A Rule-Based Model and Genetic Algorithm Combination for Persian Text Chunking

Samira Noferesti* and Mehrnoush Shamsfard*
Shahid Beheshti University, Tehran, IRAN

Abstract

This paper introduces a hybrid approach to Persian text chunking. The approach is that of two methods applied in sequence; a rule-based method and a genetic algorithm for shallow parsing. At the first phase, the rule-based model was applied to assign IOB tags to some tokens of an input sentence. Then, in the second phase, the search capabilities of the genetic algorithm were used to assign untagged tokens to corresponding chunks using the previously captured training information. The proposed algorithm was evaluated on the Peykareh corpus and it achieved 94.82 percent accuracy. Test results show that this proposed algorithm outperformed other algorithms for Persian text chunking.

Key Words: Rule-based model, genetic algorithm, chunking, shallow parsing, Persian language.

1 Introduction

Shallow parsing, also known as chunking or light parsing aims at discovering the main non-overlapping constituents of sentences such as noun phrases and verb phrases. It is a well-known tool used for natural language processing. It is commonly applied in many areas such as information extraction, bilingual alignment, summary generation, semantic role labeling, machine translation and named entity recognition.

Although shallow parsing has been extensively studied in other languages, investigation of the available literature revealed only two previous studies on Persian text chunking: one of which applied a rule-based algorithm of hand crafted rules [21] and the other a rule-based method to create a tagged corpus for training a neural network followed by a multilayer neural network and Fuzzy C-Means clustering, to chunk new sentences [13].

The aim of this paper was to introduce an approach for Persian text chunking using the combination of a genetic algorithm and a rule-based model. Genetic algorithms detect general chunk patterns in text and rule-based models handle special cases and exceptions to patterns in text. For the rule-based model, linguistic rules of [21] (termed general rules), were extended by analyzing errors of the genetic algorithm and defining new rules for handling these errors (termed correction rules). Although the genetic algorithm is capable of handling most of the general rules, application of these newly defined correction rules was made to speed up the work of the genetic algorithm. The proposed algorithm for text chunking contained two phases. The first phase used the rule-based model to identify some chunks. Then the second phase made good use of the search capabilities of the genetic algorithm to assign untagged tokens to corresponding chunks. Therefore, a faster genetic algorithm was achieved by producing more tagged tokens that had been generated from the rule-based model.

There is a remarkable amount of ongoing research on applying machine learning approaches to text chunking in the English language. Most machine learning approaches, such as those methods based on hidden Markov models, use information extracted from a tagged corpus to assign a suitable tag to each word according to preceding tags. Since these approaches are purely statistical, as such they are most suitable for cases that have a corpus large enough to contain all possible combinations of n-grams. In contrast, evolutionary algorithms offer a more generalized method that can be applied to any statistical model. For example, they can be applied to perform tagging operations according to the Markov model (tag prediction for a current word based on preceding tags) or improve the Markov results by using more contextual information (for example, using tags of preceding words or those of following words). In other words, HMM or other models can be used as part of the fitness function in a genetic algorithm. Therefore, a genetic algorithm provides more flexibility than any of the other classical approaches such as HMM based methods.

Accurate results have been obtained by applying evolutionary algorithms for POS tagging and stochastic tagging algorithms. According to [3] evolutionary algorithms have the advantage of being applicable to any statistical model without the need to rely on the Markov assumption (i.e., the tag given to a word is dependent only on previous words), as is the requirement in classic search algorithms for tagging, such as the widely used Viterbi algorithm [12]. Thus a hybrid approach for shallow parsing was chosen for this study. Results of the tests in this study show that our proposed algorithm outperformed other algorithms for Persian text chunking as well as the classical HMM based method.

In the following part of this paper, a brief review of the

* Electrical and Computer Engineering Department. Email: snoferesti@ece.usb.ac.ir.

related studies is discussed in Section 2. Section 3 describes the task of chunking. Section 4 introduces the annotated corpus of Persian texts. Section 5 explains our chunking model. Experiment results are discussed in Section 6. Finally, Section 7 concludes the paper.

2 Related Work

Sentence chunking was first introduced in [1]. Since then, chunking has been studied extensively. In [17] chunking algorithms were classified according to two main groups: the linguistic approach that is based on handwritten linguistic rules and learning approaches that are mostly corpus driven. Although the pioneered works were focused on rule-based methods [2], with the development of chunking corpus, most previous works applied various kinds of machine learning algorithms to the chunking task such as a support vector machine [23], conditional random field [18], transformation based learning [19], memory based learning [10, 9], decision trees [16]; the hidden Markov model [6] and hybrid methods have also been proposed [24, 7].

There are also evolutionary algorithms for the task of chunking [20, 4, 5]. In these methods, chunk classification has been seen as an optimization problem in which the best chunk tags should be assigned to the words of a sentence. As we are exploiting a genetic algorithm, the closest work to this one is that of Atkinson and Matamala [4, 5], with the difference of choosing the fitness function and involving data driven lexical information into the decisions. In fact, tests were done using the method for text chunking Persian to assess the potential of the algorithm but the method did not achieve good results. Tests showed that results improved by combining a structure based fitness function with the addition of a data driven function to calculate the probability of a specific word being inside or outside of a chunk. Tests indicated that the new proposed fitness function performed better for text in the Persian language. In addition, a genetic algorithm was combined with a rule-based model and results showed that this hybrid algorithm produced improved results.

3 Description of the Chunking Task

A chunk is a syntactic structure, which groups several consecutive words to form a phrase. The phrase structures of a sentence can be encoded using IOB [19] or IOB2 [22] styles. In the IOB tagging style, each token is tagged with one of three specific chunk tags; I (inside), O (outside), or B(begin). In the IOB2 style, which was used in this paper, the IOB tagging scheme was merged with chunk type to create a new extended tag set. For example B_NP means that the word was the first word of a noun phrase. In this work, seven chunk types were defined: noun phrase (NP), verb phrase (VP), adverb phrase (AP), adjective phrase (JP), prepositional phrase (PP), object phrase (RP) and others (O).

4 The Corpus and Tag Set

An annotated corpus of Persian text is needed in order to train and evaluate the chunker. This corpus must be annotated with POS and chunk tags. Each word in a sentence must be assigned a tag that indicates whether the word is inside or outside a chunk and shows the appropriate type of corresponding chunk.

For the current work, a subset of Persian POS tagged corpus was used, known as Peykareh[1] [8]. This collection was gathered from daily news and common texts and contained about 2.6 million manually tagged tokens. The main corpus was tagged with a rich set of POS tags consisting of 550 different tags from which 21 tags were selected for the system. Those that could be detected by the present Persian POS taggers were selected for use in the system applied to this study.

Only a portion of the Peykareh corpus containing 128,800 tokens had IOB tags. This IOB tagged dataset was created in NLP Lab-Shahid Beheshti University [14]. The chunked corpus was divided into three sets: (1) a training set including 91,933 tokens, (2) a chunked held-out data set containing 15,604 tokens and (3) a test set containing 21,263 tokens. A further held-out data set was used to define rules in the rule-based model. In fact, a portion of the Peykareh corpus was set aside that had no IOB tags and it was called the unchunked held-out data set.

5 The Proposed Algorithm

This paper proposes a hybrid approach for chunking Persian text. First, a rule-based chunker was developed to tag as many words as possible. In this stage, parts of each given sentence were assigned IOB tags. Then, a genetic algorithm was run for the tokens, which had not been assigned an IOB tag in the previous stage.

5.1 The Rule-Based Model

Shamsfard and SadrMousavi [21] presented a rule-based model for chunking Persian text. The rule-based part of this work incorporated some of the rules cited in [21]; (30 rules or less than 25 percent of rules) but in many cases the rules were changed; some rules were eliminated and some new ones (105 rules) were introduced according to the different tag sets. Some of the new rules were defined to handle errors of the genetic algorithm. Twelve files of the Peykareh corpus were used, and these included 42,006 tokens as the unchunked held-out data set. This data set was different from the chunked data corpus used for training and testing. To define the rules, the initial rule-based model and the genetic algorithm were run on the held-out data and errors were analyzed to introduce rules that would fix them. For example, in most cases a preposition was the first token of a PP chunk. Thus, a rule was identified in the initial set of rules to show that some exceptions were observed during error analysis. In Persian some compound verbs consist of prepositions or particles before or after the main verb. In fact, a preposition is part of a compound verb,

[1] This corpus also is known by its author's name, Bijankhan.

but in the Peykareh corpus it was tagged as a preposition. In such cases, the preposition must be inside the verb phrase. For example, if the previous word is the verb عبا رتند[2] (ebã:rætænd) 'include', the preposition must be inside the VP chunk.

In this way, a set of 135 hand-crafted rules was developed. Then, the most suitable sequence of rules was determined in terms of avoiding bleeding and creeping; n fact, a tree was constructed. The first level of the tree contained some general rules. Level 2 consisted of some rules for handling exceptions of the first level and so it continued. However, each node in level i handled an exception of the rule of its parent node in level i-1 (if that rule had exceptions).

At the first stage of the proposed algorithm, each rule was taken individually from the rule-set one at a time and the function was performed only if the rule was applicable to the input word.

Rules were categorized according to three classes: syntactic, morphological and lexical. Syntactic rules used part-of-speech to tag words (either as the target word or a neighboring word) in a sequence to determine the IOB tag, while morphological rules considered internal and morphological structures of a word to do this task, and the lexical rule considered real words.

Frequency counts for the rule-categories are shown in Table 1.

Table 1: Frequency counts for the rule-categories

Syntactic	morphological	Lexical
55	9	71

In the rest of this section these categories are discussed in more detail and some examples are given from each category.

5.1.1 Syntactic Rules. Some POS categories enforced a special IOB tag on words or on neighboring words. The accusative case marker ر ا (rã:), punctuation marks, prepositions, and auxiliary verbs were among this set.

— Punctuation

At the first stage of the algorithm all punctuation marks were assigned the O tag by the following rule. However, the IOB tag of some punctuation marks such as commas could be changed during the algorithm.

If POS(X) =PUNC Then IOB_Tag(X) = O

— Accusative case marker ر ا (rã:)

- The Persian language has an accusative case mark ر ا (rã:) that follows the direct object. Although sometimes direct objects can appear without a marker, the marker never follows anything other than the object. For the purpose of this study, it was supposed that an accusative case marker should be put inside the chunk of the object and termed object phrase (RP). An alternative would have been to split RP into two chunks NP or AP with another chunk for the accusative case marker. We used the first alternative as it was compatible with the chunk-annotated corpus. The following rule dictated that the accusative case marker, which was shown in the corpus with the POS tag POSTP, was always inside the chunk.

If POS(X) =POSTP Then IOB_Tag(X)=I

- The word after an accusative case marker ر ا (rã:) was determined as either the first token of the next chunk or as an outside token. In other words, if the word after the accusative case marker was not a punctuation mark with an O tag, then it would be the first token of the next chunk.

If POS(X)=POSTP And IOB_Tag(X+1)!=O
Then IOB_Tag(X+1)=B

— Prepositions

- In the Persian language the head of a prepositional phrase is always a preposition, which is followed by an NP. The following rule showed that if a current token was a preposition, then the next token would be inside the phrase.

If POS(X) =P Then IOB_Tag(X+1) =I

- In some cases a preposition was attached to the next pronoun and the whole combination was tagged as a preposition. These cases are exceptions of the above rule. For example, the word بـر ا یـم (bærã:jæm) 'For me' is an abbreviation of بـر ا ی مـن (barã:jemæn), in which بـر ا ی (bærã:je) 'for' is a preposition and مـن (mæn) 'I'is a pronoun. Thus, the rule mentioned above was changed as follows:

If POS(X)=P and postfix(X)! =pronoun
Then IOB_Tag(X+1)=I

— Auxiliary verbs

- In a verb phrase in the Persian language, an auxiliary verb is placed before the main verb and so the main verb is inside a chunk.

If POS(X) =AUX and POS(X+1)=V
Then IOB_Tag(X+1) =I

[2]For each Persian word or phrase we write its transliteration within parenthesis and its English meaning within single quotes. International Phonetic Alphabet (IPA) is used to represent Persian language pronunciations.

For example, in the sentence به مدرسه خواهم رفت من فردا 'I will go to school tomorrow.', the auxiliary verb خواهم (xã:hæm) 'will' appeared before the main verb رفت (ræft) 'went' and together they made a verb phrase. Thus the main verb was tagged as I.

- An auxiliary verb can sometimes appear alone without a main verb after it. In such cases, an auxiliary verb was assigned as the first token of the chunk and the next token was the first token of the next chunk or outside the chunk. This rule is written as follows;

If POS(X) =AUX and POS(X+1)!=V And IOB_Tag(X+1)!=O Then IOB_Tag(X)=B And IOB_Tag(X+1) =B

For example, in the sentence باید به من قول بدهی تو 'You must promise me', the auxiliary verb باید (bã:jæd) 'must' is alone and gets the IOB tag B. In addition, the next token is tagged as B.

5.1.2. Morphological Rules. The following rules are examples of morphological rules that determine structures that take the special IOB tag.

— When a number is shown with a POS tag NUM in corpus ends in the postfix مین (mi:n), it means that it is an ordinal number. The following rule was written since the ordinal number and a proceeding noun are placed in the same chunk;

If POS(X)=NUM and postfix(X)=مین Then IOB_Tag(X+1)=I

For example, in the sentence کتاب را خواندم من سومین 'I read the third book.', the word کتاب (ketã:b) 'book' appears after the ordinal number سومین (sevomi:n) 'third' is inside the chunk.

— Another way to construct an ordinal number is by adding the letter م (mi:m)at the end of a number. This kind of ordinal number often appears after nouns. Thus, if the last character of a number is م (mi:m) and the previous token is a noun, then the number will be inside the chunk. For example, اتاق دوم (otã:gedovoum) 'second room' in a sentence dictates that the ordinal number should be inside the chunk.

If POS(X)=NUM and postfix(X)= م and POS(X-1)=NOUN Then IOB_Tag(X)=I

5.1.3 Lexical Rules. Lexical rules consider real form of words as shown in the following examples:

— Conjunctions

- Since, conjunctions like و (væ) 'and' join two phrases of the same syntactic type, if there is a و (væ) 'and' in a sentence and the previous and following words have the same POS tag, these words are in the same chunk.

If WORD(X)= و and POS(X-1)=POS(X+1) and POS(X+1)!=V Then IOB_Tag(X)=I and IOB_Tag(X+1)=I

The above rule excepts the POS tag verb. The reason is that sometimes a sentence that has only a single verb appears after the conjunction و (væ) 'and'. In these cases, the single verb is the first token of the chunk. For example, in the sentence کتاب را به او دادم و رفتم 'I gave the book to him and went.', the verb رفتم (ræftæm) 'went' is a complete sentence and it takes the IOB tag B.

- If we have *n* structures with the same POS tags separated by a comma except for the last two structures, which are separated by, the conjunction *and*, then they will be in the same chunk.

— Verbs

- To detect verb phrases, a database of compound verbs in the Persian language was used. The database of compound verbs was produced in the NLP Lab, Shahid Beheshti University and contained about 5,000 compound verbs in Persian. Whenever there was a verb in a sentence, some of the previous words were considered. If this sequence of words existed in the compound verb database, then they were tagged as a VP chunk.

— Prepositions

- If the word از (æz) 'from' appeared after a comparative adjective shown with a POS tag AJCOMP in corpus, it would be placed inside the chunk. This was shown with the following rule:

If WORD(X)= از and POS(X-1)=AJCOMP Then IOB_Tag(X)=I

After running the rule-based model, some of the tokens remain untagged. Thus, a genetic based algorithm was used to identify the remaining chunks.

5.2 Genetic Chunking Algorithm

The proposed genetic algorithm receives a natural language sentence and assigns a corresponding chunk according to previously computed training information from the annotated corpus. Formally, given a sequence of n words and corresponding POS tags, the aim is to find the most probable chunk sequence.

In our implementation, each gene can take values: I or B. Tokens, which need the IOB tag O were determined by the rule-based model. Individuals of the first generation were produced randomly. After producing an individual, all tokens of a given sentence were assigned IOB tags (some of tokens get IOB tag by the rule-based model and others get IOB tag by the genetic algorithm). The next step was to assign a type to each bounded phrase. This was done automatically by applying the following rules:

Default chunk_type =NP
If POS(first_token)=P Then chunk_type=PP
Else If POS(first_token)=ADJ Then chunk_type=JP
Else If POS(first_token)=AUX or POS(first_token)=V Then chunk_type=VP
Else If POS(first_token)=ADJ,COMP or POS(first_token) =ADJ,SUP Then chunk_type=NP
Else For each token X in the chunk
If POS(X)=ADV Then chunk_type=AP
If POS(last-token)=RA Then chunk_type=RP
For each token X in the chunk
If POS(X)= V Then chunk_type=VP

An initial population was created randomly by assigning a random IOB value to each untagged gene (some genes were assigned IOB tags from the rule-based model). These individuals were sorted according to fitness value of individuals from high to low.

Three genetic operations were used for producing the next generation.

— Selection: All individuals in the population are sorted according to fitness, so the first individual was the best fit in the generation. To perform crossover, the *i*th and *(i+1)*th individuals of the current generation were selected, where i=1,2,...,[(p+1)/2] and p was the population size. The aim of selection was to choose the fitter individuals.
— Crossover: Selected two chromosomes, crossover exchanges portioned of a pair of chromosomes at a randomly chosen point called the crossover point.
— Mutation: Selected an untagged gene randomly and toggled its value, for example if its value was *B,* it was reset to *I* and vice versa.

5.2.1 Fitness Functions. To evaluate the quality of chunks generated for an individual, four functions were used; F1, F2, F3 and F4. These functions considered the context in which a word appeared. Context consisted of a current word, one tag to the left and another to the right and the previous word.

F1 considered the sequence of POS tags for each chunk of a sentence. The required features for computing F1 are POS tags (as provided by Peykareh) of the previous word, the next word and the current word and type of current chunk. The probability of the sequence of POS tags for chunks of a sequence of n words was as follows:

$$F1 = \sum_{j=1}^{Nc} \sum_{i=1}^{L(j)-1} P(POS_{i,j^t} \mid POS_{i-1,j^t}, POS_{i+1,j^t}) \text{ t} \quad (1)$$

Where, $P(POS_{i,j^t} \mid POS_{i-1,j^t}, POS_{i+1,j^t})$ represents the probability that given the current chunk j with type t, the POS_i tag occurs when the previous word in the j-th chunk had the POS_{i-1} tag and the next word in the j-th chunk had the POS_{i+1} tag. Nc was the number of chunks in a sentence and $L(j)$ was the length of chunk j.

F2 function computed the probability of taking IOB tag x if the current word was $word_i$ and the previous word was $word_{i-1}$.

$$F2 = \sum_{i=1}^{n} P(IOB\_Tag_x \mid word_i, word_{i-1}) \quad (2)$$

Where, n was the number of tokens in an input sentence. In order to compute F1 and F2 functions, the HMM model can be used with the Viterbi algorithm.

For computing F3 function, a data driven approach was applied to calculate the probability that a specific word has a special IOB tag.

$$F3 = \sum_{i=1}^{n} P(IOB\_Tag_x \mid word_i) \quad (3)$$

Where, $P(IOB\_Tag_x \mid Word_i)$ represents the probability of i-th token in the sentence has IOB_Tag_x.

F3 is defined because some words in Persian were mostly assigned a special IOB tag. For example, the word شاید (shã:jæd) 'maybe' appeared as the first token of a chunk in most cases.

F4 function was the probability that a token gets IOB tag x when it occurs after a specific word in the training corpus.

$$F4 = \sum_{i=1}^{n} P(IOB\_Tag_x \text{ for token}_i \mid word_{i-1}) \quad (4)$$

This function was defined because some words in Persian enforced special IOB tags for the next token. For example, the word به استثناء (be estesnã:e) 'except' may be enforced such that the next token is placed inside the chunk.

The effect of each of these functions is assessed in Section 6. The following fitness function was used to evaluate the genetic algorithm:

$$Fitness\ Function = \sum_{i=1}^{4} w_i \cdot F_i \quad (5)$$

Where $w_i s$ are constant parameters chosen from [0,1] and show relative importance of syntactic and lexical information. It was assumed that 0 is a legal value to show the effect of removing one or more functions from the formula. To adjust w_i parameters in the fitness function formula, variable

structure-learning automata were applied on chunked held-out data. For more information you can see [15]. Finally, the value of w_1 was set to 0.3 and the values of w_2, w_3 and w_4 were set to 1.

6 Experiments

To evaluate the performance of our proposed algorithm, three measures were taken [25]: accuracy (the percentage of correctly predicted output classes containing all types of chunks included O-class), precision (the percentage of predicted chunks that were correct) and recall (the percentage of predictable chunks that were found). A chunk was only counted as correct when its boundaries, its type and its function were all identified correctly.

Since performance was related to both precision and recall, the *F*-measure was given as the final evaluation.

$$F_{measure} = \frac{2.precision.recall}{precision + recall} \quad (6)$$

6.1 Tuning Parameters of the Genetic Algorithm

The efficiency of a genetic algorithm greatly depends on how its parameters are tuned. To adjust the genetic parameters selected, a set of 15,604 tokens were selected, as the chunked held-out data set. Then, the proposed algorithm was run on this set.

Beginning with a baseline configuration, such as Dejong's setting [11] with 1,000 generations, 50 chromosomes in each generation and 0.6 for crossover probability, the algorithm was run for different mutation probabilities (P_m) from 0.01 to 0.3. Figure 1 shows that the best results were obtained using the mutation probability 0.01.

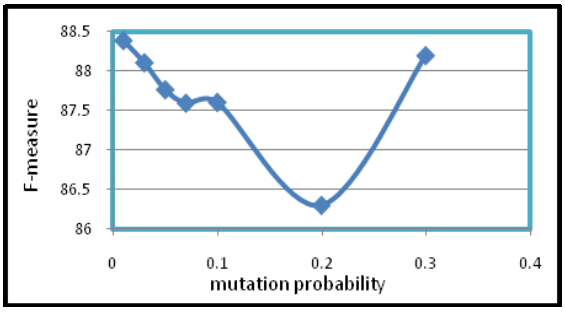

Figure 1: Average fitness values of executions of the GA using different mutation probabilities

In the same way, crossover probability was set to 0.4. In Figure 2 the results of running the genetic algorithm using mutation probability 0.01, crossover probability 0.4, population size 50 and different number of generations are shown.

Table 2 shows the optimum values of genetic algorithm parameters.

A further experiment was done to assess the effect of each part of the fitness function on chunker performance. As mentioned before, the fitness function consisted of four functions. Function F1 considered syntactic information and others considered lexical information. To assess the effect of these functions, lexical functions were removed one by one and the proposed algorithm was run with the new fitness function. Table 3 shows the effect of different fitness functions on performance of the proposed algorithm. The first column presents the fitness function. The second column shows performance of the proposed hybrid algorithm by using this fitness function in terms of F-measure. Finally, the last column shows accuracy of the genetic algorithm by using this fitness function.

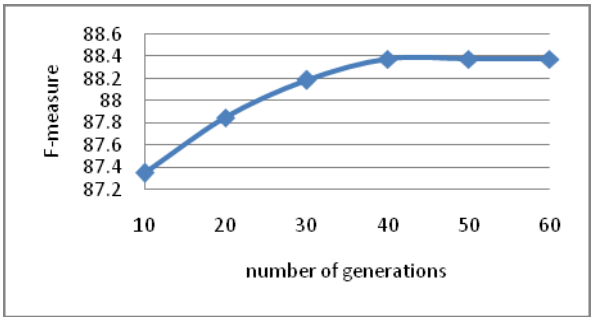

Figure 2: Average fitness values of executions of the GA using different number of generations and population sizes

Table 2: Optimum values of the genetic algorithm parameters

Genetic parameter	Value
Number of generations	40
Population size	50
Mutation probability	0.01
Crossover probability	0.4

Table 3: The effect of different fitness functions on the performance of the proposed algorithm

Fitness function	Proposed algorithm F-measure	Genetic algorithm accuracy
$w1.F1 + w2.F2 + w3.F3 + w4.F4$	88.06	87.60
$w1.F1 + w3.F3 + w4.F4$	86.70	86.01
$w1.F1 + w2.F2 + w4.F4$	84.71	83.21
$w1.F1 + w2.F2 + w3.F3$	86.27	85.01

6.2 Effectiveness of the Proposed Algorithm

The experiment applied 91,933 chunk-annotated tokens as the training set and 21,263 tokens as the test set. Parameter settings shown in Table 2 were used for the genetic algorithm.

Table 4 compares overall accuracy from the combination of the rule-based model and the genetic algorithm. Approximately 82 percent of tokens were tagged by the rule-based model with 96.39 percent accuracy. In fact, from tokens in the test set, 17,472 tokens were tagged by the rule-based model and among them 16,841 tokens were assigned correct

tags. In contrast, the genetic algorithm assigned correct tags to 3,321 tokens from 3,791 tokens and achieved 87.60 percent accuracy.

Table 4: Comparing the accuracy of the rule-based model versus genetic algorithm

	#tagged tokens	#correctly tagged tokens	Accuracy
Rule based model	17472	16841	96.39
Genetic algorithm	3791	3321	87.60

In the next experiment, performance of the proposed genetic algorithm was compared to that of the HMM model (see Table 5). For this reason, the HMM model was used with big rams and Viterbi decoding. At first, the rule-based model was run. After that, the HMM model tagged tokens, which didn't get tagged by the rule-based model. In fact, the aim was to examine the effect of using the genetic algorithm instead of the Viterbi algorithm and results demonstrated that the proposed algorithm outperformed the Viterbi algorithm. Simulation results showed that the heuristic nature of the genetic algorithm was useful for tagging.

Table 5: Comparing the proposed algorithm with HMM model

	Accuracy	Precision	Recall	F_measure
Proposed chunker	94.82	87.70	88.43	88.06
HMM model	93.56	86.97	85.74	86.35

Table 6 compares the approaches applied to these tests with other available chunkers for Persian text. As can be seen, our proposed algorithm out performed the other algorithms. The recall was computed (see Table 5) but was not compared to others because they reported only precision.

Table 6: Comparing the performance of the proposed algorithm with other methods

Method	Precision
Shamsfard and SadrMousavi, 2008 [21]	70%
Kiani et al., 2009 [13]	85.7%
Our proposed method	87.70%

In the above experiments correct POS tags were used from the Peykareh corpus. The reason for this was that results from the proposed algorithm were compared to those from other available chunkers and comparison showed that the POS tags were correct. In order to examine the effect of using estimated tags, the test data was tagged by using Tnt tagger with 96.73 percent accuracy. In this experiment, the proposed algorithm achieved the F-measure of 87.46 percent.

Since about 82 percent of tokens got IOB tags in the first stage and the training corpus was analyzed only once (at the first iteration), the genetic algorithm found the solution in reasonable time.

7 Conclusion and Future Work

This paper proposes an approach for text chunking Persian that combines genetic algorithms with rule-based models. Genetic algorithms provide a search strategy to learn general chunk patterns in text optimizing a measure of probability that is effective globally. However, the rule-based model handles special cases and exceptions to general patterns. Results of the tests reported in this study show that the proposed algorithm outperformed other algorithms for Persian text chunking and classical HMM based methods.

Although this paper presents an algorithm for Persian text chunking, the principles can be applied to other languages. A genetic algorithm can be used for any language to find common statistical patterns for chunking. Obviously, there may be exceptions to these patterns. So some rules were defined to handle exceptions in the rule-based model that served to improve performance of the genetic algorithm. In fact, hybridizing a genetic algorithm with the rule-based model improves performance of the chunking process.

In future work, linguistic rules may be extended by analyzing errors of test data. It is also observed that input of the IOB-tag set has a major influence on accuracy. Errors in the training data have caused some problems, and these can be reduced by correcting the training data.

In addition, it is intended that new attributes be added to the fitness function of the genetic algorithm. One advantage of the genetic algorithm compared to other classical approaches such as HMM based methods is that new attributes can be added to the system and this facilitates examination of the effect of different attributes on tagging without altering the system's basic structure. Thus, tests will be done on new attributes applied to the fitness function of the genetic algorithm and to evaluate effects on chunking accuracy.

References

[1] S. Abney, "Parsing by Chunks," *Principle-Based Parsing*, R. Berwick, S. Abney and C. Tenny, Ed., Kluwer Academic, 1991.

[2] S. Abney, "Partial Parsing via Finite-State Cascades," *Natural Language Engineering*, 2(4):337-344, 1996.

[3] E. Alba, G. Luque, and L. Araujo, "Natural Language Tagging with Genetic Algorithms," *Information Processing Letters*, 100:173-182, 2006.

[4] J. Atkinson and J. Matamala, "Evolutionary Shallow Parsing," *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, pp 420-425, 2009.

[5] J. Atkinson and J. Matamala, "Evolutionary Shallow Natural Language Parsing," *Computational Intelligence*, 28(2):156-175, 2012.

[6] S. Baskaran, "Hindi POS Tagging and Chunking," *Proceedings of the NLPAI Machine Learning 2006 Competition*, 2006.

[7] R. A. Bhat and D. M. Sharma, "A Hybrid Approach to Kashmiri Shallow Parsing," The 5th Language and

Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, 2011.
[8] M. BijanKhan, "The Role of the Corpus in Writing a Grammar: an Introduction to a Software," *Iranian Journal of Linguistics*, 19(2), 2004.
[9] A. V. Bosch and S. Buchholz, "Shallow Parsing on the Basis of Words Only: A Case Study," *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pp 433-440, 2002.
[10] W. Daelemans, S. Buchholz, and J. Veestra, "Memory Based Shallow Parsing," *Journal of Machine Learning Research Archive*, 2:559-594, 2002.
[11] K. A. DeJong and W. M. Spears, "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," *Proceedings of the First Workshop Parallel Problem Solving from Nature*, Springer-Verlag, Berlin, pp 38-47, 1990.
[12] G. D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE*, 61(3):268-278. 1973.
[13] S. Kiani, T. Akhavan, and M. Shamsfard, "Developing A Persian Chunker using a Hybrid Approach," *Proceedings of the International Multiconference on Computer Science anid Information Technology*, pp. 227-233, 2009.
[14] S. Noferesti, "Building IOB Tagged Dataset", Unpublished Results, Technical Report, NLP-Lab., Shahid Beheshti University, 2010.
[15] S. Noferesti and M. Rajayi "A Hybrid Algorithm Based on Ant Colony System and Learning Automata for Solving Steiner Tree Problem," *International Journal of Applied Mathematics and Statistics*, 22(11):79-88, 2011.
[16] S. C. Pammi and K. Prahallad, "POS Tagging and Chunking using Decision Forests," Workshop on Shallow Parsing for South Asian Languages, 2007.
[17] F. Pla, A. Molina, and N. Prieto, "An Integrated Statistical Model for Tagging and Chunking Unrestricted Text," *Proceedings of the Third International Workshop on Text, Speech and Dialog*, pp. 15-20, 2000.
[18] Q. Qi, and L. Liu, "Chinese Chunking Based on Frequency used Words," *Computer Engineering and Technology (ICCET)*, 2:432-435 (2010).
[19] L. A. Ramshaw and M. P. Marcus, "Text Chunking using Transformation-Based Learning," *Proceedings of the Third Workshop on Very Large Corpora*, pp 82-94, 1995.
[20] J. I. Serrano and L. Araojo, "Evoluntionary Algorithm for Noun Phrase Detection in Natural Language Processing," *Proceedings of the 2005 IEEE Congress on Evolutionary Computing*, pp. 640-647, 2005.
[21] M. Shamsfard. and M. SadrMousavi, "Thematic Role Extraction using Shallow Parsing," *International Journal of Computational Intelligence*, 4(2):126-132, 2008.
[22] K. S. Tjong and S. Buchholz, "Introduction to the CoNLL-2000 Shared Task: Chunking," *Proceedings of Conference on Natural Language Learning*, pp 127-132, 2000.
[23] Tsuchiya, Shime and Takagi, "Chunking Japanese Compound Functional Expressions by Machine Learning," 11th Conference of the European Chapter of the Association for Computational Linguistics, *Proceedings of the Workshop on Multi-Word-Expressions in a Multilingual Context*, pp. 25-32, 2006.
[24] A. Wajid and H. Samad, "A Hybrid Approach to Urdu Verb Phrase Chunking," *Processing of 8th Workshop on Asian Language Resources* (*ALR-8*), pp 137-143, 2008.
[25] Y. Ch. Wu, Ch. H. Chang, and Y. Sh Lee, "A General and Multi-Lingual Phrase Chunking Model Based on Masking Method," *Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Text Processing*, Mexico City, Mexico, LNCS 3878:144-155, 2006.

Samira Noferesti is a Ph.D student at ShahidBeheshti University. Her interests include artificial intelligence, natural language processing and opinion mining.

Mehrnoush Shamsfard received her BS and MS both in Computer Software Engineering from Sharif University of Technology, Tehran, Iran. She received her PhD in Computer Engineering-Artificial Intelligence from AmirKabir University of Technology in 2003.

Dr. Shamsfard is an Assistant Professor at Shahid Beheshti University from 2004. She is the head of NLP Research Laboratory of Electrical and Computer Engineering Faculty. Her main fields of interest are natural language processing, ontology engineering, text mining and semantic web.