Journal of mathematics and computer science 11 (2014), 330-342

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

TabSum- A new Persian text summarizer

Saeid Masoumi[*], Mohammad-Reza Feizi-Derakhshi[#], RaziyehTabatabaei[*]

[*]*M.Sc in Software Engineering at University of Tabriz, Tabriz, Iran*
[#]*Assistant professor at University of Tabriz, Tabriz, Iran*

[*]***Saeid_masoumi_88@yahoo.com***

Article history:
Received May, 2014
Accepted June, 2014
Available online July 2014

Abstract

With the rapid increase in the amount of online text information, it became more important to have tools that would help users distinguish the important content. Automatic text summarization attempts to address this problem by taking an input text and extracting the most important content of it. However, the determination of the salience of information in the text depends on different factors and remains as a key problem of automatic text summarization. In the literature, there are some studies that use lexical chains as an indicator of lexical cohesion in the text and as an intermediate representation for text summarization. Also, some studies make use of genetic algorithms in order to examine some manually generated summaries and learn the patterns in the text which lead to the summaries by identifying relevant features which are most correlated with human generated summaries. In this study, we combine these two approaches of summarization. Firstly, some of preprocessing operations like normalizer, tokenizer, stop word remover, stemmer, and POS tagger are done on the text. After that for each sentence we have only semantic words that are independent. Then, by set of position, thematic, and coherence features we score sentences. The final score of each sentence will be the integration of those features.

Each feature has its own weight and should be identified to have well summary. For this reason first system goes throw learning phase to determine ache feature weight by genetic algorithm. The next phase is testing phase. In this phase system receives new documents and uses Persian WordNet and lexical chains to extract deep level of knowledge about the text. This knowledge is combined with other higher level analysis results. Finally, sentences are scored, sorted, and selected and summary is made.

We evaluated our proposed system by two methods. 1) Precision/recall, 2) TabEval (a new evaluation tool for Persian text summarizers). We compared our system with two other Persian summarizers (FarsiSum, Ijaz). Results showed that our system had higher performance rather than others (i.e. higher precision/recall average and the best average score of TabEval).

Keywords: Summarization, Text Summarizer, Mono-Document Summarization, Extractive Summarization, Persian Text Summarization.

1. Introduction

Nowadays there is a vast amount of textual information on the web. It is too difficult for users to read and locate their needs in such a bulky information repository. Therefore, a summarization system would be helpful to allow users (1) to find the resources they need more rapidly and (2) to access the most important parts of the texts. A summary is defined to be a “brief restatement within the document (usually at the end) of its salient findings and conclusions” that ”is intended to complete the orientation of a reader who has studied the preceding text” [1]. It contains the most important information about the document.

In other words, text summarization is the process of extracting the most important parts of information from source document(s) to produce a compact version for a particular user or task.

Automatic text summarization can be used in various areas of applications such as intelligent tutoring systems, telecommunication industry, information extraction and text mining, question answering, news broadcasting and word processing tools.

The most fundamental distinction that can be made between summarization types is the one between extracts and abstracts. An extract is a summary consisting entirely of material copied from the input. On the other hand, an abstract is a summary at least some of whose material is not present in the input [2]. Extracts are generally produced by shallow approaches, where the sentences of the text are analyzed to a syntactic level. These approaches extract salient parts of the source text and present them. On the other hand, abstracts are produced by deeper approaches. These approaches analyze the source text to a sentential semantics level. In order to retrieve important information from the text, approaches like template filling [10], term rewriting [11] and concept hierarchy [12] are used. After the analysis phase, these approaches go through a synthesis phase, which usually involves natural language generation.

Most of the studies in this area are based on extraction. While abstraction deals heavily with natural language processing, extraction can be viewed as selecting the most important parts of the original document and concatenating them to form the summary.

In this paper, we introduce TabSum, an automatic summarization system developed for extractive summarizing mono-documents in the Persian language. The fundamental components of this system are normalizer, tokenizer, stop word remover, stemmer, and POS tagger. Moreover, the concept of lexical chain and WordNet are used to extract the coherences between words. This system processes text via some feature sets like position, thematic, and coherence.

The remainder of the paper is organized as follows: Section 2 discusses related works. Section 3 introduces TabSum and Section 4 shows the experimental results. Finally, the Conclusion discusses current-and future efforts being made to improve the summaries generated.

2. Related works

The main steps of text summarization are identifying the essential content, “understanding” it clearly and generating a short text. Understanding the major emphasis of a text is a very hard problem of NLP [3]. This process involves many techniques including semantic analysis, discourse processing and inferential interpretation and so on. Text Summarization methods can be classified into extractive and abstractive summarization. An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. The importance of sentences is decided based on statistical and linguistic features of sentences. Simply extractive model based on selecting some pieces of original text in the other hand Abstractive models based on paraphrasing and generating a shorter text. It’s clear that the implementing of abstractive models is more difficult than the Extractive one. Most of the researcher chose the extractive methods.

There are many summarization methods and systems available for languages such as English. Although some of them claim to be language-independent, they need at least language resources to work with. The lack or shortage of these resources such as training and test data, lexical ontologies or semantic lexicons, lists of stop words and cue-words and even fundamental language processing tools such as reliable tokenizers, stemmers and parsers all make text summarization a hard task for languages such as Persian with less resources. In contrast to English summarization systems, summarization document written in Persian is a new, ongoing research effort.

The oldest work on Persian text summarization is FarsiSum [4]. It is an HTTP client/server application programmed in Perl based on SweSum [5], a summarizer for the Swedish language. FarsiSum extracts data from single documents with the main body of language independent modules implemented in SweSum. In FarsiSum, the Persian stop-list has been added in Unicode format and the interface modules are adapted to accept Persian texts.

The second work is a single document Persian text extractor based on lexical chains and graph-based methods [8]. This System uses 5 measures: namely similarity to other sentences, similarity to user's query, similarity to the title and the number of common words and cue words to score a sentence. Some specific Persian resources to prepare the chains and graphs are used in its scoring module.

Honarpisheh and his colleagues [9] have developed a multi-document multi-lingual text summarizer based on singular value decomposition and hierarchical clustering. Their approach relies on only two resources for any language: a word segmentation system and a dictionary of words in conjunction with their document frequencies. The summarizer initially receives a collection of related documents and transforms them into a matrix; it then applies singular value decomposition to the resulting matrix. Using a binary hierarchical clustering algorithm, it then chooses the most important sentences of the most important clusters to create the summary.

The next one is Parsumist [6]. It exploits a combination of statistical, semantic and heuristic-improved methods. It can generate generic or topic/ query- driven extracts summaries for single- or multiple Persian documents.

The last system I introduced is a summarization system that it work base on fuzzy logic [7]. They used MATLAB because it is possible to simulate fuzzy logic in this software. To do so; first, they consider each characteristic of a text such as sentence length, similarity to little, similarity to keyword and etc, which are the input of fuzzy system. Then, they enter all the rules needed for summarization, in the knowledge base of this system.

Afterward, a value from zero to one is obtained for each sentence in the output based on sentence characteristics and the available rules in the knowledge base. The obtained value in the output determines the degree of the importance of the sentence in the final summary.

Our system is somehow similar to the system in [6] as they uses lexical chains as well, they have improved their work by using semantic features and representing a conceptual meaning of the text using synonym sets, applying redundancy checking, smoothing the summary for coherence and making it applicable.

3. Proposed system

The aim of this paper is to combine two approaches of summarization. Firstly, lexical chains are computed to exploit the lexical cohesion that exists in the text. Then, this deep level of knowledge about the text is combined with other higher level analysis results such as location analysis and thematic analysis. Finally, all these results that give different levels of knowledge about the text are combined to obtain a general understanding.

In this thesis, we use a sentence extraction procedure that makes use of these properties of the text to weight the sentences. Each sentence in a text is given a sentence score that is calculated using the different text feature scores. After that, the sentences are sorted in descending order of their score values. And then appropriate number of highest score sentences are selected from the text to form the summary, according to the summarization ratio.

While weighting the sentences, not all the properties of the text will have the same importance. However, weighting the text feature scores with predetermined constant weights does not seem to be powerful enough for a good summarization. For this reason, the system first goes through a training phase, where the weights of each text feature are learned using machine learning methods.

In order to be able to learn the weights of different text features, a set of manually summarized documents is used. These human generated extracts are expected to give an idea about the patterns which lead to the summaries. In this study, we made a corpus from some of Iranian famous newspapers. Our corpus has 30 documents and each document has 5 ideal summaries.

After the feature score weights are learned through the training phase, the system will go through a testing phase where new documents are introduced to the system for summarization. In this phase, sentence scores will be calculated for each sentence in a document using the text feature scores for that sentence and their respective score weights. Then the sentences will be sorted in a descending order of their score values, and the highest score sentences will be selected to form the extractive summary.

3.1. Text Features

In this system, the sentences are modeled as vectors of features extracted from the text. The system uses 8 text features to score sentences. For each sentence of a document, a sentence score will be calculated using the feature scores of these text features for that sentence. Each feature score can have a value between 0 and 1.

The text features used in this system are grouped into three classes, according to their level of text analysis. Table 1 shows the features and their corresponding classes.

Table 1: Text features

Location Features	Sentence Location
	Sentence Relative Length
Thematic Features	Average TF
	Sentence Resemblance to Title
	Sentence Centrality
Cohesion Features	Number of Synonym Links
	Number of Co-occurrence Links
	Lexical Chain Score

3.2.Location Features

These features exploit the structure of the text at a shallow level of analysis. Depending on the location and length of the sentence, the importance of its content is tried to be predicted. Based on this prediction, a sentence will be given a higher or a lower score.

3.2.1.Sentence Location

This feature scores the sentences according to their position in the text. In this work, we assume that the first sentences of the text are the most important ones. So, the first sentence of a document gets a score value of 1, the second sentence gets 0.9, the tenth sentence gets 0.1 and the rest of the sentences get 0.

3.2.2. Sentence Relative Length

This feature uses the sentence length to score a sentence, assuming that longer sentences contain more information and have a higher possibility to be in the summary. Thus, shorter sentences are penalized. The feature score is calculated as follows for the sentence s in the document d:

$$SRL(s,d) = \frac{length(s)}{maxSentenceLength(d)} \tag{1}$$

3.3.Thematic Features

These features study the text more deeply to analyze the term based properties. The term frequencies of each document and each sentence are calculated.

3.3.1.Average TF

This feature calculates the Term Frequency (TF) score for each term in a sentence and takes their average. The TF metric makes two assumptions:

(i) Multiple appearances of a term in a document are more important than single appearances.

(ii) Length of the document should not affect the importance of the terms.

The TF score for a term t in the document d is calculated as follows:

$$TF(t,d) = \frac{frequency\ Of\ Term\ In\ Document(t,d)}{maxTermFrequency(d)} \tag{2}$$

So, the feature score for a sentence s is the average of the TF scores of all the terms in s.

3.3.2.Sentence Resemblance to Title

This feature considers the vocabulary overlap between a sentence and the document title. If a sentence has many words in common with the document title, it is assumed to be related to the main topic of the document. So, it is assumed to have more chance to be in the summary.

The feature score is calculated as follows for a sentence s:

$$SRT(s) = \frac{|m \cap k|}{|m \cup k|} \tag{3}$$

where m is the set of terms that occur in sentence s, and k is the set of terms that occur in the title.

3.3.3.Sentence Centrality

This feature considers the vocabulary overlap between a sentence and the other sentences in the document. If a sentence has many words in common with the rest of the document, it is assumed to be about an important topic in the document. So, it is assumed to have more chance to be in the summary.

The feature score is calculated as follows for a sentence s in the document d:

$$SC(s,d) = \frac{m}{k} \tag{4}$$

Where m is the number of terms that occur both in sentence s and in a sentence of document d other than s, and k is the total number of terms in document d.

3.4. Cohesion Features

Cohesion can be defined as the way certain words or grammatical features of a sentence can connect it to its predecessors and successors in a text. Cohesion is brought about by linguistic devices such as repetition, synonymy, anaphora and ellipsis. In this system, three cohesion based features are used.

3.4.1. Number of Synonym Links

In order to compute this feature, first the nouns in a sentence are extracted by a Persian part-of-speech tagger. Then nouns in the given sentence s are compared to the nouns in other sentences in the document. This comparison is made by taking two nouns from the two sentences and looking whether they have a synset in common in WordNet. For instance, if a noun from sentence s has a synset in common with a noun from another sentence t, this means there is a synonym link between the sentences s and t.

So, the feature score is calculated as follows for a sentence s in the document d:

$$NSL(s) = \frac{n}{k} \tag{5}$$

Where n is the number of synonym links of sentence s (i.e., the number of sentences t) and k is the total number of sentences in document d.

3.4.2. Number of Co-occurrence Links

In order to compute this feature, first all the bigrams in the document are considered and their frequencies are calculated. If a bigram in a document has a frequency greater than one, then this bigram is assumed to be a collocation.

Secondly, terms of the given sentence s are compared to the terms in other sentences in the document d. This comparison procedure checks if a term from sentence s forms a collocation with a term from another sentence. If it does, this means there is a co-occurrence link between this sentence and the sentence s.

So, the feature score is calculated as follows for a sentence s in the document d:

$$NCL(s) = \frac{n}{k} \tag{6}$$

Where n is the number of co-occurrence links of sentence s and k is the total number of sentences in document d.

3.4.3. Lexical Chain Score

In order to use lexical chains as a means for scoring the sentences of a document, first the chains are computed for the whole document. Then these constructed chains are scored and the strongest ones among them are selected. Finally, sentences of the document are scored according to their inclusion of strong chain words. The details of the lexical chain computing and scoring processes are explained in the next part.

So, after the chains are constructed and scored for a document d, the lexical chain score of a sentence s is as follows:

$$LC(s) = \frac{\sum_i frequency(i) | i \in s \text{ and } i \text{ is a word in a strong chain}}{maxLCScore(d)} \tag{7}$$

3.5.Computing Lexical Chain Scores

Lexical chains are composed of words that have a lexical relation. In order to find these relations among words, Persian WordNet lexical knowledge base is used. In WordNet, words have a number of meanings corresponding to different senses. Each sense of a word belongs to a synset (a set of words that are synonyms). This means, ambiguous words may be present in more than one synset. Synsets may be related to each other with different types of relations (like hyponym, hypernym, antonym, etc.).

In computing lexical chains, each word must belong to exactly one lexical chain. There are two challenges for this. First, there may be more than one sense for ambiguous words and a heuristic must be used to determine the correct sense of the word. Second, a word may be related to words in different chains. For example, a word may be in the same synset with a word in one lexical chain, while having a hyponym/hypernym relationship with another word in another chain. The aim here is to find the best way of grouping words that will result in the longest and strongest lexical chains.

This process consists of four steps:

- Selecting candidate words
- Constructing lexical chains from these words
- Scoring these chains
- Selecting the strong chains

3.5.1. Selecting Candidate Words

Candidate words for lexical chains are the nouns. So, firstly, the text is put through Persian part of speech (POS) tagging. This tagging process is necessary to determine the nouns in the document. After the nouns are determined, they are added to the lexical chain candidate words list.

3.5.2. Constructing Lexical Chains from Candidate Words

When the candidate words list is constructed, the words in the list are sorted in ascending order of their number of senses. This way, the words with the least number of senses (i.e., the least ambiguous ones) are treated first.

For each word, the system tries to find an appropriate chain that the candidate word can be added, according to a relatedness criterion among the members of the chain and the candidate word. This search continues for every sense of the candidate word, until an appropriate chain is found. If such a chain is found, the current sense of the candidate word is set to be the disambiguated sense, and the word is added to the lexical chain.

This relatedness criterion compares each member of the chain to the candidate word to find out if

- the sense of the lexical chain word belongs to the same synset as the sense of the candidate word
- the synset of the lexical chain word has a hyponym relation with the synset of the candidate word
- the synset of the lexical chain word has a hypernym relation with the synset of the candidate word
- the synset of the lexical chain word has a co-occurrence relation with the synset of the candidate word
- the synset of the lexical chain word has a related-to relation with the synset of the candidate word

If the system cannot find an appropriate lexical chain to add the candidate word for any sense of the word, a new chain is constructed for every sense of the word. For instance, this will create five new lexical chains in the system for a word that has five different senses. This way, when a new candidate word is compared to these chains, it will be possible to find a relation between the new candidate word and any of these five senses of the previous word.

The problem here is that, there may be more than one chain in the system for the same word, which continue growing at the same time. For example a word with two senses will create two different lexical chains. When a second word arrives, it may be related to the first sense of the first word and be added to the first chain. After that, if a third word arrives and is related to the second sense of the first word, it will be added to the second chain and the two chains will continue growing independently. This will conflict the requirement that says each word must belong to exactly one lexical chain.

This problem is eliminated by removing the rest of the chains for the word in the system, as soon as a second word is related with one of the senses of the word.

3.5.3. Scoring the Chains

Once the lexical chains are computed, each chain is given a score number that shows its strength. This score number will be used to select the strongest chains of the document and the sentences that contain words that occur in strong chains will be given a higher sentence score.

The score of a chain depends both on its length and on its homogeneity. The length of a chain is the number of occurrences of members of the chain. Its homogeneity is inversely related with its diversity. For instance, if there are three distinct words in a chain that has seven members, this chain is assumed to be stronger than a chain with the same number of members, but five distinct words.

So, the score of a chain is calculated as follows:

$$score = length * homogeneity \qquad (8)$$

Where

$$homogeneity = 1 - \frac{number\ Of\ Distinct\ Occurrences}{length} \qquad (9)$$

3.5.4. Selecting the Strong Chains

In this work, strong lexical chains are assumed to be the ones whose score exceeds the average of the chain scores by standard deviation. That is, a strong chain must satisfy the criterion;

$$score(chain) > average(chainScores) + standardDeviation(chainScores) \qquad (10)$$

Moreover, chains that contain only one word are not accepted as strong chains.

3.6.Feature Weighting With Genetic Algorithm

In this paper we use 8 different text features to score sentences. After each sentence of a document is scored, the sentences of the document are sorted according to their scores and the highest scored sentences are selected to form the summary of that document.

However, not all the feature scores have the same importance while calculating the sentence score. A sentence score is a weighted sum of that sentence's feature scores. Each feature may have a different weight and these weights are learned from the manually summarized documents, using machine learning methods. Thus, a sentence's score is calculated as follows:

$$Score(s) = w1f1(s) + w2f2(s) + w3f3(s) + w4f4(s) + w5f5(s) + w6f6(s) + w7f7(s) + w8f8(s) \quad (11)$$

$f_i$are the feature scores of each sentence and their values can range from 0 to 1. They are computed separately for each sentence s. $\mathrm{w_i}$can range from 0 to 15. They are learned using genetic algorithms.

The system has two modes of operation: Training Mode (where the feature weights are learned from the corpus) and Testing Mode (where new documents are summarized using the weighted feature scores). Figure 1 shows these two modes.

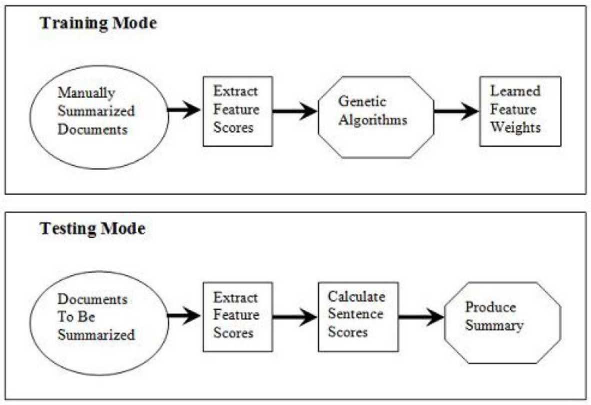


Figure 1: Model of the automatic summarization system

In the training mode, the weights of each feature are learned by the system, using the manually summarized documents.

Firstly, the text feature scores are calculated for every sentence. Since these scores are constant for each sentence, they are calculated once before the machine learning procedure starts.

Then, these feature scores are integrated by a weighted score function in order to score each sentence. On each iteration of the training routine, random weights are assigned to 8 text features, and thus sentence scores are calculated. According to these sentence scores, a summary is generated for each document in the corpus. The precision of each automatically generated summary when compared to its manually generated summary is calculated using the following formula:

$$P = \frac{|S \cap T|}{|S|} \quad (12)$$

where T is the reference summary and S is the machine generated summary.

The average of these precisions gives the performance of that iteration. This performance metric shows how appropriate the random weights of that iteration werefor this summarization system. The best of all iterations is selected using geneticalgorithms.

In this work, each individual of the population is a vector of feature weights. There are 8 features and each feature weight can have a value between 0 and 15. When these weights are represented in binary mode using 4 bits, they form a vector of length 32. This vector is the individual of the GAs.

The fitness of an individual is the performance metric. Each individual represents a set of feature weights. Using these weights, sentence scores are calculated and summaries are generated for each document in the corpus.

The precision of the automatically generated summary when compared to the manually generated summary is calculated for each document and the average of these precision values is the fitness of that individual.

In the training mode, genetic algorithms were run with the following properties:

- There are 100 individuals in a population.
- At each iteration, one fittest individual is selected for the next generation as an elite.
- Rest of individuals is selected through selection, crossing over and mutation.
    - Rolette wheel for selection
    - Two point crossover
    - Swap for mutation
- The algorithms are run for 1000 iterations.
- Summarization ratio is 30

Table 6.2 shows the weights of each text feature calculated by the training module.

Table 2: feature weights from learning phase

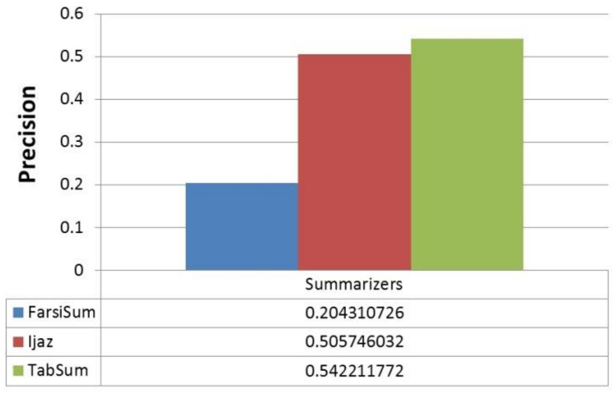
Sentence Location	Sentence Relative Length	Average TF	Sentence Resemblance to Title	Sentence Centrality	Number of Co-occurrence Links	Number of Synonym Links	Lexical Chain Score
7	12	6	14	10	13	14	1

4. Evaluation

We used the intrinsic evaluation method and a summary evaluation tool (TabEval). Frist one judges the quality of a summary based on the coverage between it and the manual summary and the second one uses semantic relation between sentences of machine and human summaries. For testing the performance of our proposed system we compared it with two of exist Persian summarizers (FarsiSum, Ijaz).

First, we used precision and recall as the performance measures. Assuming that T is the manual summary and S is the machine generated summary, the measurement of precision P and recall R are defined as follows:

$$P = \frac{|S \cap T|}{|S|} \quad , \quad R = \frac{|S \cap T|}{|T|}$$



	Summarizers
🟦 FarsiSum	0.204310726
🟥 Ijaz	0.505746032
🟩 TabSum	0.542211772

Figure 2:results of evaluation by Precision metric

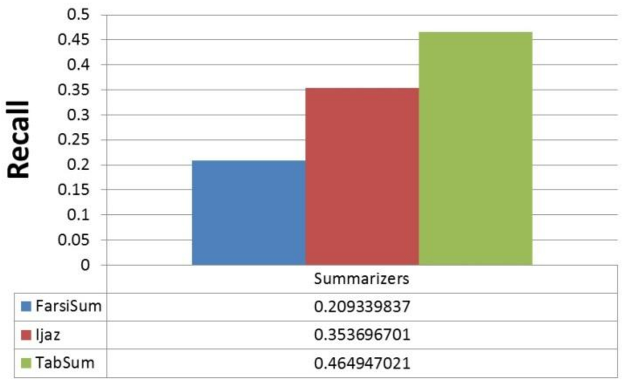


Figure 3:results of evaluation by Recall metric

We used F-measure metric for balancing amounts between precision and recall where it is defined as:

$$F = \frac{2 * P * R}{P + R}$$

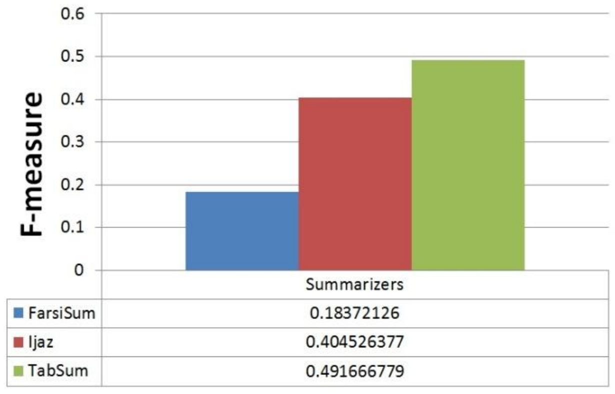
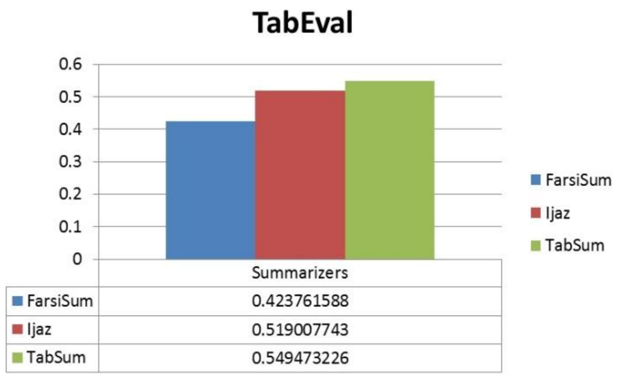


Figure 4:results of evaluation by F-measure metric

Results of intrinsic evaluation showed that our proposed system has better Precision and Recall among all systems and its performance is acceptable too.

TabEval evaluates Persian text summarizers semantically. We sent our system’s results through it and got the score.



	Summarizers
FarsiSum	0.423761588
Ijaz	0.519007743
TabSum	0.549473226

Figure 5:results of evaluation by TabEval tool

The results of evaluating proposed system with TabEval show that our system is the best Persian summarizer and considers semantic metrics besides lexical ones.

5. Conclusion

In this study, we have combined two approaches used in automatic text summarization: using Lexical Chains to detect the lexical cohesion that exists throughout the text, and using Genetic Algorithms to efficiently learn the weights to be used in sentence scoring.

We have computed lexical chains in a text depending on the lexical relations among words in the text. These relations were determined using WordNet. All these computed chains were scored in order to select the strongest chains in a given text.

Then we have computed different text features for each sentence in a text. These features tried to analyze the sentence to different levels. We used lexical chains as the basis for one of these feature functions. We gave higher lexical chain feature scores to sentences that contained more strong lexical chain words. After all the feature scores were computed, we used genetic algorithms to determine the appropriate feature weights. These feature weights were then used to score the sentences in the testing mode. The highest scored sentences were selected to be included in the summary.

The contribution of this study is that it puts the benefits of lexical chain approach and genetic algorithms approach together. It combines information coming from different levels of analysis on text. Different from other work in this area, location features like sentence location, thematic features like sentence centrality and cohesion features like sentence inclusion of strong lexical chain words are all considered together in this study. It also makes use of machine learning approach to determine the coefficients of this combination.

As a future work, the model can be tested on different text genres. The corpus we used in this study consisted of newswire documents. However, the tests can be run on scientific documents or some other genre in order to see the change in the text feature performances and in the overall system performance.

References

[1] Kiani, A. and M. R. Akbarzadeh, “Automatic Text Summarization Using: Hybrid Fuzzy GA-GP", In IEEE International Conference on Fuzzy Systems, 2006.
[2] Mani, I., Automatic Summarization, John Benjamins Publishing Company, Amsterdam/Philadelphia, 2001.

[3] Inderjcet Main, the MITRE corporation 11493 Sanset Hills noad , USA , 2003.
[4] Mazdak, N., 2004. "FarsiSum-a persian text summarizer". Master thesis,Department of linguistics, Stockholm University.
[5] Dalianis, H., 2000. "SweSum-A Text Summarizer for Swedish, Technical report", TRITANA-P0015, IPLab-174.
[6] M.Shamsfard, T.Akhavan and M.E.Joorabchi, 2009. "Persian Document Summarization by Parsumist". World Applied Sciences Journal 7 (Special Issue of Computer & IT): 199- 205, 2009.
[7] F.Kiyomarsi and F.R.Esfahani. "Optimizing Persian Text Summarization Based on Fuzzy Logic Approach". 2011 International Conference on Intelligent Building and Management.
[8] Karimi, Z. and M. Shamsfard, 2006. "Summarization of Persian texts".In Proceedings of 11th International CSI computer Conference, Tehran, Iran.
[9] Honarpisheh, M.A., G. Ghasem-sani and G. Mirroshandel, 2008. "A Multi-Document Multi-Lingual Automatic Summarization System". Proceedings of the 3rd Joint Conference on Natural Language Processing, pp: 733-738.
[10] Jong, G. F. D., “An overview of the FRUMP system", W. G. Lehnert and M. H. Ringle (Editors), Strategies for Natural Language Processing, Erlbaum, Hillsdale, NJ, 1982.
[11] Hahn, U. and I. Mani, “Automatic Text Summarization: Methods, Systems, and Evaluations", In International Joint Conference on Artificial Intelligence (IJCAI), 1998.
[12] Hovy, E. and C. Y. Lin, “Automated Text Summarization in SUMMARIST", I. Mani and M. T. Maybury (Editors), Advances in Automatic Text Summarization, pp. 81{94, The MIT Press, Cambridge, MA, 1999.