I.J. Information Technology and Computer Science, **2014, 09, 39-45**
Published Online August 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.09.05

A Memetic-Based Approach for Web-Based Question Answering

Iman Khodadi
Faculty of Electrical and Computer Engineering of Tarbiat Modares University, Tehran, Iran
E-mail: iman.khodadi@modares.ac.ir

Mohammad Saniee Abadeh
Faculty of Electrical and Computer Engineering of Tarbiat Modares University, Tehran, Iran
E-mail: saniee@modares.ac.ir

Abstract— In this paper we proposed an evolutionary approach for answering open-domain factoid questions, which include searching among sentences that are candidate for the final answer with Memetic Algorithm (MA), and using lexical and syntactic features for calculating fitness of the sentences. Our main purpose is making a search engine with accurate answering ability, or a web-based Question Answering (QA) system. The Text Retrieval Conference (TREC) QA Tracks data are used to develop and evaluate the approach. The answering process begins with retrieving related documents from a search engine. Then, MA searches among all the sentences of these documents and finds the best one. Finally, one or more words will be extracted based on our hand-made patterns. The results of different approaches for local search, mutation, and crossover, and also different values for number of reproduction and retrieved documents are investigated in the empirical study section. The results are promising with sufficient retrieved documents, and we have obtained a threshold value for this variable. Using MA instead of examining all the sentences is a trade-off between lowering the process time and sacrificing the accuracy, but the results show that the Mametic-based approach is more efficient.

Index Terms— Question Answering, Memetic Algorithm, Information Extraction (IE), Natural Language Processing (NLP), Local Search, Evolutionary Computing, Dynamic Mutation Ratio

I. Introduction

Question Answering systems are advanced form of search engines and can provide accurate answer to a query, instead of a list of links to potentially relevant web pages. So, QA systems have additional step that extract exact answer from one of the retrieved sources. One of the main advantages of these systems is providing an easy interaction with huge set of text sources. These systems can resolve the information need indicated in a query, retrieve the related information, and extract an answer from them in a form with respect to the question [1]. An important usage of these systems is being an expert interface in systems such as automatic servicing to customers, for example Anna in Ikea web site, and answering user’s questions in instruments, like Siri in Apple.

Early QA systems were designed in order to enable users to ask questions from structured data, like personnel data [2]. These structured-based systems can answer questions from a specific subject and that is why they are called Restricted-domain QA (RDQA) systems. But there are systems that can answer questions, independent of the domain and they are called Open-domain QA (ODQA) systems. The first ODQA system was MURAX [3] and it used Information Retrieval (IR) with NLP to answer questions. The ODQA approach differs significantly from RDQA, where a natural language query is transformed into a Structured Query Language (SQL). Instead, in ODQA, the answer must be extracted rather than executed [2].

Although off-line QA systems existed before the search engines, but the first web-based QA was developed many years after the appearance of the search engines in 2004, called START [4].

The direction of research in ODQA systems has been mainly handled by TREC. The Text Retrieval Conference arranged a competition for ODQA systems in 1999, called QA Track. The early competitions focused on factoid question, that is, questions requiring a simple fact. But two other question types were added later, named list questions, that is, questions requiring a list of items, and “other” questions, that is, questions requiring a fact about a subject that is not mentioned in the factoid and list questions. In addition to these three types, there are other question types based on TREC classification, including: definition, hypothetical, causal, relationship, procedural, and confirmation [5]. The target of this paper is answering the factoid questions.

Another related issue is the corpus that the answers are extracted from. There are famous collections such as AQUAINT, which is used in TREC QA Track, but our approach was designed for web data, so the Internet texts are used.

The remainder of this paper is organized as follows: in section 2, related works will be mentioned. In section 3, the overall structure of our approach will be represented, and sections 4 and 5 are empirical study and conclusions.

II. Related Works

Evolutionary algorithms have been used in natural language processing problems such as ranking how-to questions [6], natural language tagging [7], text classification [8], and also question answering systems [9].

In Atkinson et al. [6] Genetic algorithm is used as evolutionary optimization for ranking how-to questions based on user generated contents. They represented each pair of question and answer sentences as a triplet {actor, action, object}. So, a how-to question is interpreted as the information required for how an actor (for example a person) can perform an action (for example solve) to an object (for example a Rubik) [6]. They performed crossover and other Genetic functions on these triplets in order to find the nearest content.

In Alba et al. [7] the Genetic algorithm is used for natural language tagging. They used language tags of each word as chromosomes. The fitness of a chromosome is a measure of the total correctness probability of its tag sequence, according to their training data [7].

In Tsai et al. [8] the Genetic algorithm is used for text classification.

Reference [9] used Artificial Immune System algorithm for a QA system.

In Heie et al. [10] a lexical-syntactic approach is used for evaluating sentences in a QA system. They used question keywords and their hand-made patterns for sentence evaluation.

III. Structure Of The Proposed Approach

All QA systems have three mutual parts: question analysis, information retrieval, and information extraction. In the question analysis part, system analyses the question, and extracts some features, for example keywords and question type, that will be used in the next parts. In IR part, system retrieves data elements based on the extracted features of the previous part. And in IE part, system extracts the requested information from the best retrieved element and represents it as the final answer. This part commonly contains a set of handcrafted rules or regular expressions, to extract information.

Our overall process is illustrated in Fig. 1, where the main task is categorized in three groups of document retrieval, sentence extraction, and word extraction, which will be described in the next three parts.

- **Document retrieval** part retrieves top-*n* (*n* is a variable and different values for it will be examined) related documents from a search engine like Google, and extracts their texts. The extraction phase is just from specific tags that have valuable text, for example from <p>*text*</p> tag.
- **Sentence extraction** part first enforces a preprocess phase on the retrieved texts that contains splitting their sentences and eliminating a set of stop-words and punctuations from them. Then it finds the best sentence by Memetic algorithm.
- **Word extraction** part extracts one or more words from the best sentence by means of our hand-made patterns and represents the word(s) as the final answer.

The first part needs no more description but the other two parts will be described with details in the next two parts.

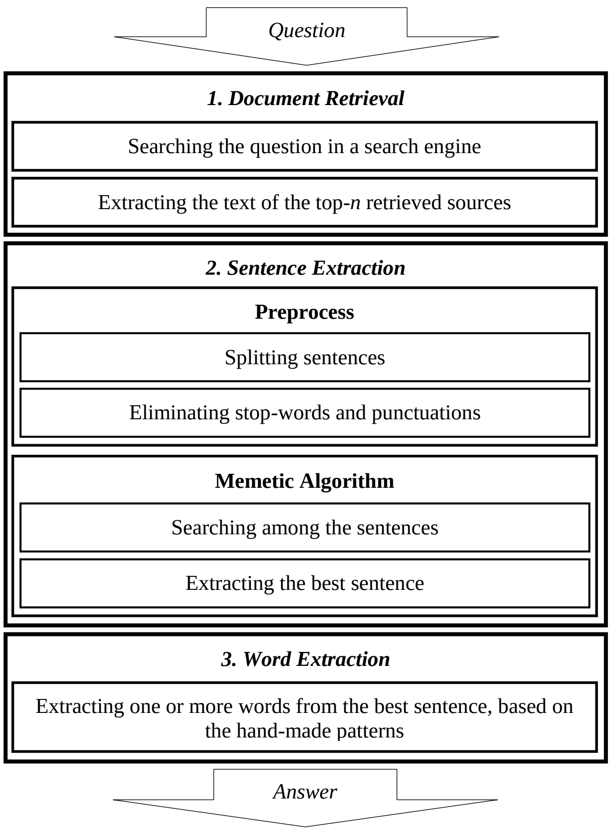
Question
1. Document Retrieval
Searching the question in a search engine
Extracting the text of the top-n retrieved sources
2. Sentence Extraction
Preprocess
Splitting sentences
Eliminating stop-words and punctuations
Memetic Algorithm
Searching among the sentences
Extracting the best sentence
3. Word Extraction
Extracting one or more words from the best sentence, based on the hand-made patterns
Answer

Fig. 1. The overall process of the proposed approach, named CallSimorgh.

A. Sentence Extraction

In our proposed approach, named CallSimorgh, Memetic algorithm is used for searching among the candidates and it employs lexical and syntactic features for its fitness function. The lexical features are based on *N*-grams [11] and keywords.

The syntactic feature that is used in the fitness function has a binary state that whether or not, the question and the candidate belong to a same pattern (one of our hand-made patterns). The details of Memetic algorithm used in our approach will be described first, and its fitness function will be described after that.

Memetic algorithm was introduced by Moscato [12] in 1989. Memes are knowledge bricks and they can be altered and integrated with other memes to produce new ones. The first Memetic algorithm was a developed version of Genetic algorithm, including a new function, called local search operator [13].

MA has been used to solve complex problems recently and it has resulted in a high performance. In order to describe MA, it is possible to define it with respect to its implementation features [13]. In this case, MA can be defined as follows: Memetic algorithm is a meta-heuristic, including of an evolutionary framework and a set of local search functions, which modify the generation in order to gain better individuals.

Here the MA has been chosen to investigate the effects of the local search functions in QA systems area.

The local search function lets some offspring to examine their adjacent candidates and update their fitness with the adjacent fitness, or update their fitness and also the sentence with the adjacent. The first procedure is called Baldwinian version and the second one is called Lamarckian version.

The pseudo code of MA that is used in this paper is illustrated in Fig. 2. The overall steps are as follows: selecting the initial population, evaluating the initial population fitness, performing the local search function on initial population, and saving the best answer. Then the algorithm iterates Rep_{count} times and performs the “For” loop. This loop contains: selecting the parents, performing crossover function, performing mutation function, evaluating the children, performing the local search function, replacing the new population, and saving the best answer. Each element of this code will be described in the following parts.

- **Population:** the preprocess part splits sentences and then assigns a number to each sentence starting from 1 to up. MA uses these numbers in binary form as the chromosomes or population. Each chromosome has a satellite data that indicates whether the sentence is first, last, or middle in its paragraph.
- **Inputs and outputs:** inputs are illustrated in Table 1, and output is the best sentence.
- **Selection function:** The algorithm chooses Pop_{size} members of the population based on Proportional selection that is, selecting each individual with the probability of (1).

$$P_{proportional\_selection}(x_i) = \frac{fit(x_i)}{\sum_j fit(x_j)} \cdot \tag{1}$$

- **Crossover function:** The crossover and mutation function should produce new sentences, and also give credit to adjacent sentences of a high fitness candidate. Both Uniform and One-point crossovers are used and the results are represented in section 4. The crossover function is performed on every two possible parents with the probability of P_c, unless both are the same.
- **Mutation function:** Fogarty [14] showed that dynamic mutation ratio increases the accuracy. He proposed (2) for changing the mutation probability P_m in t^{th} iteration for j^{th} bit (j is from 1 to n_b; n_b is the least significant bit). This method performs the Multiple-bits flipping mutation.

$$P_m(j,t) = \frac{28}{1905 \times 2^{j-1}} + \frac{0.4026}{2^{t+j-1}} \cdot \tag{2}$$

We updated this formula in order to apply the effect of the sentence fitness. If a sentence has a high fitness, its less significant bits will be changed with a high probability and if it has a low fitness, its more significant bits will be changed with a high probability. Our proposed equation is (3):

$$P'_m(j,t) = \left(\frac{28}{1905 \times 2^{j-1}} + \frac{0.4026}{2^{t+j-1}}\right) / (fit \times 1000) \cdot \tag{3}$$

Table 1. The inputs of MA in the proposed approach

Parameter	*Description*
Pop_{size}	Population size
Pop	Population with their fitness
$Prob_{size}$	Number of all sentences in documents
P_c	Probability of the crossover
P_{m1}	Probability of the first mutation
P_{m2}	Probability of the second mutation
$MPop_{size}$	Memetic population size
$MPop$	Memetic population
Rep_{count}	Number of reproduction
Par_{size}	Parent size
S_{best}	Best sentence

In order to set the final value between 0 and 1, the fitness is restricted to 3 digits of precision. So, *(fit×1000)* and $P\square_m$ are always between 0 and 1.

```
Function MemeticAlgorithm Returns best visited sentence
Inputs: $Pop_{size}$, $Prob_{size}$, $P_c$, $P_{m1}$, $P_{m2}$, $MPop_{size}$, $Rep_{count}$, $Par_{size}$
Output: $S_{best}$
    Pop = InitializePopulation($Pop_{size}$, $Prob_{size}$)
    EvaluatePopulation(Pop)
    MPop = SelectMemeticPopulation(Pop, $MPop_{size}$)
    Pop = LocalSearch(MPop)
    $S_{best}$ = GetBestSolution(Pop)
    For  i = 0 To i < $Rep_{count}$  Do
        Parents = SelectParents(Pop, $Par_{size}$)
        Foreach parent1, parent2 In Parents Do
            child1, child2 = Crossover(parent1, parent2, $P_c$)
            Children = Mutation(child1, $P_{m1}$)
            Children = Mutation(child1, $P_{m2}$)
      Children = Mutation(child1, $P_{m1}$)
            Children = Mutation(child2, $P_{m2}$)
        End
        EvaluatePopulation(Children)
        MPop = SelectMemeticPopulation(Children, $MPop_{size}$)
        Children = LocalSearch(MPop)
        Pop = Replace(Pop, Children)
        $S_{best}$ = GetBestSolution(Pop)
    End
Return $S_{best}$
```

Fig. 2. The pseudo code of MA in the proposed approach.

Another mutation is also used that is, changing an odd value to even and vice versa with the probability of P_{m2}. The reason of performing this mutation is that the results showed the offspring of even values with One-point and Uniform crossovers are mostly even too (same for odd values). So offspring values with a probability will be added or subtracted by 1.

- **Local search function:** The first local search that is used examines the next and previous K sentences of the randomly chosen high fitness sentences in order to replace a better neighbor with the original one. We proposed (4) for this purpose that with taking the fitness value, it gives a number that shows how many next and previous sentences must be examined.

$$K = \left\lfloor \frac{fit \times 5}{9.107} \right\rfloor . \qquad (4)$$

To make this equation, some assumptions have been made. First, the highest value for the fitness must be indicated. For this reason, sentences are restricted to contain 1 to 16 words. With this assumption the highest fitness will be 9.107 according to our fitness function. And also a maximum for K must be indicated, that is, at most how many words can be examined. With assumption of 5 for the highest K, the output of (4) will be between 0 and 5.

Our second local search is examining the first and last sentences in paragraph of the candidate. The first and last sentences of a paragraph mostly contain useful information that can be the final answer.

In local search function the algorithm chooses $MPop_{size}$ of the high fitness sentences with Proportional selection and performs the two local search functions on them.

- **Replacement function:** The replacement function chooses 0.1 of the best parents and 0.9 of the best offspring. These values have the best result between three series of values that we tested.
- **Termination condition:** The termination condition is that the number of iterations must reach to a certain number of Rep_{count}. Different values for this variable were tested and the results are represented in section 4.
- In the remaining paragraphs of this section, the details of the fitness function will be described.

The question Q and the candidate (sentence) A_C can be represented as a set of features. The feature groups that have been used for this purpose with their level of sophistication are illustrated in Fig. 3. Fig. 3 also contains the features that are used in this paper; containing lexical and syntactic features. The details of the features that we used will be described in the following paragraphs.

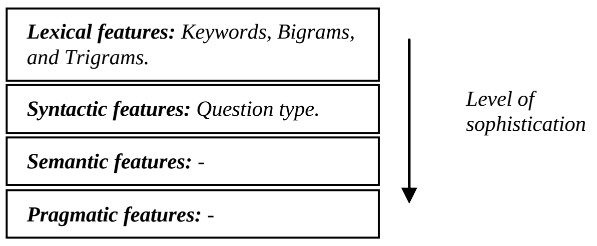
Lexical features: Keywords, Bigrams, and Trigrams.
Syntactic features: Question type.
Semantic features: -
Pragmatic features: -
Level of sophistication

Fig. 3. The groups of the features for evaluating the sentences, and the features that are used in the proposed approach.

With a list of candidates, the probability that A_C contains the final answer is shown as $P(A_C|Q)$. Each candidate that has the best value will be the sentence of the final answer A based on (5).

$$A = \arg\max_{A_C} P(A_C \mid Q) \cdot \qquad (5)$$

But the question Q can be represented as a set of features F:

$$F=\{Q_K, Q_G, Q_T\}. \qquad (6)$$

- **Q_T (Question type):** the words of the question, mostly the beginning ones that show the question type, like where, when, how many, and so on. The candidate type must match with the question type. Our hand-made question types will be described in part B of this section.
- **Q_K (Question keywords):** the keywords of the question, excluding the Q_T words and propositions.
- **Q_G (Question N-grams):** the bigrams and trigrams of the question. The Q_K unigrams are separated from Q_G because there is no need to examine all words, including the Q_T words and propositions, and also different weights are assigned to each one.

For example in question "Where is the capital of Iran?", "Where" and "is" belong to Q_T, "capital" and "Iran" belong to Q_K, and "capital of", "of Iran", and "capital of Iran" belong to Q_G. Sometimes only Q_G discriminates the candidates. For example in the previous question, two candidates "Tehran is the capital of Iran" and "Tehran is the capital of Tehran province and it is the most populated city of Iran" have question keywords and type, but only the first one has N-grams and this feature will separate them.

The main equation that is used for calculating the relevance of a candidate to a question is (7):

$$P(A_C|Q)=\alpha\times P(A_C|Q_T)+\beta\times P(A_C|Q_K)+\gamma\times P(A_C|Q_G). \qquad (7)$$

The weights α, β, and γ are assigned manually to this features based on our experiments. The values are 0/1, 0/5, and 0/4. The calculation details of each part of (7) will be described in the following parts.

Q_T is a set of question and answer hand-made patterns, called P that we made from TREC 2006 questions and answers. If the question and the answer patterns match, Q_T will become 1 and if they don't, it will become 0 based on (8). We constructed 26 hand-made patterns for this purpose.

$$P(A_C \mid Q_T) = \begin{cases} 1 & if \quad \exists i \in P \quad |A_C \in P_i \quad \cap \quad \mathrm{Q_T} \in P_i \\ 0 \text{ otherwise} \end{cases} . \qquad (8)$$

To implement the Q_K, first all subsets of the question, called S_i (i is from 1 to n), must be made. Then for every candidate, number of intersection of words with each subset must be calculated and divided by the length of the intersection. The division of these results by n will be the final value for Q_K based on (9).

$$P(A_C \mid Q_K) = \frac{1}{n} \times \sum_{i=1}^{n} \frac{|A_C \cap S_i|}{|S_i|} . \qquad (9)$$

And in order to calculate Q_G, all bigrams and trigrams of the question, called G_i (i is from 1 to m), must be made.

Then, for every candidate, Q_G will be division of number of matched N-grams by m based on (10).

$$P(A_C \mid Q_G) = \sum_{i=1}^{m} \frac{|A_C \cap G_i|}{m}. \tag{10}$$

B. Word Extraction

In order to show the overall process and also the word extraction part, an example will be mentioned. In question "For which newspaper does Krugman write?", the candidate "He is an op-ed columnist for The New York Times newspaper" will get the values 0.5, 1, and 0 for Q_K, Q_T, and Q_G and with multiplying these values by the weights, we will have 0.25, 0.1, and 0 and the fitness will be 0.35. Another candidate is "Krugman writes as a columnist in The New York Times newspaper." That will get 0.5, 0, and 0.2 values and 0.25, 0, and 0.08 with multiplying by the weights, and the fitness will be 0.33. The first candidate has a higher fitness value because the question type is matched, although second one has a bigram.

Now for extracting words, one of the patterns must be used. The related pattern for this question is:

QUESTION = <"For which [NOUN]">

And its related answer pattern (one of its alternations) is:

ANSWER = <"for [FINAL_ANSWER=The New York Times] [NOUN=newspaper]">

So, "The New York Times" will be the final answer. The noun phrase of the sentence can be indicated by a POS-tagger.

In the next paragraphs all of our patterns will be described, including some of the TREC 2007 questions which have that pattern.

- ***Where* questions:** The answer of the "Where" questions are mostly contains propositions like [in|near|at|from] that fallowed by one or more words that their first character is capital case, for example "in Tehran" or "at Tarbiat Modares". TREC example: "Where does Butcher live?".
- ***When* questions:** The answer of the "When" questions are mostly contains propositions like [in|on|at] that fallowed by a set of numbers and/or group of first character capital case words, for example "on March 22" or "in 2014". TREC example: "When did Irving Berlin die?".
- ***On what, In what,* and *By what* questions:** The answers of this group contains one of the [in|near|at|from|on|by] propositions that fallowed by a set of numbers and/or group of first character capital case words, for example "by Ferdosi" or "in Iran". TREC examples: "On what TV show does Hammond regularly appear?", "In what year was the IMG founded?", and "By what other name is Merrill known?".
- ***On which, At which, From which,* and *For which* questions:** The answers of this group contains one of the [in|near|at|from|on|for] propositions that fallowed by a set of numbers and/or group of first character capital case words, for example "from Persian gulf", "for Bazinama magazine". TREC examples: "At which university does Krugman teach?", "From which university did Krugman receive his Ph.D.?", "On which street is Merrill Lynch headquarters located?", "For which Kurt Weill song did Bobby Darin receive a Grammy award?", and sometimes in a form like "Hammond's shows appear on which network?".
- ***How old, How much,* and *How many* questions:** The answers of these questions are mostly contains numbers in the numeric or alphabetic form. TREC examples: "How old is Darrell Hammond?", "How many songs did Irving Berlin compose?", and "How much does an American Girl doll cost?".
- ***Which, How, Whose,* and *What* questions:** The answers of these questions are mostly the noun phrase of the answer's sentence, except the question keywords. TREC example: "Which company is Jay-Z president of?", "Whose place would Miers have taken on the Supreme Court?", "How did Irving Berlin die?", and "What is Krugman's academic specialty?".
- ***Who* questions:** The answers of the "Who" questions are mostly contains propositions like [by|with] that fallowed by one or more words that their first character is capital case, for example "by Cyrus", and "with Dariush". TREC example: "Who is Cunard's president and managing director?"

IV. Empirical Study

For evaluation of our approach, we used TREC 2007 (the last QA track) questions and answers. The questions are from 70 subjects and each one has about 4 factoid questions [15].

In TREC QA tracks, the accuracy measure has been used to assess system performance and it investigates whether the final answer is the exact requested answer or not, that means, not more and not less. We have tested TREC 2007 questions with web data because our purpose was building a web-based system. The TREC questions were searched in Google and sentences of the top-n web sites were used as the population of MA. The accuracy of the system was calculated for different values of n and the results are illustrated in Fig. 4. The best accuracy is 0.37 for n=19.

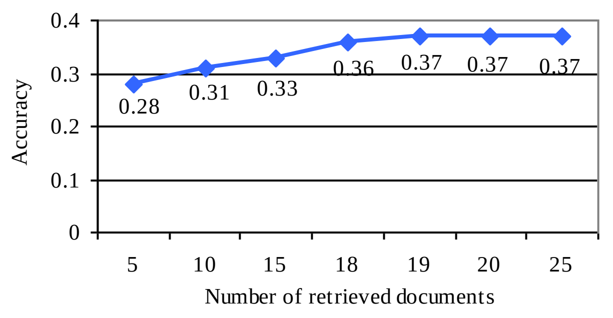

Fig. 4. The accuracy of the proposed approach with different *number of retrieved* documents values.

The *number of reproduction* variable is set to 12 in the previous test. But this variable has a direct effect in calculating the accuracy. So the accuracy with different values for this variable was calculated and the results are illustrated in Fig. 5. In this test and following tests, the *number of the retrieved documents* (or *n*) is equal to 19.

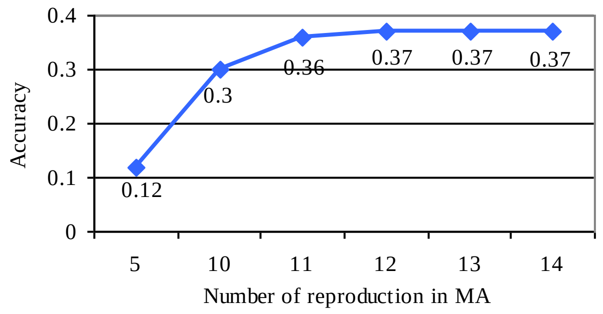

Fig. 5. The accuracy of the proposed approach with different *number of reproduction* values in MA.

In the both previous tests, Lamarckian approach is used for the local search function. The comparison of Lamarckian and Baldwinian approaches for the local search functions (for two local searches that are used) is illustrated in Fig. 6. The results show that Lamarckian approach has a better output in our implementation.

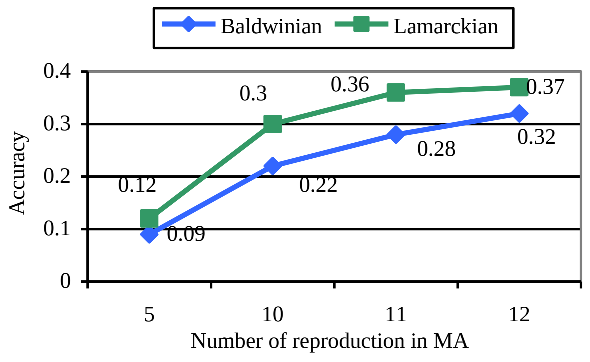

Fig. 6. The accuracy of the proposed approach with Lamarckian and Baldwinian approaches for the local search function.

In the three previous examples, Uniform approach is used for the crossover function. The comparison of One-point and Uniform approaches is illustrated in Fig. 7. The results show that the Uniform approach has a better output in our implementation.

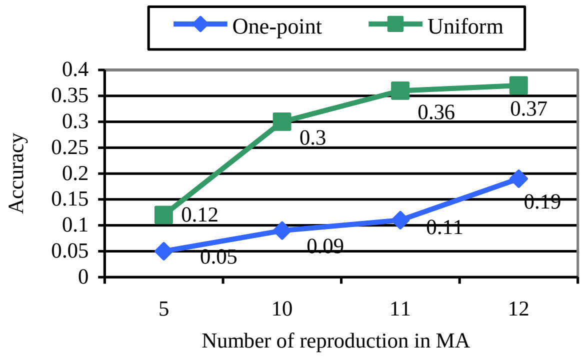

Fig. 7. The accuracy of the proposed approach with One-point and Uniform approaches for the crossover function.

In the four previous examples, our proposed mutation approach is used for the mutation function. Comparison of three different approaches, including: Single-bit flipping, Fogarty's [14] approach, and the proposed mutation (3) is illustrated in Fig. 8. The results show that our proposed approach has a better output in our implementation.

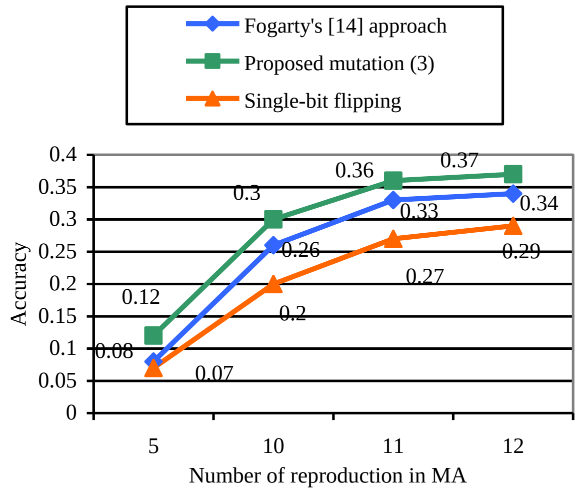

Fig. 8. The accuracy of the proposed approach with three types of mutation functions.

In order to highlight the usage of MA, the accuracy and process time comparisons between *evaluating all the sentences* approach and *searching with MA* approach are illustrated in Fig. 9 and Fig. 10. The results show that evaluating all the sentences has a better accuracy, 0.42 in best case, but this better accuracy has the side effect of more process time. The MA has a lower accuracy but the difference is tolerable with respect to much better process time.

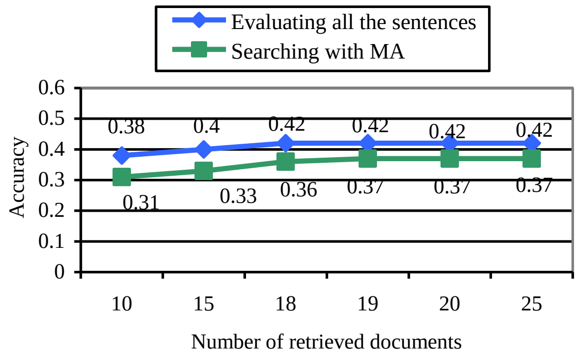

Fig. 9. Comparison of the accuracy of the proposed approach with the accuracy of the *evaluating all the sentences* approach.

The overall evaluation shows that our results are promising with sufficient retrieved documents. To quantify this sufficient value, we obtained a threshold for the *number of retrieved documents* variable (*n*=19).

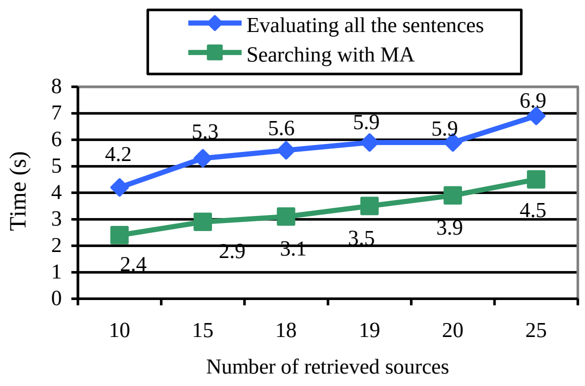

Fig. 9. Comparison of the process time of the proposed approach with the process time of the *evaluating all the sentences* approach.

V. Conclusion

In this paper we proposed an evolutionary approach for Question answering systems. Memetic algorithm is used because its local search function has a valuable effect to reach a high accuracy. The results showed that our approach is more efficient than examining the sentences one by one, with respect to the accuracy and process time. A set of equations are proposed for sentence fitness, local search, and dynamic mutation ratio. And a set of patterns are made for questions and answers. The best values for number of reproduction and number of retrieved documents are also investigated.

But we only performed answering of the factoid questions and answering to other types can be investigated in future works. The features that were used can also become more sophisticated in future works. And our hand-made patterns can be upgraded with evaluating more questions and answers.

References

[1] Nitin Indurkhya, Fred J. Damerau, *Handbook of Natural Language Processing*, 2nd ed., Chapman & Hall/CRC, 2010.

[2] Alexander Clark, Chris Fox, Shalom Lappin, *The Handbook of Computational Linguistics and Natural Language Processing*, Wiley-Blackwell, 2010.

[3] Kupiec, J., *“MURAX: A Robust Linguistic Approach for Question-Answering Using an Online Encyclopedia”*, In 16th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA, pp. 181–190, 1993.

[4] Boris Katz, Jimmy J. Lin, Sue Felshin, *“The START Multimedia Information System: Current Technology and Future Directions”*, in: Proceedings of the International Workshop on Multimedia Information Systems, pp. 117–123, 2002.

[5] Oleksandr Kolomiyets, Marie-Francine Moens, *“A Survey on Question Answering Technology from an Information Retrieval Perspective”*, Information Sciences, vol. 181, pp. 542–543, 2011.

[6] John Atkinson, Alejandro Figueroa, Christian Andrade, *"Evolutionary Optimization for Ranking How-to Questions Based on User-generated Contents"*, Expert Systems with Applications, vol. 40, pp. 7060–7068, 2013.

[7] Enrique Alba, Gabriel Luque, Lourdes Araujo, *“Natural Language Tagging with Genetic Algorithms”*, Information Processing Letters, vol. 100, pp. 173–182, 2006.

[8] Tsai, C.-F., et al., *“Evolutionary Instance Selection for Text Classification”*, J. Syst. Software (2014), http://dx.doi.org/10.1016/j.jss.2013.12.034, in press.

[9] Mohsen Shakiba Fakhr, Mohammad Saniee Abadeh, *"AISQA - An Artificial Immune Question Answering System"*, International Journal of Information Technology and Computer Science (IJMECS), vol. 4, No. 3, 2012.

[10] Matthias H. Heie, Edward W.D. Whittaker, Sadaoki Furui, *“Question Answering Using Statistical Language Modeling”*, Computer Speech and Language, vol. 26, pp. 193–209, 2012.

[11] Markov, A., *“An Example of Statistical Investigation in the Text of Eugene Onegin Illustrating Coupling of Tests in Chains”*. Proc Academy of Sciences of St. Petersburg, vol. 7, 1913.

[12] P. Moscato, *“On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms”*, Caltech concurrent computation program (report 826), 1989.

[13] Ferrante Neri, Carlos Cotta, *“Memetic Algorithms and Memetic Computing Optimization: A Literature Review”*, Swarm and Evolutionary Computation, vol. 2, pp. 1–14, 2012.

[14] T. C. Fogarty, *“Varying the Probability of Mutation in the Genetic Algorithm”*, In Proceedings of the third international conference on genetic algorithms, San Mateo, C. A., Morgan Kaufmann, pp. 104-109, 1989.

[15] Hoa Trang Dang, Diane Kelly, Jimmy Lin, *“Overview of the TREC 2007 Question Answering Track”*, In Proceedings of the Sixteenth Text Retrieval Conference, 2007.

Authors’ Profiles

Iman Khodadi was born in Tehran, Iran. He is currently a M.Sc. student in Software Engineering at Tarbiat Modares University, Tehran, Iran, in 2014. He received his B.Sc. in Software Engineering from Science and Culture University, Tehran, Iran, in 2012. His research interests are natural language processing, evolutionary computing, and machine learning. He is a Lecturer at Science and Culture University, Tehran, Iran.

Mohammad Saniee Abadeh received his B.S. degree in Computer Engineering from Isfahan University of Technology, Isfahan, Iran, in 2001, the M.S. degree in Artificial Intelligence from Iran University of Science and Technology, Tehran, Iran, in 2003 and his Ph.D. degree in Artificial Intelligence at the Department of Computer Engineering in Sharif University of Technology, Tehran, Iran in February 2008. His research has focused on developing advanced meta-heuristic algorithms for data mining and knowledge discovery purposes. His interests include data mining, bio-inspired computing, computational intelligence, evolutionary algorithms, fuzzy genetic systems and Memetic algorithms. He is currently a faculty member at the Faculty of Electrical and Computer Engineering at Tarbiat Modares University.