation between sample size and sample quality

1. **foreach** *word k in* $D(k \in [1, N])$ **do**
   `/* Update the occurrence times of each feature's value in D */`
   foreach *sample set i (with the size* S_i*)* **do**
   r ← a random real value between 0.0 and 1.0.
   if $(r < Si/N)$ **then**
   update corresponding statistical measure for all features of sample i
2. **foreach** *sample set i* **do**
   calculate its Q_i and output (S_i, Q_i);

4. OUR PROPOSED GENETIC ALGORITHM FOR NER

The improved GA for feature selection is shown in Algorithm 2 below. The algorithm terminates when the population contains only one individual.
The function parallelComputeFitness in the above algorithm is shown in Algorithm 3.
In the function parallelComputeFitness mentioned above, the fitness of each individual in the population is only computed when it is not in the hashMap. This action is used to prevent recomputing the fitness of an individual if is already computed in the previous generation.

5. EXPERIMENTS AND RESULTS

The CoNLL-2003 data set is used in our experiment. This is the data set that is commonly used to evaluate the performance of different language-independent NER systems, offered by the Message Understanding Conferences (MUC). The training and testing data sets consist of approximately 116,000 words and 40,000 words, respectively. Four types of named entities which are concentrated in the CoNLL-2003 data set are persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. Maximum Entropy is used in our experiment as the machine learning method for NER. The F-score of the NER system when using the entire feature set (88 features) is 91.12%. The running time of the MaxEnt algorithm for this data set is $21s$.
Our GA was configured with the following parameters:

- Maximum generation of the GA: $maxGeneration = 40$
- Mutation rate: $mutationThresholdm = 0.05$
- Crossover rate: $crossoverThresholdc = 0.9$

After each generation, the sample size increases 1.004 times (increasing data rate $a = 1.004$). The process of measuring the fitness of individuals was computed in parallel in 4 threads. The initial population size startSize is 19,000 words. This value is computed by the progressive sampling process described in Section 3.2. Since the genetic algorithm uses many random processes, different runs of GA will give different outputs. Therefore, our experiments were carried out five times to objectively evaluate our GA. The average number of features in five time running are 37 features; the average F-score is 94.58%; and the average running time is 1355 minutes. It shows that the optimal feature subset returned by our GA is smaller than the original feature set and provides better performance (the F-score increases from 91.12% to 94.58%). There are several works on feature subset selection for the NER task using GA (e.g., [2, 4, 6]). However, as far as we know, only [6] used the same data set with us. Kitoogo et al. [6] proposed a multi-objective genetic algorithm to select the best features from domain independent features with a maximum entropy model. Their experiments with the CoNLL-2003 data set obtained the F-score of 90.81%. Some other works on automatic feature subset selection for the NER task that used the same data set are [7, 9]. The feature induction method for CRFs in [9] achieved the performance of 89%. Klinger and Friedrich [7] proposed filtering methods using information gain or χ^2 as well as an iterative approach for pruning features with low weights. Their experiments shown that with only 3% of the

Algorithm 2: The improve GA for feature selection.

Input: Training data D; all features $featureNames$;crossover rate c; mutation rate m; increasing data rate a; maximum generation of the GA $maxGeneration$.
Output: The optimal feature subset

1. Compute the starting sample size by using the progressive sampling process described in Section III.B. startSize = computeSOSS($D, S_1, S_2, ..., S_n$)
2. Initialize population P of $startSize$ individuals. Each individual corresponds to a set of randomly selecting features in the set $featureNames$. $PopulationP = createPopu(startSize, featureNames)$
3. Create a starting sample set from the training data D, with the size startSize $dataTrain = sampling(D, startSize)$
4. Compute parallelly fitness of individuals in the population and push them on a hashmap in order to retrieve them easily parallelComputeFitness(P,dataTrain,dataTest)
5. Select the best individual and recompute its fitness using the training data set D. The fitness of the best individual is recomputed since the previous fitness is computed from a subset of the training data, thus it may not be the real fitness of this individual. Chromosome $ch_max = getBest(P, D)$
6. int k=0;

```
while (true) do
    k++;
    // move to the next generation
    if (k > maxGeneration) then
        break;
    // Increase the sample size
    dataTrain = sampling(D,dataTrain.size()*a);
    // Compute the new population size (see
       Section III.A)
    newSize = computeNewSize(P);
    // Crossover, mutate to create new
       population
    P = newGeneration(P,newSize);
    parallelComputeFitness(P,dataTrain,dataTest);
    Chromosome ch = getBest(P,D);
    if (ch.fitness > ch_max) then
        ch_max = ch;
```

7. return $ch\_max.Feature$;

Algorithm 3: The parallelComputeFitness function.

```
Input: A population Popu; training data set dataTrain;
testing data set dataTest
for (Chromosome ch:Popu) do
    /* Check whether the fitness of the
       individual ch is already in hashMap     */
    if (fitnessHashMap.keySet().contains(ch)) then
        ch.fitness= fitnessHashMap.get(ch);
    else
        ch.fitness=computeFitness(ch,
        dataTrain,dataTest);
```

original number of features a 60% inference speed-up is possible, whereas the F1 measure decreases only slightly (0.57% from approximately 88%). Our GA was also compared with different installations of GA, using the CoNLL-2003 data set:

1. running the normal GA with the MaxEnt algorithm (without the progressive sampling and parallel computing) and the population $size = 40$;
2. running GA with the MaxEnt algorithm, using parallel computing (without the progressive sampling) and the population $size = 40$;
3. running GA with the MaxEnt algorithm, using the progressive sampling (without parallel computing);
4. running GA with the MaxEnt algorithm, using the progressive sampling and parallel computing. The average values of five time running shown in Table 2 was used in this case.

Experimental results in Table 2 show that the progressive sampling can efficiently reduce the computational time of GA, even when the parallel computing process is not used. When using the parallel computing process, the computational time can reduce further, however, it does not inverse proportionally to the number of threads being used.

Comparing the feature set returned by different installations of GA.

As mentioned above, different runs of the same GA can give different outputs since the generic algorithm uses many random decisions during its evolutionary process. Therefore, we compared the feature sets returned by 5 running times of our proposed GA and detected the common features shared among these feature sets (Table 3). These common features can be considered as the most important features from the set of 88 features proposed by us. The more times a feature appearing in feature sets, the more important it can be. One feature that never appears in all result feature sets is PrevPosAndCWord. It indicates that this feature is not important in the NER task. The features shared among three other installations of genetic algorithm (MaxEnt + GA, MaxEnt + GA + parallel, MaxEnt + GA + sampling) are Bigram, Capitalization, CurrentWord, DigitAndSymbol-Prev, InfrequentWordNext, Next2InPerDict, PosTag.
By looking at the feature sets returned by different installations of GA, we found that Bigram, Word, Capitalization,

Table 2: Experimental results when using different installations of GA

Run	MaxEnt + GA	MaxEnt + GA + parallel	MaxEnt + GA + PS	MaxEnt + GA + PS + parallel
#features	37	25	41	37
F-score(%)	94.58	93.75	94.41	93.67
Running time (in minute)	1355	389	461	322

Table 3: Shared features returned by 5 time running of our proposed GA

Sharing times	Shared features
5	AllLowerCase, CurrentWord, FirstWord
4	Bigram, DigitAndSymbolPrev, PrevInLocDict, InfrequentWordPrev2 IsUrl, StopWord, Suffix, LengthOfWord, LengthOfWordNext, LengthOfWordPrev2,
3	Capitalization, PosTagPrev2, PrefixPrev2 Prev2InPerDict, PrevInOrgDict, PrevInPerDict, WordPair, SuffixNext2

DigitAndSymbolPrev, InfrequentWord, PosTag, InPerDict are important features in the NER task.

6. CONCLUSION AND FUTURE WORK

This paper focuses on implementing a genetic algorithm for selecting an optimal feature subset that is used in Maximum Entropy classifier for NER task. Several strategies have been proposed to reduce the computation time of GA, including: (i) reducing population size after some generations; (ii) parallel computing the fitness of individuals in each generation; and (iii) progressive sampling for finding the optimal sample size of the training data. Experimental results show that our GA can generate a feature subset which is much smaller than the original one, but it can provide a higher accuracy for NER task. In addition, our improved GA run three times faster than the basic GA, whereas the accuracy of the optimal feature subset of the improved GA does not decrease compared to the basic one. Our future work is to find other strategies to reduce computation time of GA, such as improving the method to compute the fitness of individuals in GA. Instead of computing the real fitness of individuals, these values can be predicted based on the fitness of individuals in previous generations of GA.

7. REFERENCES

[1] B. Gu, B. Liu, F. Hu, and H. Liu. Efficiently determining the starting sample size for progressive sampling. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 192–202, London, UK, 2001. Springer-Verlag.

[2] M. Hasanuzzaman, S. Saha, and A. Ekbal. Feature subset selection using genetic algorithm for named entity recognition. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation, PACLIC 24, Tohoku University, Japan, 4-7 November 2010*, pages 153–162, 2010.

[3] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[4] J. Huang, Y. Cai, and X. Xu. A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recogn. Lett.*, 28(13):1825–1844, Oct. 2007.

[5] M. M. Kabir, M. Shahjahan, and K. Murase. Involving new local search in hybrid genetic algorithm for feature selection. In *ICONIP (2)*, volume 5864 of *Lecture Notes in Computer Science*, pages 150–158. Springer, 2009.

[6] F. Kitoogo, V. Baryamureeba, and G. De Pauw. Towards domain independent named entity recognition. In *Strengthening the Role of ICT in Development*, pages 38–49, Kampala, Uganda, 2008. Fountain Publishers, Fountain Publishers.

[7] R. Klinger and C. M. Friedrich. Feature subset selection in conditional random fields for named entity recognition. In *RANLP*, pages 185–191. RANLP 2009 Organising Committee / ACL, 2009.

[8] P. Lanzi. Fast feature selection with genetic algorithms: a filter approach. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 537–540, Apr 1997.

[9] A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, pages 403–410, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[10] T. Pahikkala, A. Airola, and T. Salakoski. Speeding up greedy forward selection for regularized least-squares. In *The Ninth International Conference on Machine Learning and Applications, ICMLA 2010, Washington, DC, USA, 12-14 December 2010*, pages 325–330, 2010.

[11] A. Parsi, M. Salehi, and A. Doostmohammadi. Swap training: A genetic algorithm based feature selection method applied on face recognition system. *World of Computer Science and Information Technology Journal*, pages 125–130, 2012.

[12] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 23–32, New York, NY, USA, 1999. ACM.

[13] J. Umamaheswari and G. Radhamani. A hybrid approach for classification of dicom image. *World of Computer Science and Information Technology Journal (WCSIT)*, 1(8):364–369, 2011.

[14] Z. Zhu, Y.-S. Ong, and M. Dash. Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition*, 40(11):3236 – 3248, 2007.