A Hybrid Algorithm for Recognizing the Position of Ezafe Constructions in Persian Texts

Samira Noferesti, Mehrnoush Shamsfard

Faculty of Electrical and Computer Engineering,

Shahid Beheshti University, Iran.

***Abstract* — In the Persian language, an Ezafe construction is a linking element which joins the head of a phrase to its modifiers. The Ezafe in its simplest form is pronounced as –e, but generally not indicated in writing. Determining the position of an Ezafe is advantageous for disambiguating the boundary of the syntactic phrases which is a fundamental task in most natural language processing applications. This paper introduces a framework for combining genetic algorithms with rule-based models that brings the advantages of both approaches and overcomes their problems. This framework was used for recognizing the position of Ezafe constructions in Persian written texts. At the first stage, the rule-based model was applied to tag some tokens of an input sentence. Then, in the second stage, the search capabilities of the genetic algorithm were used to assign the Ezafe tag to untagged tokens using the previously captured training information. The proposed framework was evaluated on Peykareh corpus and it achieved 95.26 percent accuracy. Test results show that this proposed approach outperformed other approaches for recognizing the position of Ezafe constructions.**

***Keywords* — Ezafe construction, genetic algorithm, genitive construction, rule-based model.**

I. INTRODUCTION

THE "Ezafe"[1] is a Persian language grammatical construct which links two words together. Ezafe means "addition" and is an unstressed vowel –e– which marks genitive cases. The constructs linked by the Ezafe particle are known as "Ezafe constructions". Some common uses of the Persian Ezafe are [1]:

— a noun before an adjective:
e.g. توپ قرمز[2] (tu:p-Ezafe Germez) "red ball"

— a noun before a possessor:
e.g. کتاب علی (ketã:b-Ezafe Ali:) "Ali's book"

— some prepositions before nouns:
e.g. زیر میز (zi:r-Ezafe mi:z) "under the table"

The Ezafe in its simplest form is pronounced as –e, but generally not indicated in writing. In some cases the Ezafe has an explicit sign in writing. For example, with nouns ending in ا (ã:) or و (ou), the Ezafe appears as an ی (j) at the end; with nouns ending in a silent ه (h) (short e followed by a mute h), the Ezafe may appear as a superscript ء (hamze) or a ی (j).

The Ezafe is also found in Urdo [2], Kurdish [3] and Turkish [4]. The Persian Ezafe has been discussed extensively [5]-[8]. This construction raises several issues in syntax and morphology. There are three issues on the function of the Ezafe in the literature: (1) the Ezafe is a case marker [9], (2) the Ezafe is inserted at PF to identify constituenthood [10], and (3) the Ezafe is a phrasal affix [11].

Determining the position of an Ezafe construct may facilitate text processing activities in natural language processing (NLP) applications, such as segmenting a phrase or detecting the head word of a phrase [12]. Moreover, recognizing words which need an Ezafe is advantageous for tokenization [13], morphological analysis, and syntax parsing [14], and it is essential for speech synthesis [15].

Some NLP tasks in Persian, such as machine translation [16], construction of morphological lexicons [14], and grammar construction [17], have benefited from the availability of an Ezafe construction. However, they have determined the position of the Ezafe manually, exploited cases in which the Ezafe is visually represented, or extracted some insertion rules which are not general; therefore, they could not determine the Ezafe tags for all tokens in a text.

The Persian Ezafe has been discussed extensively in theory [5], [18], but there are few works on the automatic detection of this construction in Persian texts. The most completely reported works on this subject are one work based on probabilistic context free grammar (PCFG) [19] and another based on classification and regression tree (CART) [20]. The former uses a bank including trees of noun groups in Persian for training PCFG. Then, a bottom-up parser extracts the most probable noun groups of the input. Finally, using lexical analysis, the system determines which words need an Ezafe. The disadvantage of this method is that writing a PCFG requires a large amount of linguistic knowledge. In addition, it is not sensitive to lexical information. The latter uses morpho-syntactic features of words to train and construct binary classification trees to predict the presence or absence of an

[1] It is also known as Kasreh.

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

DOI: 10.9781/ijimai.2014.262

Ezafe between two adjacent words. In fact, there are two kinds of rules: rules which predict Ezafe words, and rules which predict non-Ezafe words. Although this method can predict the absence of an Ezafe with high accuracy, it is not sufficient in detecting words which need an Ezafe. In other words, the rules which predict the non-Ezafe words act more precisely.

The main contributions of this paper were (1) introducing a framework for combining genetic algorithms with rule-based models, and (2) using the proposed framework to develop an Ezafe tagger.

Combining genetic algorithms with rule-based models brings the advantages of both approaches and overcomes their problems. Genetic algorithms can detect general patterns in text, but sometimes they cannot handle exceptions and special cases. In such cases, rule-based models can provide significant improvements by defining rules for handling special cases and exceptions. In our proposed framework, for the rule-based model, linguistic rules were extended by analyzing errors of the genetic algorithm and defining new rules for handling these errors (named as correction rules). In contrast, a rule-based model needs a great deal of knowledge external to the corpus that only linguistic experts can generate. In fact, acquiring rules through interviews with experts is cumbersome and time-consuming. Furthermore, certain application domains are very complicated and may require a large number of rules. Therefore, the acquired rules may be incomplete or even partially correct. In order to overcome these problems, we can handle general patterns by genetic algorithms and only define correction rules to handle special cases, which means less time handling by expert humans. We can also define a set of general rules besides correction rules in order to reduce the run time of the genetic algorithm.

There is a remarkable amount of ongoing research on applying machine learning approaches to different tasks of NLP 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. On the other hand, the effectiveness of using hybrid approaches has been demonstrated in different NLP tasks [21]. Thus a hybrid approach for determining the position of Ezafe construction was chosen for this study. Results of the tests in this study show that our proposed algorithm outperformed other algorithms for Persian Ezafe tagging as well as the classical MM based method.

The rest of the paper is organized as follows. Section 2 introduces the annotated corpus of Persian texts. Section 3 explains our proposed model. Experiment results are discussed in section 4. Finally, section 5 concludes the paper.

II. THE CORPUS

An annotated corpus of Persian text is needed in order to train and evaluate the Ezafe tagger. This corpus must be annotated with POS and Ezafe tags. For the current work, a subset of Persian POS tagged corpora known as Peykareh[3] [21] was used. 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 20 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.

The tagged corpus was divided into three sets: (1) a training set including 423,721 tokens, (2) a held-out data set containing 1,010,375 tokens, and (3) a test set containing 39,850 tokens.

A big portion of the Peykareh corpus was set aside as a held-out dataset. The held-out dataset was used to find exceptions to general rules in the rule-based model. Since the exceptions occur only rarely, much more data was needed to determine the exceptions. Furthermore, to determine the classes of conjunctions and prepositions which never take an Ezafe or always require an Ezafe, the held-out data set was searched. Thus, a sufficiently large data set was needed to find as many words as possible.

III. THE PROPOSED ALGORITHM

This paper proposes a hybrid approach to determine the position of Ezafe constructions in Persian texts. The Ezafe tagger contained two phases. The first phase used the rule-based model to tag as many words as possible. Then the second phase ran the genetic algorithm to assign tags to the tokens which had not been assigned an Ezafe tag in the previous phase. Therefore, a faster genetic algorithm was achieved by producing more tagged tokens that had been generated from the rule-based model.

The Ezafe tagger assigned each word of an input sentence with one of two tags: *true* or *false*. Tag *true* for a word meant that it requires an Ezafe, and the tag *false* meant it does not require one.

A. The rule-based model

Initially, some general rules such as “verbs do not take an Ezafe” were defined using linguistic knowledge. Although the genetic algorithm could detect these tags correctly by training on annotated examples, we preferred to define such rules in

[3] This corpus also is known by its author’s name, Bijankhan.

order to reduce the run time of the algorithm. The more tokens detected by the rule-based model there were, the less chromosome length and lower number of generations were needed.

Next, the exceptions of each rule were explored on the held-out data, and some new rules were defined to handle exceptions. This process was repeated. In other words, if the generated rules had exceptions, new rules were defined. In some cases we could not find suitable rules to fix errors. In these cases, probabilistic rules were used.

The genetic algorithm tagged tokens according to the context; however, experiments showed that words which appeared in an infrequent context usually took an incorrect tag from the genetic algorithm. We tried to handle these cases through hand-crafted rules. Thus, 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.

In this way, a set of 53 hand-crafted rules was developed. Then, the most suitable sequence of rules was determined in terms of avoiding bleeding and creeping; in 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 various dimensions:

— Deterministic vs. probabilistic rules: Deterministic rules are those which are always valid and correct; probabilistic rules may have exceptions. In other words, probabilistic rules are valid most of the time, but as they may have exceptions we apply them with a probability of less than 100%. This probability is extracted from the corpus.

— Negative vs. positive rules: Negative rules find and tag negative examples which are the structures which never take an Ezafe, while the positive rules determine structures which need an Ezafe.

— Syntactic, morphological and lexical rules: Syntactic rules use part-of-speech to tag words (either as the target word or a neighboring word) in a sequence to determine the Ezafe tag, while morphological rules consider internal and morphological structures of a word to do this task, and lexical rules consider real words.

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

1) Syntactic Rules

Some POS categories enforced a special tag on words or on neighboring words. The accusative case marker را (rã:) and verbs were among this set.

— Verbs

- In the Persian language the Ezafe is not used with verbs. The following rule dictated that verbs, which were shown in the corpus with the POS tag V, never take an Ezafe.
  If POS(X) =V Then EZ-Tag(X) =false
- If a verb appears as a stand-alone, the word before it does not take an Ezafe. We presented this by the following rule:
  If POS(X) =V Then EZ-Tag(X-1) =false
- If the verb is not a stand-alone and appears as an attachment (enclitic) to another word, then the previous word (before the combination) may take an Ezafe. This is the case for some of the enclitics representing the copula verb 'to be' such as ی (i:) "to be- single second person" and ًم (æm) "to be- single first person". These enclitics are ambiguous and, in addition to copula verbs, can be interpreted as an indefinite marker or as a single first-person possessive pronoun, respectively. For example, the word شاعری (ʃã:?eri:) may mean شاعر هستی (ʃã:?er hæsti:) "you are poet" or یک شاعر (jek ʃã:?er) "a poet". In the first case, even though the whole word was tagged as a verb in the corpus, it is actually a combination of a noun and a verb. Even though its verb part and its previous word do not take an Ezafe, the word before the noun part of it may take one. As another example, in the following sentence the word دولتم (dolætæm) "I'm government" is an abbreviation of دولت هستم (dolæt hæstæm) "I am government". In Peykareh corpus, this word was tagged as verb with POS tag V,AJCC. However, the previous word takes an Ezafe.
  من در استخدام دولتم.
  I am a government employee.
  Thus, we used POS tag V, ACJJ for this kind of verbs to prevent applying the previous rule for them.

— The accusative case marker را (rã:)

- The Persian language has an accusative case marker را (rã:) that follows the direct object, adverb or prepositional object. The following rule dictated that the accusative case marker, which was shown in the corpus with the POS tag POSTP, never takes an Ezafe.
  If POS(X) =POSTP Then EZ-Tag(X)=false
- The word before را (rã:) does not take an Ezafe too.
  If POS(X) =POSTP Then EZ-Tag(X-1)=false
- In some cases the accusative case marker is attached to the previous noun or pronoun. For example the word مرا[4] (mærã:) is an abbreviation of من را (mæn rã:), in which را (rã:) is an object marker and من (mæn) "me" is a pronoun. This rule was written as follows:
  If postfix(X) = accusative-case-marker EZ_Tag(X) =false

2) Morphological rules

The following rules are examples of morphological rules that determine structures that take an Ezafe.

—When a word ending in the plural suffix ها (hã:) needs the

[4] Sometimes it means 'my' and other times it means 'me'

Ezafe, the letter ی (j) must be attached to the end of the word in writing. Thus, if a plural word ends in ها (hã:), this word should not be followed by an Ezafe unless it is followed by a ی (j) clitic.

If postfix(X)= ها Then EZ-Tag(X)=false

Consider the following example.

نامه های علی را خواندم.

I read Ali's letters.

The word نامه ها (nã:me hã:) "letters" is the plural form of نامه (nã:me) "letter". When this word requires Ezafe, we add ی (j) at the end of it.

—If the last character of a word is اً (Tanvin)[5], then it does not take an Ezafe.

If LastChar(X) = اً Then EZ-Tag(X)=false

3) Lexical rules

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

—Prepositions

Reference [23] showed that the class of prepositions in the Persian language is not uniform with respect to the Ezafe. Some prepositions reject the Ezafe (These prepositions were called Class P1.), while others either permit or require it. We divided the latter group into two classes. The first class which always requires an Ezafe was called Class P2. The other class which permits an Ezafe but does not necessarily require one was called Class P3. Table I shows some examples of each class. We applied the following rules to handle prepositions:

If POS(X) =P and WORD(X)∈ClassP1 Then EZ-Tag(X)=false

If POS(X) =P and WORD(X)∈ClassP2 Then EZ-Tag(X)=true

TABLE I
EXAMPLES OF PREPOSITION CLASSES

Class name	Examples
Class P1	به (be) "to" از (æz) "from" با (bã:) "with" در (dær) " in, on"
Class P2	وسط (væsæt) "in the middle" دور (du:r) "around" بیرون (bi:ru:n) "outside" داخل (dã:xel) "inside"
Class P3	زیر (zi:r) "under" رو (ru:) "on" بالا (bã:lã:) "up" جلو (dʒoulou) "in front of"

— Conjunctions

Same as prepositions, we divided conjunctions into two classes. Some conjunctions never take an Ezafe (These conjunctions were called Class C1), while others always take an Ezafe (These conjunctions were called Class C2). In order to determine these classes we searched 300 files of Peykareh corpus which were selected as held-out data. Some examples of Class C1 and Class C2 are presented in Table II. The following rules applied to conjunctions:

If POS(X) =CONJ and WORD(X)∈ClassC1 Then EZ-Tag(X)=false

If POS(X) =CONJ and WORD(X)∈ClassC2 Then EZ-Tag(X)=true

TABLE II
EXAMPLES OF CONJUNCTION CLASSES

Class name	Examples
Class C1	و (væ) "and" زیرا (zi:rã:) "because" یا (jã:) "or" که (ke) "that" یعنی (jæni:) "means"
Class C2	علی رغم (?ælã:ræy me) "in spite of" باستثناء (beestesnã:?e) "except" سوای (sævã:je) "except" برخلاف (bærxælã:fe) "in spite of"

—Adverbs

We also divided Persian adverbs into three classes. Class A1 contained adverbs which never take an Ezafe; class A2 included adverbs with an obligatory Ezafe; class A3 contained adverbs with an optional Ezafe. Examples of these classes are shown in Table III. The following rules applied to adverbs:

If POS(X) =ADV and WORD(X)∈ClassA1 Then EZ-Tag(X)=false

If POS(X) =ADV and WORD(X) ∈ClassA2 Then EZ-Tag(X)=true

TABLE III
EXAMPLES OF ADVERB CLASSES

Class name	Examples
Class A1	بویژه (bevi:ʒe) "specially" هیچگاه (hi:tʃgã:h) "never" شاید (ʃã:jæd) "maybe"
Class A2	مثل (mesle) "like" مانند (mã:nænde) "like" از قبیل (æz Gæbi:le) "such as"
Class A3	گذشته (gozæʃte) "past" سالیانه (sã:li:jã:neh) "annual"

4) Probabilistic rules

We also defined 5 probabilistic rules which were correct and valid in most cases but had some exceptions in a few cases. Defining each rule, the probability of that rule was calculated according to the corpus. The lowest probability among these rules was 0.95. Here, we discuss some of the probabilistic rules.

—Long vowels

There are three long vowels in Persian: ا (ã:), ی (i:) and و (u:). Generally, when a word ending in ا (ã:) or و (u:) needs an Ezafe, the letter ی (j) is added to the end of it. However, this rule has some exceptions.

In the case of ا (ã:), these exceptions happen when we replace أ (Alef Hamze) by ا (ã:) (single alef). Alef Hamze is a single Arabic character that represents the two-character

[5] This sign was taken from Arabic alphabet

combination of Alef plus Hamze and in Persian writing is sometimes replaced by the letter ا (ã:). Consider the following example:

آقای احمدی منشا فساد را فقر می داند.

"Mr. Ahamdi believes that the source of evil is poverty."

The word آقا (ã:Gã:) "Mr." takes a ی (j) at the end because it requires the Ezafe; however, the word منشا (mænʃæ) "source" also ends in ا (ã:) and needs the Ezafe, but it does not get ی (j). In fact, the last character of this word is أ (Alef Hamze) which is written the same as ا (ã:).

To compute the probability of the rule, the algorithm searched the held-out data set and computed the percentage of words ending with ا (ã:) and the Ezafe which had the letter ی (j) added. In other words, this probability was computed by the following formula:

$$p = \frac{\text{count(words ending with ی ا (ã:je) and } True \text{ Tag)}}{\text{count(words end ing with ی ا (ãje))}} \quad (1)$$

Thus, the following rule was defined with a 95% probability:

If LastChars(X) =ی ا Then with a 0.95 probability EZ-Tag(X) =true

In the same way, the following rules were defined:

If LastChars(X) = ا Then with a 0.9978 probability EZ-Tag(X) =false

If LastChars(X) =و ی Then with a 0.96 probability EZ-Tag(X) =true

—Tanvin

Tanvin is a sign which is derived from Arabic. The following rule says that with a probability of 96.24% the word preceding a word that has ً (Tanvin) as the final character does not take an Ezafe:

If LastChars(X) = ً Then with a 0.9624 probability EZ-Tag(X-1) =false

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

TABLE IV
NUMBER OF EXTRACTED RULES IN EACH CATEGORY

Syntactic	Morphological	Lexical	Probabilistic
25	4	19	5

After running the rule-based model, some of the tokens remained untagged. Thus, a genetic based algorithm was used to tag the remaining words.

B. Genetic tagging algorithm

The proposed genetic algorithm receives a natural language sentence and assigns a corresponding tag 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 Ezafe tag sequence.

In our implementation, each gene can take values: *true* or *false*. Individuals of the first generation were produced randomly. After producing an individual, all tokens of a given sentence were assigned Ezafe tags (some of tokens get Ezafe tag by the rule-based model and others get Ezafe tag by the genetic algorithm).

An initial population was created randomly by assigning a random value to each untagged gene (some genes were assigned Ezafe 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 *true*, it was reset to *false* and vice versa.

1) Fitness Functions

To evaluate the quality of Ezafe tags 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 and next word.

F1 considered the sequence of POS tags of a sentence. The probability of the sequence of POS tags of a sequence of *n* words was as follows:

$$F1 = \sum_{i=1}^{n} P(EZ - POS_i | POS_{i+1}) \quad (2)$$

Where, $P(EZ - POS_i | POS_{i+1})$ represents the probability that the current word with POS_i tag gets the Ezafe when the next word has the POS_{i+1} tag. The probability of assigning the Ezafe to a word given the next POS tag was computed as:

$$P(EZ - POS_i | POS_{i+1}) = \frac{count(EZ - POS_i, POS_{i+1})}{count(POS_i, POS_{i+1})} \quad (3)$$

Where, count(POS_i, POS_{i+1}) was the number of occurrences of the (POS_i, POS_{i+1}) sequence within the training corpus, and $count(EZ - POS_i, POS_{i+1})$ was the number of (POS_i, POS_{i+1}) occurrences when the first token has an Ezafe within the same corpus. In order to compute F1 function, the HMM model can be used with the Viterbi algorithm [24].

For computing F2 function, a data driven approach was applied to calculate the probability that a specific word has the Ezafe.

$$F2 = \sum_{i=1}^{n} P(EZ_i | word_i) \quad (4)$$

F2 was defined because some words in Persian are mostly assigned a special tag. For example, the word تقریباً (tæGri:bæn) "approximately" never take an Ezafe.

F3 function was the probability that a token gets an Ezafe when it occurs before a specific word in the training corpus.

$$F3 = \sum_{i=1}^{n} P(EZ_i | word_i , word_{i+1}) \quad (5)$$

This function was defined to handle compound words such as اختلاف نظر (extelã:f-Ezafe næzær) "difference in opinion".

In Persian, some words such as سایر (sã:jer) "other" get the Ezafe most of the time. Therefore, we defined the F4 function to consider these words. The F4 function was the probability that a specific word occurs after a word with the Ezafe in the training corpus.

$$F4 = \sum_{i=1}^{n} P(word_{i+1} | EZ_i) \quad (6)$$

The following fitness function was used to evaluate the genetic algorithm:

$$fitness - function = \sum_{i=1}^{4} w_i . F_i \quad (7)$$

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 [25]. Finally, the values of w_1, w_2, w_3 and w_4 were set to 0.8, 0.5, 0.1 and 0.1 respectively.

IV. Experiments

The proposed algorithm was implemented using java language and was run on a Pentium IV processor. First, the rule-based model was run followed by the genetic algorithm, and the best solution was selected. Approximately 78% of the tokens were tagged with the rule-based model, because about 80% of the tokens selected as test data did not require an Ezafe, and most of them were tagged by the rule-based system.

To evaluate the performance of our proposed algorithm, three measures were taken: accuracy (the percentage of correctly tagged tokens), precision (the percentage of predicted tags that were correct) and recall (the percentage of predictable tags that were found).

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

$$F - measure = \frac{2.precision.recall}{precision + recall} \quad (8)$$

A. 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, a subset of 34,832 tokens from held-out data set was selected. Then, the proposed algorithm was run on this set.

Beginning with a baseline configuration, such as Dejong's setting [26] with 1000 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. Fig. 1 shows that the best results were obtained using the mutation probability 0.05.

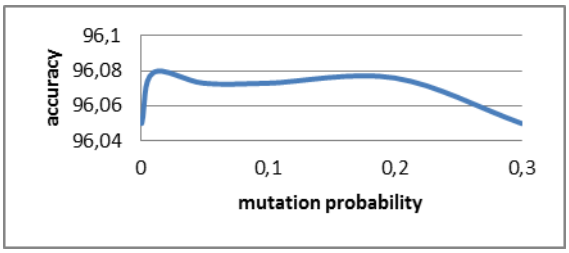

Fig. 1. Average fitness values of executing of the GA using different mutation probabilities

In the same way, crossover probability was set to 0.6. In Fig. 2 the results of running the genetic algorithm using mutation probability 0.05, crossover probability 0.6, population size 50 and different number of generations are shown.

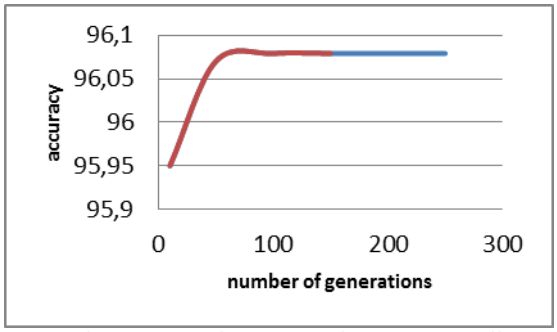

Fig. 2. Average fitness values of executions of the GA using different number of generations

Table V shows the optimum values of genetic algorithm parameters.

TABLE V
SELECTED VALUES FOR GENETIC ALGORITHM PARAMETERS

P_m	P_c	population size	generations
0.05	0.6	50	150

B. Effectiveness of the Proposed Algorithm

The experiment applied 423,721 annotated tokens as the training set and 39,850 tokens as the test set. Parameter settings shown in Table V were used for the genetic algorithm.

Table VI compares our approach with a baseline method and other available methods based on PCFG [19] and CART [20]. We also implemented the binary Markov model with Viterbi decoding (a typical algorithm widely used for stochastic tagging). As can be seen, our proposed algorithm

outperformed these algorithms in terms of F-measure. The baseline assumed all words have an Ezafe, resulting in 100% recall but very low precision (15.79%). We could define another baseline where no word has the Ezafe tag. In this case, we would achieve 84.21% precision, but the recall would be 0.

TABLE VI
EVALUATING OUR PROPOSED ALGORITHM IN TERMS OF F-MEASURE

	recall	precision	F-measure
Our proposed algorithm	88.81	87.85	88.33
Baseline	100	15.79	27.27
PCFG method [19]	86.74	87.54	87.14
CART method [20]	88.85	84.13	86.43
Viterbi method	95.51	78.63	86.25

Since we had no access to the corpus that was used for training in the CART method [20] or a description of the exact features used, we could not regenerate the exact results. For this reason, we used two approaches for comparison. In the first approach, we compared results of our proposed method with the best results reported in [20], and in the second one, we implemented the CART method using the same features as our proposed method and tested it on our test corpus. Since the performance of the second approach was much lower than what was reported in [20], we only presented the results of the first approach in Table VI.

In the above-mentioned experiments, correct POS tags were used, because results from the proposed algorithm were compared to those from other available Ezafe taggers. Since these taggers had used correct tags, we also used the correct tags to enable the comparison. By using a Tnt tagger, the proposed algorithm achieved a 95.08% accuracy, while with correct tags it achieved an accuracy of 95.49%. This indicates that the tagging error decreased the Ezafe detection accuracy by only about 0.41%. The reason for this is that both the rule-based model and the genetic algorithm consider other features besides POS tags, and these features can, to some extent, cover the errors of the POS tagger.

Considering accuracy as the percentage of correctly assigned tags, we evaluated the performance of the proposed algorithm from two different aspects: (1) the overall accuracy by taking all tokens in the test corpus into account, and (2) the accuracy for words with an Ezafe and without an Ezafe, respectively. Table VII shows that the overall accuracy of the proposed algorithm was around 95.26%. Additionally, the accuracy for detecting words without an Ezafe was significantly higher than that for words with an Ezafe (96.89% versus 88.81%).

TABLE VII
EVALUATING OUR PROPOSED ALGORITHM IN TERMS OF ACCURACY

	Number of correctly tagged tokens		
	with Ezafe	without Ezafe	Total
Corpus	8054	31796	39850
Our proposed Algorithm	7153	30807	37960
Accuracy	88.81	96.89	95.26

Table VIII compares overall accuracy from the combination of the rule-based model and the genetic algorithm. Approximately 78% of tokens were tagged by the rule-based model with 99.21% accuracy. In fact, from tokens in the test set, 30,972 tokens were tagged by the rule-based model and among them 30,728 tokens were assigned correct tags. In contrast, the genetic algorithm assigned correct tags to 7,232 tokens from 8,878 tokens and achieved 81.46% accuracy.

TABLE VIII
COMPARING THE ACCURACY OF THE RULE-BASED MODEL VERSUS GENETIC ALGORITHM

	Number of tagged tokens	Number of correctly tagged tokens	Accuracy
Rule-based model	30972	30728	99.21
Genetic algorithm	8878	7232	81.46

Table IX compares the accuracy of the rule-based model versus the genetic algorithm. In RBM1, we ran the rule-based model and assigned the false tag to tokens which did not get the Ezafe tag after applying the rules. In contrast, the untagged tokens got true tags in RBM2. We also ran the genetic algorithm alone (without the rule-based model). Results show that the combination of the rule-based model and the genetic algorithm outperformed both individual algorithms. As might be expected, the main problem of the RBM models was missing rules, which caused some tokens remained untagged, and the main problem of the genetic algorithm was special cases that could not be handled by general patterns.

TABLE IX
COMPARING THE ACCURACY OF THE RULE-BASED MODEL VERSUS GENETIC ALGORITHM

	Accuracy
RBM1	85.29
RBM2	91.21
GA	89.21
Combination of rule-based and GA	95.26

Since the ratio of words with an Ezafe to words without an Ezafe was low, the Kappa coefficient was used to evaluate the proposed algorithm. This measure was first suggested for linguistic classification tasks [27] and has since been used to avoid dependency of the score on the proportion of non-breaks in the text. The Kappa coefficient (K) was calculated as:

$$K = \frac{Pr(A) - Pr(E)}{1 - Pr(E)} \quad (9)$$

Where, Pr(A) was accuracy, and Pr(E) was the ratio of words without an Ezafe to total words. Table X shows how to evaluate an algorithm in terms of Kappa value. Using (9) the Kappa coefficient became 0.77. According to Table X, our proposed algorithm is assessed as good.

TABLE X
DECISION MAKING BY USING KAPPA [19]

Kappa values	Strength of agreement
K<0.2	bad
0.2<K≤0.4	average
0.4<K≤0.6	relatively good

0.6<K≤0.8	good
0.8<K≤1	very good

Table XI shows that our proposed algorithm outperformed previously reported algorithms in terms of Kappa value.

TABLE XI
COMPARING THE PERFORMANCE OF THE PROPOSED ALGORITHM WITH OTHER METHODS

Kappa value		
Our proposed algorithm	PCFG method	CART method
0.77	0.74	0.72

In the final experiment, we assessed the impact of training corpus size on the performance of the proposed algorithm. The corpus size was reduced slightly until it reached 32% of the initial training corpus size. The results are presented in Fig. 3. As can be seen, the proposed algorithm's accuracy did not show a significant drop when reducing the training corpus size from 100% to 60%.

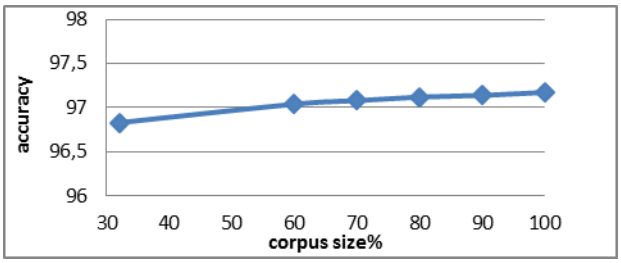

Fig. 3. The impact of training corpus size on performance

V. CONCLUSION AND FUTURE WORK

This paper proposes a framework for recognizing the position of Ezafe constructions in Persian written texts that combines genetic algorithms with rule-based models. Genetic algorithms provide a search strategy to learn general Ezafe 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 Ezafe tagging and classical HMM based methods.

Although this paper presents an algorithm for Persian Ezafe tagging, the principles can be applied to other NLP tasks such as POS tagging or chunking in any language. A genetic algorithm can be used for any language to find common statistical patterns for tagging. Obviously, there may be exceptions to these patterns, so some rules are defined to handle exceptions in the rule-based model that serve to improve performance of the genetic algorithm. In fact, combining a genetic algorithm with the rule-based model improves performance of the tagging process.

In addition, we showed that the accuracy of the proposed algorithm does not depend highly on the training corpus size. This feature is advantageous for practical applications, because annotating training corpora for text analysis purposes is an extremely demanding task.

In future work, linguistic rules may be extended by analyzing errors of test data. It is also observed that input of the Ezafe-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 tagging accuracy.

It was also observed that high accuracy is extremely influenced by input tag set. A richer tag set with POS information produces more accurate results. For example, we can consider additional information with a POS noun, such as time, location, and so on. In addition, in [5] there is a class of lexical words called eventive adjectives, and they cannot co-occur with an Ezafe in contrast with other lexical words. Consider the following examples. Predicative adjectives may only appear in Light Verb Constructions (a) and not in Ezafe Constructions (b).

(a) علی کتاب را فراموش کرد "Ali forgot the book."

(b) * فراموش کتاب توسط علی "forgetting the book by Ali"

We are going to enrich the tagset with more POS tags such as eventive adjectives to define more accurate rules.

REFERENCES

[1] A. Kahnemuyipour, "Persian Ezafe construction: case, agreement or something else," in *Proceedings of the 2nd Workshop on Persian Language and Computer, Tehran University, Tehran, Iran*, 2006.

[2] T. Bögel, M. Butt, and S. Sulger, "Urdu ezafe and the morphology-syntax interface," *Proceedings of LFG08*, 2008.

[3] A. Holmberg and D. Odden, "The Izafe and NP structure in Hawrami," Durham Working Papers in Linguistics, 2004.

[4] G. Van Schaaik, *The noun in Turkish: Its argument structure and the compounding straitjacket*: Otto Harrassowitz Verlag, 2002.

[5] G. Karimi-Doostan, "Lexical categories in Persian," *Lingua*, vol. 121, pp. 207-220, 2011.

[6] A. K. Ahranjani, "The head noun in possessive construction in English and Persian languages," *International Journal of Academic Research*, vol. 2, 2010.

[7] A. K. Ahranjani and R. Tohidian, "Ezafe construction in complex noun phrases in Persian medieval poems," *International Journal of Academic Research*, vol. 3, 2011.

[8] A. Moinzadeh, "The Ezafe phrase in Persian: How complements are added to Ns and As," *Journal of Social Sciences & Humanities of Shiraz University*, vol. 23, pp. 45-57, 2006.

[9] R. Larson and H. Yamakido, "Ezafe and the deep position of nominal modifiers," in *Barcelona Workshop on Adjectives and Adverbs, Barcelona*, 2005.

[10] J. Ghomeshi, "Non-projecting nouns and the ezafe: construction in Persian," *Natural Language & Linguistic Theory*, vol. 15, pp. 729-788, 1997.

[11] P. Samvelian, *A (phrasal) affix analysis of the Persian Ezafe*: Cambridge Univ Press, 2007.
[12] P. Samvelian, "The Ezafe as a head-marking inflectional affix: Evidence from Persian and Kurmanji Kurdish," *Aspects of Iranian Linguistics: Papers in Honor of Mohammad Reza Bateni*, pp. 339-361, 2007.
[13] M. Ghayoomi and S. Momtazi, "Challenges in developing Persian corpora from online resources," in *Asian Language Processing, 2009. IALP'09. International Conference on*, 2009, pp. 108-113.
[14] B. Sagot and G. Walther, "A morphological lexicon for the Persian language," in *Proceedings of the 7th Language Resources and Evaluation Conference (LREC'10)*, 2010.
[15] M. Sheikhan, M. Tebyani, and M. Lotfizad, "Continuous speech recognition and syntactic processing in Iranian Farsi language," *International Journal of Speech Technology*, vol. 1, pp. 135-141, 1997.
[16] J. W. Amtrup, H. M. Rad, K. Megerdoomian, and R. Zajac, *Persian-English machine translation: An overview of the Shiraz project*: Citeseer, 2000.
[17] S. Muller and M. Ghayoomi, "PerGram: A TRALE implementation of an HPSG fragment of Persian," in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, 2010, pp. 461-467.
[18] M. Bijankhan, J. Sheykhzadegan, M. Bahrani, and M. Ghayoomi, "Lessons from building a Persian written corpus: Peykare," *Language resources and evaluation*, vol. 45, pp. 143-164, 2011.
[19] S. Isapour, M. Homayounpour, and M. Bijabkhan, "The Prediction of Ezafe Construction in Persian by Using Probabilistic Context Free grammar," in *In Proceedings of 13th Annual Conference of Computer Society of Iran*, Kish Island, 2008.
[20] A. Koochari, B. Qasemzade, M. Kasaeiyan, and M. Namnabat, "Ezafe Prediction in Phrases of Farsi Using CART," in *Proceedings of the I International Conference on Multidisciplinary Information Sciences and Technologies*, 2006, pp. 329-332.
[21] R. Dehkharghani and M. Shamsfard, "Mapping Persian Words to WordNet Synsets," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 1, 2009.
[22] M. Bijankhan, "The role of the corpus in writing a grammar: An introduction to a software," *Iranian Journal of Linguistics*, vol. 19, 2004.
[23] V. Samiian, "The Ezafe construction: some implications for the theory of X-bar syntax," *Persian Studies in North America*, pp. 17-41, 1994.
[24] G. D. Forney Jr, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268-278, 1973.
[25] S. Noferesti and M. Rajaei, "A Hybrid Algorithm Based on Ant Colony System and Learning Automata for Solving Steiner Tree Problem," *International Journal of Applied Mathematics and Statistics*, vol. 22, pp. 79-88, 2011.
[26] K. A. De Jong and W. M. Spears, "An analysis of the interacting roles of population size and crossover in genetic algorithms," in *Parallel problem solving from nature*, ed: Springer, 1991, pp. 38-47.
[27] J. Carletta, "Assessing agreement on classification tasks: the kappa statistic," *Computational linguistics*, vol. 22, pp. 249-254, 1996.

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

Dr. Mehrnoush Shamsfard has received her BS and MS both on 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 has been 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.