Automatic Feature Selection for Named Entity Recognition Using Genetic Algorithm

Huong Thanh Le
Hanoi University of Science and Technology
Hanoi, Vietnam
huonglt@soict.hut.edu.vn

Luan Van Tran
Hanoi University of Science and Technology
Hanoi, Vietnam
tranvanluan7@gmail.com

ABSTRACT

This paper presents a feature selection approach for named entity recognition using genetic algorithm. Different aspects of genetic algorithm including computational time and criteria for evaluating an individual (i.e., size of the feature subset and the classifier's accuracy) are analyzed in order to optimize its learning process. Two machine learning algorithms, k-Nearest Neighbor and Conditional Random Fields, are used to calculate the accuracy of the named entity recognition system. To evaluate the effectiveness of our genetic algorithm, feature subsets returning by our proposed genetic algorithm are compared to feature subsets returning by a hill climbing algorithm and a backward one. Experimental results show that feature subsets obtained by our genetic algorithm is much smaller than the original feature set without losing of predictive accuracy. Furthermore, these feature subsets result in higher classifier's accuracies than that of the hill climbing algorithm and the backward one

Categories and Subject Descriptors

G.2.1 [Combinatorics] - *Combinatorial algorithms, Counting problems*

General Terms

Algorithms, Measurement, Performance, Design, Reliability, Experimentation, Languages, Theory.

Keywords

Optimizing, genetic algorithm, feature selection, named entity recognition

1. INTRODUCTION

Named Entity Recognition (NER) is the task of locating and classifying atomic elements in text into predefined categories (e.g., people, organizations, locations, expressions of time, quantities, etc.). NER is useful in many applications such as search engine, question-answering system, etc. Most NER's systems concentrate on machine learning approaches, whose performance relies on features being used. Selecting a subset of good features is an optimization problem of multiple objectives, such as number of features and accuracy. Using a small feature subset requires less computational time than using a larger one. However, the system's accuracy is lower than using a larger, suitable set of features. Another problem is, using more features increases the system complexity, but does not always increase the system's accuracy. A bad feature may even degrade the performance of the system. Most machine learning approaches for NER determine the features manually. In this paper, we propose a genetic algorithm (GA) to automatically select a minimal subset of features that can produce high accuracy for the NER system. Different aspects are discussed in order to optimize the GA learning process.

The rest of this paper is organized as follows. Section 2 introduces related work of feature selection using GA and the contribution of this paper. Section 3 describes our approach to apply GA in optimizing the set of features for NER system. Experimental results are discussed in Section 4. Finally, Section 5 concludes the paper and indicates some future research directions.

2. RELATED WORK

Feature selection is the process of selecting an optimal subset from original features. In a NER system, an optimal subset of features is the smallest subset that most contribute towards accuracy. The smaller size of the feature subset is, the faster the NER system runs. However, the size of feature's subsets is not proportional with the accuracy. Therefore, this is an optimization problem of multiple objectives.

Carrying out an exhaustive search in the feature space has a high computational cost, since the total search space for M features is 2^M, which is a NP-hard problem. Therefore, heuristic search is often used to deal with this problem.

Two approaches often used in single solution are forward selection [15,16] and backward selection [1,8]. The forward selection starts with no feature in the model. It tests the addition of each feature using a chosen model comparison criterion and adding the feature (if any) that improves the model the most. This process repeats until none improves the model. The backward selection starts with all candidate variables. It tests the deletion of each variable using a chosen model comparison criterion and deleting the variable (if any) that improves the model the most by being deleted. This process repeats until no further improvement is possible. The disadvantage of these strategies is, the searching process cannot go back to previous features. Therefore, if a feature is already removed, it cannot be reconsidered anymore. This treatment is not good in case some features are correlated in recognizing an entity.

Two methods often used in multiple solutions are filter and wrapper. Filter techniques [3,6] evaluate the relevance of features and rank them based on certain statistical criteria. The features with the highest ranking values are selected and the low scoring features are removed. Earlier filter-based methods evaluated features in isolation and did not consider correlation between them. Recently, methods have been proposed to select features with minimum redundancy. The methods proposed use a minimum redundancy-maximum relevance feature selection framework. They supplement the maximum relevance criteria along with minimum redundancy.

Wrapper methods [5,9,18] use a classifier to score feature subsets based on their predictive power. SVM-RFE (Support Vector Machine Recursive Feature Elimination) [18], a wrapper method applied to cancer research, uses a backward feature elimination scheme to recursively remove insignificant features from subsets of features. In each recursive step, it ranks the features based on the amount of reduction in the objective function. Bottom ranked features are then eliminated from the results. Many experimental results have proved that the wrapper methods can yield better performance, although they have the disadvantage of high computational cost.

Genetic algorithm [4,10,12] is a multiple solution that is often considered in feature selection. It can be implemented as either a filter approach or a wrapper one. Feature selection using genetic algorithm is performed by combining different objectives into one formula. Genetic algorithm does not evaluate each feature separately, it can thus prevent the problem in forward and backward methods. The disadvantage of genetic algorithm is that its complexity is very high. The total number of feature subsets it has to consider is 2^M in the worst case with M is the number of features. More ever, computing the fitness of an individual is also time consuming. In case of NER, it is the total time of training the system, testing it with another set of data, and computing the system's accuracy.

To reduce the time of computing the fitness of individuals, Lanzi [12] 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.

Umamaheswari and Radhamani [19] used fuzzy rough set and genetic algorithm for feature selection. They applied a roughest algorithm to select an optimal feature subset and then applied a genetic algorithm to select another one. The union of these two subsets is used for classification. The classification consists of multi linear discriminent analysis and support vector machine. Classification is done on the base of parameter extracted by gray level co-occurrence matrix and histogram texture feature extraction method.

Parsi et al. [14] proposed swap training method in a genetic algorithm for feature selection in a face recognition system. K-Nearest Neighbor is used as the NER algorithm in this approach. In each iteration of genetic algorithm, the system switches the training and test data with each other in order to prevent converging to local minimums. Obtained results from implementing the proposed technique on Yale Face database show performance improvement of genetic algorithm in selecting proper features.

This paper is a research on using GA to select a good subset of features used in named entity recognition. Background of GA and NER algorithms used in GA to evaluate the fitness of individuals are introduced next.

3. BACKGROUND

3.1 Genetic algorithm

A genetic algorithm [7] is a method for solving optimization problems that is based on natural selection. In a genetic algorithm, a population of string, 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 been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

In GA for NER, each feature subset is an individual in a population and is encoded by a string of bits. The length of string corresponds to the total number of features considered in NER task. Bit's value 1 means the feature is selected and 0 if vice versa.

The basic flowchart of the GA system for feature selection is shown in Figure 1 below.

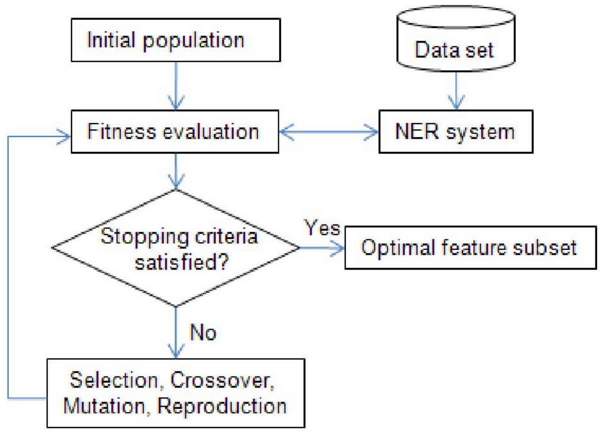

Figure 1. The flowchart of the genetic algorithm for feature selection

The fitness of an individual in GA is evaluated by using machine learning algorithm for NER. In our research, two machine learning algorithms are tested, which are Conditional Random Fields (CRF) and k-Nearest Neighbor (kNN). A brief introduction of CRF and kNN is introduced next.

3.2 Conditional Random Fields

3.2.1 Definition of Conditional Random Fields

The theory of Conditional Random Fields is proposed by Lafferty et al. [11]. Conditional Random Fields are undirected graphical models trained to maximize a conditional probability. A linear-chain CRF with parameters Δ = {λ, ...} defines a conditional probability for a state (or label) sequence $y = y_1 \dots y_T$ given an input sequence $x = x_1 \dots x_T$ (in which T is the length of sequence) to be

$$P_\lambda(y \mid x) = \frac{1}{Z_x} \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_{t-1}, y_t, x, t) \right)$$

where Z_x is the normalization constant that makes the probability of all state sequences sum to one; $f_k(y_{t-1},y_t,x,t)$ is a feature function which is often binary-valued, but can be real-valued; λ_k is a learned weight associated with feature f_k. Large positive values for λ_k indicate a preference for such an event, while large negative values make the event unlikely.

3.2.2 Training CRF

The weights of a CRF, Δ={λ, ...}, are a set to maximize the conditional log-likelihood of labeled sequences in some training set, $D = \{(x,1)^{(1)},(x,1)^{(2)}, \dots, (x,1)^{(N)}\}$:

$$L_\Delta = \sum_{j=1}^{N} \log(P_\Delta(l^{(j)} \mid x^{(j)}) - \sum_k \frac{\lambda_k^2}{2\delta^2}$$

in which the second sum is a Gaussian prior over parameters (with variance δ) that provides smoothing to help cope with sparesity in the training data.

When the training labels make the state sequence unambiguous (as they often do in practice), the likelihood function in exponential models such as CRF is convex, so there are no local maxima, and thus finding the global optimum is guaranteed. It has recently been shown that quasi-Newton methods, such as L-BFGS, are significantly more efficient than traditional iterative scaling and even conjugate gradient [13].

3.2.3 Inference in CRF

Inference in CRF is to find the most probably state sequence y* corresponding to the given observation sequence x.

$$y^* = \arg\max_{y^*} p(y \mid x)$$

In order to find y*, one can apply the dynamic programming technique with a slightly modified version of the original Viterbi algorithm for HMMs.

As far as we known, CRF is one of the best machine learning method for NER. The drawback of CRF is the high complexity of its learning algorithm. To recognize named entities from text, Stanford Named Entity Recognizer [17], written by Finkel and his colleges, is used in our system. SNER is a CRF toolkit for segmenting and labeling named entities, written in Java. By using SNER, we can redefine feature sets and specify the feature templates in a flexible way.

3.3 K_Nearest Neighbors

K_Nearest Neighbor is one of simplest classifying algorithm. In kNN, the label of a new sample is determined by the major label of its nearest K training cases. The distance between two samples in our NER task is calculated as follow:

$$D(X,Y) = \sum_{i=1}^{N} sim(x_i, y_i)$$

in which X and Y are two samples; x_i and y_i are values of feature i in the samples X and Y.

Since the time to evaluate the goodness of a feature subset using in a machine learning algorithm for the NER task includes both the training and testing time, using kNN is often faster than most machine learning algorithms. This is because kNN does not have the training process; it only needs to calculate the distance between the new sample and other sample in the training data to get its nearest k training cases. However, the accuracy of kNN is often lower than modern classification algorithms such as CRF. Since GA requites fast fitness evaluation, kNN is a good candidate for NER in GA. Therefore, kNN will be integrated in our genetic algorithm for the NER task.

In kNN, a word/phrase is recognized as a certain named entity (e.g., Person, Organization, Location) if its features's values are similar to those values of majority neighbors in its k nearest neighbors.

4. OPTIMIZING GENETIC ALGORITHM FOR FEATURE SELECTION

In this section, we propose strategies to reduce the computational cost of GA without losing the optimality of the final solution. All features considered in our NER system are introduced next.

4.1 Feature set

The NER task considered in this research is to recognize Person, Organization, Location name entities from Vietnamese text. In solving this task, 21 feature types are considered, including: *word, wordPair, firstSyllable, lastSyllable, isNumberOrTimeOrTimeMarker, initUpcaseWord, allCapWord, letterAndDigitWord, isPunctuation, isBracket, isQuotationMark, firstSentenceWord, inVnFamilyName, inVnGivenName, inVnMiddleName, inLocDict, inOrgDict, perIndicateNoun, locIndicateNoun, orgIndicateNoun,* and *pos*. Vietnamese language is monosyllable. Its vocabulary has many compound words such as "*bến tàu*$_{\text{pier}}$", "*bến xe*$_{\text{car park}}$" in which the first syllable (e.g., "*bến*$_{\text{parking place}}$") is a large concept, whereas the second syllable is a specific kind of that concept (e.g., "*xe*$_{\text{car}}$"). Another type of compound words (e.g., "*tuyến đường*$_{\text{route}}$", "*mặt đường*$_{\text{road}}$") has the last syllable is a concept (e.g., "*đường*$_{\text{road}}$"), whereas the first syllable is an attribute of that concept (e.g., "*tuyến*$_{\text{line}}$", "*mặt*$_{\text{face}}$"). Therefore, we propose to use firstSyllable and lastSyllable in our NER system to detect words belonging to the same group.

In general, each feature type is considered in the window size of five (two words before the current word, the current word, and two words after the current word). However, since some feature types at certain positions have no meaning in the NER task, we restrict the feature's positions as shown in Table 1.

Table 1. Features for Named Entity Recognition

Index	Feature group	Feature type	Position
1		word	0, ±1, ±2
2		wordPair	(-1 0), (0 1), (-1 1)
3	Context	firstSyllable (e.g., *bến* tàu$_{\text{pier}}$, *bến* xe$_{\text{car park}}$)	0, -1, -2
4		lastSyllable (e.g., tuyến *đường* $_{\text{route}}$, mặt *đường*$_{\text{road}}$)	0, -1, -2
5		isNumberOrTimeOrTimeMarker (e.g., 2.1, 21, 21/08/2012, 8/1995, 6am, tháng$_{\text{month}}$, năm$_{\text{year}}$)	0, ±1, ±2
6		initUpcaseWord	0, ±1, ±2
7		allCapWord	0, ±1, ±2
8		letterAndDigitWord	0, ±1, ±2
9	Orthography	isPunctuation (e.g., . ? !)	±1, ±2
10		isBracket	±1, ±2
11		isQuotationMark	±1, ±2
12		firstSentenceWord	0, -1, -2
13		inVnFamilyName	0, ±1, ±2
14		inVnGivenName	0, 1, 2
15		inVnMiddleName	0, ±1, ±2
16	Dictionary	inLocDict	0, ±1, ±2
17		inOrgDict	0, ±1, ±2
18		perIndicateNoun	0, ±1, ±2
19		locIndicateNoun	0, ±1, ±2
20		orgIndicateNoun	0, ±1, ±2
21	Part-of-Speech	pos	0, ±1, ±2

In Table 1, position 0 means the current word; positions ±1, ±2 mean the word at the position ±1, ±2 compared to the current word. Each feature is a combination of its feature type and its position. For example, there are five features with the feature type "word", including prev2Word (the word before the previous word), prevWord (the previous word), curWord (the current word), nextWord (the next word), next2Word (the word after the next word), corresponding to positions -2, -1, 0, 1, and 2, respectively. Therefore, the total number of features needed to be considered in our GA is 92.

The search space of GA is all possible combinations of these features, which is 2^{92} in this case. The time to carry out an exhausted search in the GA space is equal to the time to evaluate the fitness of an individual multiplying by the total number of individuals. Therefore, reducing the search space and the time to evaluate the fitness of an individual are crucial problems in our genetic algorithm.

4.2 Fitness Function

As mentioned in Section 4.1, the time to evaluate the fitness of an individual in a genetic algorithm should be small. Normally, the fitness of an individual (corresponding to a feature subset) is evaluated by the NER system's performance, which is F-score - a combination measure of Precision and Recall. The time to calculate this value is the total time of training and testing the NER system using 10-fold validation, which is quite long in most NER systems. To reduce the calculating time, instead of using F-score, the error rate of the NER is used to evaluate the individual's fitness[1]. It is easy to see that, the smaller the number of data items misclassified, the more suitable the feature subset.

The focus of this paper is to find the smallest feature subset for which the NER system's performance does not deteriorate below a certain level comparing to the highest one. Therefore, information about the size of the feature subset is also integrated into the fitness formula, which is shown below:

$$\text{Fitness}(i)=w1*f/k+w2*n/ne$$

in which i is the individual i in the population; f is the total number of features in the feature subset (or the total number of values 1 in the individual i); k is the total number of features (or the length of the individual i); n is the total words in the training set; ne is the number of words misclassified. w_1 and w_2 are weights of the size of the feature subset and the error rate of the classifier ($w_1 + w_2 = 1$). In our experiments, w_1 and w_2 are chosen as 0.2 and 0.8, respectively, meaning that the error rate is more important than the number of features being used.

It is easy to see that the fitness is inversely proportional to the number of features selected and the error rate of the NER system.

4.3 The proposed genetic algorithm

To reduce computational time of our genetic algorithm, the following strategies are used:

1. reduce individuals being visited
2. reduce the time of computing the fitness of an individual
3. prevent repeated fitness tests.

Solution to the problem (2) has been discussed in Section 4.2. One solution for reducing individuals being visited is to keep the population size small. However, the system will quickly converge to a local optimal solution in this case. To deal with this problem, the population size is set to be large at the beginning and reducing by the time (after one half of generations), so that the system can explore the search space at the beginning and converge to some optimal solutions at the end. To reduce the computational time at the beginning when the population size is

[1] The idea of using error rate to evaluate the individual's fitness is inherited from [2].

large, the data set using in GA is small at the beginning and increasing by the time (after one half of generations).

To prevent repeated fitness tests, the individuals' fitness is stored in the memory so that if those individuals appear in the future, the system does not need to re-evaluate them. This strategy is very important when the time to compute each fitness is long. One problem with this implementation is, when too many individuals are kept in the memory, the memory may be overflow, or it takes too long to find a specific individual. To prevent this problem, a fixed size of memory is used to store these individuals. The individual with lowest fitness will be removed from the memory when the memory is full.

The improve GA for feature selection is shown in Figure 2 below. The algorithm terminates when the population contains only one individual.

```
Input: Feature sets F = f1, f2, ..., fm
       Data set D, initial data size iniD,
       initial population size N, crossover rate
       c, mutation rate m
Output: The optimal feature subset
Algorithm:
1. Set population size curPopSize = N
2. Set data size curDataSize = iniD. Get a data
   subset with the size curDataSize from the data
   set D as the current data set.
3. Initialize population P of curPopSize
   individuals.
4. Calculate the fitness of each individual in P,
   using the current data set.
5. Set generation index G = 1
   Do
   a. Carry out selection, crossover, mutation
      operations, resulting in a new population
      P1
   b. If G = N/2 then
        i.curPopSize = N/2; N= curPopSize;
       ii.curDataSize = $\sqrt{2}$ * curDataSize. Get
          a data subset with the size
          curDataSize from the data set D as the
          current data set. This subset does not
          contain the old data set in the
          previous step.
      iii.Update the fitness of old individuals
          in the current population by the
          formula:
          Fitness(i)= (oldFit(i)+
          $\sqrt{2}$ *newFit(i))/(1+ $\sqrt{2}$)
          in which Fitness(i) is the fitness of
          the individual i in the current
          population; oldFit(i) is the fitness
          of the individual i in the old data
          set; newFit(i) is the fitness of the
          individual i in the current data set.
          The weight of newFit(i) is increased
          by sqrt(2) to reflex the new size of
          the data set. The fitness of the
          individual i in the old data set is
          retrieved from previous runs.
   c. Calculate the fitness of new individuals
      in P1, using the current data set.
   d. Generate a new population P from P1, with
      population size curPopSize. In other
      words, the algorithm selects curPopSize
      individuals with the highest fitness in
      P1.
   e. Generation G = G + 1
6. While(curPopSize>1)
7. Return P
```

Figure 2. The improve GA for feature selection

5. EXPERIMENTAL RESULTS

As far as we know, there is no public data set available for the NER task in Vietnamese. Therefore, we have to create a new data set by our own. Our data set are documents taken from newspaper websites on economics, politics, cultures and education that were annotated manually with seven labels B-per, I-per, B-loc, I-loc, B-org, I-org, and O (other). This data set consists of 8242 sentences among which 5881 sentences (consisting of 170230 words) were used as the training set and 2361 sentences (consisting of 62000 words) were used as the test set.

The GA system was tested on two machine learning algorithms for NER (CRF and kNN). By applying the GA learning process with a training set of 2176 sentences (consisting of 60218 words) and a test set of 1071 sentences (32416 words), we obtained a feature subset. The whole training data set was then used to train the NER algorithms on that feature subset to obtain a prediction model. The NER system's error rate was calculated on the whole test set.

In the learning process, a population size N was chosen at the beginning. The initial training data size was set to 232 sentences (3000 words). The initial test data size was set to 121 sentences (2000 words). After N/2 generation, the population size was reduced by a half (N = N/2). The test data size was increased by sqrt(2).

After the optimal feature subset was obtained, the F-score of the NER system is calculated using this feature subset.

The kNN algorithm was tested with the number of neighbors being equal to 15. When using all features (92 features), the F-score of the NER system was 91.66% for kNN and 92.33% for CRF. Experimental results when using GA-kNN and GA-CRF for feature selection are shown in Table 2 below. The final row of Table 2 displays the size of the feature subset returned by GA.

Table 2. Experimental results when using GA-kNN and GA-CRF for feature selection

NER algorithm	kNN			CRF		
Initial population size	16	32	64	16	32	64
F-score of the NER system (%)	92.93	92.79	92.65	94.23	94.23	94.23
Number of features	33	32	38	39	41	44

Experimental results in Table 2 prove that we can use a feature set smaller than the original one in the NER system without losing the system's accuracy.

The optimal subsets when using kNN are smaller than a half of the original feature set and give us higher accuracies than the original feature set. It indicates that several features in the original feature set are redundant. Furthermore, some features are even degraded the NER system's accuracy. The best feature subset when using kNN is the subset of 33 features, generated by the GA with the initial population size of 16 individuals. The F-score of the NER system in this case is 92.93%.

The feature subsets returned by GA-CRF system are also smaller than a half of the original feature set. The F-score when using these subsets increase by approximately 2%, comparing to the accuracy when using the original data set. The best feature subset in this case contains 39 features, resulting in the F-score of 94.23% of the NER system. It indicates that CRF provides higher accuracy than kNN.

To evaluate the effect of GA to feature selection, two other feature selection methods were tested, which are hill climbing and backward ones. The hill climbing algorithm is a kind of forward selection. In the hill climbing algorithm, the initial feature is chosen randomly. Other features are considered to be added to the feature subset one by one. A feature is kept in the subset if the F-score of the NER system increases when it is added to the feature subset. The hill climbing algorithm terminates when no further improvements can be found. The backward selection works in the reverse direction. It starts with all variables and then continuously removes features that increase the error the most until no further improvements can be found.

In our experiment with the hill climbing algorithm, kNN was used as the NER algorithm. Experimental results with the hill climbing algorithm are given in Table 3. Since the initial feature was chosen randomly, three different tests with the hill climbing algorithm gave us three different feature subsets.

Table 3. Experimental results when using the hill climbing algorithm for feature selection

Run	**1**	**2**	**3**
F-score of the NER system (%)	91.21	91.58	91.50
Number of features	17	11	9

The sizes of feature subsets returned by the hill climbing algorithm in Table 3 are small. However, the F-scores of the NER system are lower than those of the feature subsets generated by GA. This is because the hill climbing algorithm does not consider the correlation between features. If a feature is removed, it cannot be reconsidered anymore. Therefore, the hill climbing algorithm may ignore good feature subsets in recognizing named entities. As a result, the F-score of the NER system in this case is not high. The best feature subset generated by the hill climbing algorithm consists of 11 features, corresponding to the F-score of the NER system of 91.58%.

The backward algorithm starts with all features and then continuously removes features that reduce the F-score of the NER system the most. Experiments with the backward algorithm using kNN as the classifier result in a subset of 82 features, which is much larger than feature subsets produced by other algorithms considered in this paper. The F-score of the NER system in this case is 91.66%, approximately to the F-score of the NER system when using the hill climbing algorithm.

It is interesting to see that our proposed features (firstSyllable and lastSyllable) appear in all optimal subsets returned by our experiments, meaning that they are important features in NER for Vietnamese text.

Experiment results show that using GA in feature selection for NER task provides better results compared to hill climbing algorithm and backward algorithm. By using some strategies to reduce computational cost of GA, GA can be a potential solution for feature selection task.

6. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a genetic algorithm to automatically select a minimal feature subset that can produce high accuracy for the NER system. To optimize the GA learning process, several strategies have been proposed, including: (i) change the population size and the data size during the GA learning process; (ii) reduce the time of computing the fitness of an individual; and (iii) prevent repeated fitness tests during GA learning process. Experimental results show that GA can generate a feature subset which is smaller than a half of the original one, but with a higher F-score of the NER system. In addition, GA provides better performance than the hill climbing and backward algorithms.

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 the previous generations of GA.

7. ACKNOWLEDGMENTS

This work was supported by the Vietnam National Foundation for Science and Technology Development (NAFOSTED), under Grant 102.01-2011.08.

The work of the first author was partly funded by the Vietnam Ministry project, under Grant B2012 - 01 - 24.

8. REFERENCES

[1] Abe, S. 2005. Modified backward feature selection by cross validation. In *Proceedings of ESANN 2005*, pp.163-168.

[2] Bhanu, B., Lin, Y. 2003. Genetic algorithm based feature selection for target detection in SAR images. *Image and Vision Computing*. Volume 21, pages 591–608

[3] Dash, M., Choi, K., Scheuermann, P., Liu, H. 2002. Feature selection for clustering - a filter solution. In *Proceedings of the 2002 IEEE International Conference on Data Mining* (ICDM 2002), pages 115–122.

[4] Day, M., Lu, C., Ong, C., Wu, S., and Hsu, W. 2006. Integrating genetic algorithms with conditional random

fields to enhance question informer prediction. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IEEE IRI 2006)*, pp. 414-419.

[5] Dy, J. G., Brodley, C. E. 2004. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889.

[6] Hall, M. A. and Smith, L. A. 1997. Feature subset selection: a correlation based filter approach. In *Proceedings of International Conference on Neural Information Processing and Intelligent Information Systems* (pp. 855-858). Berlin: Springer.

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

[8] Kabir, M.M., Shahjahan, M., Murase, K. 2007. A Backward Feature Selection by Creating Compact Neural Network Using Coherence Learning and Pruning. In *Proceedings of JACIII 11,* pp. 570-581.

[9] Kim, Y., Street, W. N., Menczer, F. 2002. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6(6):531–556.

[10] Kitoogo, F.E. & Baryamureeba, V. 2007. A Methodology for Feature Selection in Named Entity Recognition. In *Special Topics in Computing and ICT Research: Strengthening the role of ICT in Development.*

[11] Lafferty, J., McCallum, A. and Pereira, F. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282-290.

[12] Lanzi, P. 1997. *Fast feature selection with genetic algorithms: a filter approach.* In *Proceedings of IEEE Intl. Conf. on Evolutionary Computation*, pp.537-540

[13] Malouf, R. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of Sixth Workshop on Computational Language Learning (CoNLL-2002).*

[14] Parsi, A., Salehi, M., and Doostmohammadi, A. 2012. Swap Training: A Genetic Algorithm Based Feature Selection Method Applied on Face Recognition System. *World of Computer Science and Information Technology Journal* (WCSIT). 2(4):125-130

[15] Pahikkala, T., Airola, A., Salakoski, T. 2010. Speeding Up Greedy Forward Selection for regularized Least-Squares. *In Proc. of the Ninth International Conference on Machine Learning and Applications*, ICMLA 2010, pp. 325-330.

[16] Schaffernicht, E., Moeller, C., Debes, K., Gross, H.-M. 2009. Forward feature selection using Residual Mutual Information. In *Proceedings of 17th European Symposium on Artificial Neural Networks, ESANN 2009*, pp. 583-588.

[17] SNER 2012. Stanford Named Entity Recognizer. http://nlp.stanford.edu/software/CRF-NER.shtml. (last visited Aug. 2012).

[18] Tang Y., Zhang Y.Q., Huang Z. 2007. Development of two-stage SVM-RFE gene selection strategy for microarray expression data analysis. In *Proceedings of IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Volume 4, pp. 365-81.

[19] Umamaheswari, J. and Radhamani, G. 2011. A Hybrid Approach for Classification of DICOM Image. *World of Computer Science and Information Technology Journal (WCSIT)*, 1(8):364-369.