

# Graph-based Semantical Extractive Text Analysis

Mina Samizadeh

**Abstract**—In the past few decades, there has been an explosion in the amount of available data produced from various sources with different topics. The availability of this enormous data necessitates us to adopt effective computational tools to explore the data. This leads to an intense growing interest in the research community to develop computational methods focused on processing this text data. A line of study focused on condensing the text so that we are able to get a higher level of understanding in a shorter time. The two important tasks to do this are keyword extraction and text summarization. In keyword extraction, we are interested in finding the key important words from a text. This makes us familiar with the general topic of a text. In text summarization, we are interested in producing a short-length text which includes important information about the document. The TextRank algorithm, an unsupervised learning method that is an extension of the PageRank (algorithm which is the base algorithm of Google search engine for searching pages and ranking them) has shown its efficacy in large-scale text mining, especially for text summarization and keyword extraction. This algorithm can automatically extract the important parts of a text (keywords or sentences) and declare them as the result. However, this algorithm neglects the semantic similarity between the different parts. In this work, we improved the results of the TextRank algorithm by incorporating the semantic similarity between parts of the text. Aside from keyword extraction and text summarization, we develop a topic clustering algorithm based on our framework which can be used individually or as a part of generating the summary to overcome coverage problems.

**Index Terms**—Keyword Extraction; n-gram Extraction; Text summarization; Topic Clustering; Semantic Analysis.

## I. INTRODUCTION

The goal of text summarization is extracting a few important sentences from the document while preserving the main idea of the text. A good summary keeps the main topic of the text simultaneously, occupy less space than the original document. This is a very complex problem because it needs to emulate the cognitive capacity of human beings to generate summaries and still is an open problem in natural language processing. Since it is a difficult task most of the researches of literature focused on the extractive aspect of summarization which returns important sentences of documents without any change in the sentences in contrast to the abstractive aspect which generates new sentences that reveal the main topic of the text and it requires more resources to be trained. A summary generated by an automatic text summarizer should consist of the most relevant information in a document and at the same time, it should be condensed to take less space in comparison to the source document. Nevertheless, automatically generating summaries is a challenging task. An issue in extractive text summarization is the amount of summary coverage and diversity of topics in the input document. In an optimized summarizer, results cover

a sufficient amount of topics especially in situations which the document contains multiple topics.

In an attempt to solve these issues, in this work we proposed an extractive summarization method based on the TextRank algorithm to find important sentences in a documents and using a word2vec model to include semantics in the summary. Moreover, we proposed a topic clustering method which can also be used as a post-processing technique on the text summarization results. In this topic-clustering method, we consider semantic as well as similar keywords to measure the similarities between sentences in a text and categorize them together. Further, we proposed a keyword extraction method based on the textRank algorithm which, similar to the proposed text summarization method, considers the semantics as well as the statistics of the words in the sentences.

To summarize our works:

- Proposing a graph-based extractive semantical text summarization to summarize texts in any language
- Proposing a graph-based extractive semantical keyword extraction (n-gram extraction)
- Proposing a semantical topic clustering method
- Validating the performance of proposed methods on real-world English and Persian datasets.

In the following section, we provide a brief description of the word embeddings and text summarization methods. The proposed methods are described in Section III, followed by experimental results in Section IV. Final remarks and a discussion about our plans are reported in Section V.

## II. RELATED WORKS

### A. Word Embeddings

Word embedding is a term using to describe a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases are mapped to real-valued vectors. It involves a mathematical process where words from one dimension space embed into a higher dimensional continuous vector space, which is often tens to hundreds. A word embedding is a trained representation for text where words with similar semantic have a similar representation. In the word embedding process, each individual word is represented by a real-valued vector with dimensionally regard to trained vector space. Since, in these techniques, each word is demonstrated by a vector which is learned through a neural network training process, word embedding is often referred to as a deep learning method. The stepping stone idea to this method is using a densely distributed representation for each word. In these methods, each word represents by tens or hundreds of dimensions, contrary to sparse word representations, such as one-hot encoding which often has thousands or millions of dimensions. In Bengio

et al. (2003) Schwenk (2007) Mikolov et al. (2010) models each individual word is transformed into a real-valued vector using a pre-trained lookup table. The neural network language model can be used to obtain word representation. Which can further be utilized in other tasks, for example, Collobert and Weston (2008) Turian et al. (2010) embedding used for classification in NLP task, or using feed-forward networks Bengio et al. (2003) Gauvain et al. (2006) and then recurrent neural network models Mikolov et al. (2010) Mikolov et al. (2011) to predict the probability distribution of the next word. Many other methods have been proposed in order to create word embeddings that are based on the Distributed Hypothesis Harris (1954). Among these methods, word2vec Mikolov et al. (2013) and Glove Pennington et al. (2014) are the most popular methods with roughly the same accuracy. Word2vec consists of two models, continuous bag-of-words (CBOW) and skip-gram. These models learn a vector representation of each word by using a neural network language model and can be trained efficiently on the large size of the corpus. Word2vec allows models to learn the complex semantic relationship between words by using vector operations. For instance, Equation 1 represents that if we subtract the embedded vector of Italy from Rome (Subtract the country from its capital), then add France (another country), we get a vector approximately similar to the vector of Paris (capital of France). This example shows that the model learned the semantics of text and captured that the relationship between Italy and Rome is similar to the relationship between France and Paris.

$$\vec{v}(Rome) - \vec{v}(Italy) + \vec{v}(France) \approx \vec{v}(Paris) \quad (1)$$

In this work, we have adopted the implementation of Doc2vec from Gensim Python library. The goal of Doc2vec is to create a vector representation of a document, regardless of its length. But documents do not have logical structures such as words, so another method has to be found. While a word vector is trying to represent the concepts of the word, a document vector purpose is to represent the concept of a document. Mikolov and Le Le and Mikolov (2014) employed an easy and inventive concept. They used Word2vec model and added another vector (Paragraph ID) to it to overcome the length difficulty. It has two different models, PV-DM (Distributed Memory version of Paragraph Vector) and PV-DBOW (Distributed Bag of Words of Paragraph Vectors), the former is a type of extension to CBOW model in Word2vec, but instead of just using words to predict the next word, it uses another feature vector which is unique per document, as well. It plays a role as a memory that remembers what is missing from the current context. The later is more similar to the skip-gram model in Word2vec and uses a distributed bag of words. In Gensim implementation there is a variable (dm) which correlates to selecting one of these models. By default, it is 1 which is PV-DM and it is used in our proposed algorithm.

## B. Text Summarization

The first summarization method proposed by Luhn (1958), tried to weight the sentences of a document based on the

frequency of words and omitting the very high-frequency common words. Since then the research community has widely addressed automatic text summarization techniques. Very good surveys are available and have proposed by many researchers Sehgal et al. (2018) Pal and Saha (2014) Saranyamol and Sindhu (2014) Gambhir and Gupta (2017), but since we have exploited a graph-based method, we describe graph-based summarization and then explain the TextRank algorithm specifically which is the ground algorithm of our proposed method.

In graph-based methods, every sentence is assumed as a node of a graph and two sentences are connected with an edge if they possessed some words in common, or in other words, there is an edge between two sentences if with a similarity measure (such as cosine) their similarity were more than a threshold. In this regard, we can conclude two things from the generated graph. First, the isolated partitions in the graph (sub-graphs that are isolated from the rest of the sub-graphs) that embodies a distinct topic in the text. In query-specific summaries, some sentences from each sub-graphs may return as an answer to that specific question, while for common summaries, most important and representative sentences may be chosen from each of the sub-graphs, or they can be assumed as different topics in the original text and the summary for covering all the topics, should select sentences from all of the sub-graphs. The second inference can be obtained by graph-theoretic methods to find the most important and representative sentences (nodes) in the graph. This can be done by assuming more important sentences have a higher degree (more nodes are connected to them) thus, they have a higher probability to be included in the summary. By assuming documents as a graph which nodes are sentences and edges are the similarity between the nodes many different kinds of graph theory tools and measurement have been applied to find the important sentences for extractive summarization work Mihalcea (2005).

Now we want to explain the TextRank algorithm Mihalcea (2004) which is a very high-performance extractive summarization technique and its foundation is the PageRank algorithm. PageRank algorithm is commonly used in the Google search engine to compute the rank of web pages. This algorithm works with these main insights: Important pages are linked by important pages and the PageRank value of a page is essentially the probability of a user visiting that page. Scores in the PageRank algorithm calculated as follow in 2:

$$P_r(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{P_r(V_j)}{|Out(V_j)|} \quad (2)$$

In the TextRank, sentences are considered equivalent to web pages and apply the PageRank algorithm over sentences in the graph. Formally, let  $G = (V, E)$  be a directed graph with the set of vertices  $V$  and set of edges  $E$ , where  $E$  is a subset of  $V \times V$ . For a given vertex  $V_i$ , let  $In(V_i)$  be the set of vertices that point to it (predecessors), and let  $Out(V_i)$  be the set of vertices that vertex  $V_i$  points to (successors). The score of a vertex  $V_i$  is defined as Equation 3 in Page et al. (1999):

$$S(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \quad (3)$$

Where  $d$  is a damping factor and it is a number between 0 and 1 and has the role of compounding the probability of going from a specific vertex to another random vertex in the graph, into the model. In the web surfing concept, the "random surf model" is adopted in this graph-based ranking algorithm. In this model, when a user clicks on links at random with a probability of  $d$ , it goes to a new page with the probability  $1 - d$ . By starting from an arbitrary value assigned to each node in the graph, the computation iterates until it reaches a convergence point below a given threshold. After running the algorithm, each vertex is associated with a score, representing the "importance" of the vertex within the graph Mihalcea (2004).

### III. PROPOSED METHOD

We have proposed a graph-based extractive text summarization algorithm which is based on the TextRank algorithm. The essence of the TextRank algorithm is vertex voting, where the voting equals an edge between two vertexes. If a specific vertex has higher dependency and similarity to the rest of the vertexes, it would be mapped with a higher value. For illustrating this consider  $G = (V, E)$  as a graph with  $V$  being the vertex set and  $E$  the edge set. The importance metric of each vertex is as shown in Equation 4:

$$W(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{W_{ji}}{\sum_{V_k \in Out(v_j)} W_{jk}} W(V_j) \quad (4)$$

$In(V_i)$  is the set of indexes for vertexes (text units which are sentences here) and have a common window with  $V_i$  in linear order in a sentence,  $Out(V_j)$  is the set of vertexes that have a common window with  $V_i$ ,  $d$  is a damping factor, its default value is 0.85. Commonly, a weight assigned to an edge from  $V_j$  to  $V_i$  as  $w_{ji}$  and it is computed by calculating the chances of occurrence of two text units in the same text window with a fixed size, with the ordinary size of 2. In the initialization weight of all the text units are equal to one and all of the weights obtain consistency after some iteration through Equation (4). The text units (sentences) which have more weight are considered the key text units (sentences). Figure 1 demonstrates the flowchart of the algorithm.

In our work, we have used TextRank as a voting method to vote among doc2vec scores of each sentence. In other words, instead of using number of similar words in each sentence as similarity measure between sentences (nodes in graph) as it is in the TextRank, our algorithm scores the relationship between the sentences based on the score get from doc2vec trained model; hence, the semantics of words and sentences are included in the similarity rather than just considering the same words to measure the similarity. In the above flowchart, we made some alternations in the phase of computing the weight of the vertex. The novelty of our method is including semantic into the TextRank summarization and just not focus

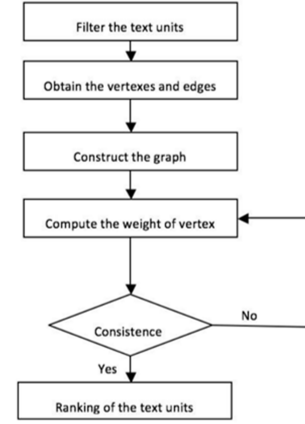


Fig. 1. TextRank Algorithm Flowchart

on a statistical analysis of similar exact words. Besides, it has a significant advantage, unsupervised learning, of the TextRank algorithm which makes it needless to huge corpus for training. This gives the algorithm the facility to be very convenient to utilize for summarizing new text, efficiently.

For implementing this algorithm we have used TextRank from Gensim, a Python library, and altered some part of it to become compatible with the Persian language to test on Persian corpus. We have trained two Doc2vec models, one on Hamshahri corpus AleAhmad et al. (2009) for Persian language usage, and another on Text8 corpus (which is a pre-processed a version of the first 100 million characters from Wikipedia dump) for English language purpose. Then we have added them to the algorithm to weigh the edges of the graph (similarity score between sentences). We could yield the most important sentences in the graph (nodes with a higher score) with this implementation. By assuming that the most important sentences contain more information about the text and have more relations with other nodes; we could return them as the summarization of the input text. In this implementation which is available at Github<sup>1</sup>, we can choose the number of words to be included in the summarization, as well as the ratio of input text we want to get as summarization, which is 20 percent of text as a default. To overcome the coverage problem in multi-topic texts we proposed a clustering method and apply the semantic text summarization described above. The proposed clustering algorithm weighs the similarity between paragraphs in a row based on a doc2vec model trained on separate documents. If the score was above a threshold (mean + standard deviation) they would merge in the same cluster; otherwise, they would not and we assume them as one cluster by itself. Then we apply the text summarization algorithm on each cluster independently. In this way, we can cover all topics of input text in our summarization.

#### A. topic clustering

In some scenarios, when the size of the text which want to summarize becomes large, the coverage issue arise. In the coverage issue, the text summarizer returns the most frequent

<sup>1</sup><https://github.com/minasmz/Persian-Summarization>

sentences as the summary. However, the text may contain other important sentences which might be neglected due to a lower frequency. To overcome this issue, we added a topic clustering method before applying the text summarization method on our text data. To do so, we apply the doc2vec method on the paragraphs of training set and obtained the representation of paragraphs of training set. Then, the distance between the paragraphs in training set is measured using the cosine similarity. Further, we calculate the mean ( $m$ ) and standard deviation of distance ( $std$ ) in the training set. To place two paragraphs in a same cluster of topic, we assumed they should have a distance less than the  $m + std$ , otherwise they belong to two different topics cluster. When we obtain the cluster of topics, we apply the text summarization method described earlier on each clusters. In this way, we resolve the coverage problem in text summarization.

### B. keyword extraction

After clustering the topics and summarizing the text, we tried to extract the keywords in the summaries. Two different methods are introduced to extract the n-grams as keywords in a text. In the first methods, the semantic of the words are not included in the algorithm and just the frequency of words using tf-idf method is used. In this method we have adopted the bm25 scoring function which regularly is used as a keyword extractor. We applied some alternation to the available bm25 algorithm in Gensim library of Python and made it compatible with Persian language. In the second method, the semantic of the words in the text is considered as well. To include the semantic of words, we trained a word2vec model and took a similar method introduced in text summarization to generate a graph of words. In this graph, each word is a node of the graph and the weight of edges are the cosine similarity between words representations connecting to each other. Then by applying the TextRank algorithm on the generated graph, we obtain the most important words. In this approach not only frequency of words are included in the obtaining the keywords, but also the meanings of them are included. In both methods, we tried to return n-gram (by default 10-gram) as the keyword. After acquiring most important and frequent words, we check them in unigram to n-gram of the words ( $n = 10$ ) if frequency of n-gram be more than half of the frequency of the word and its occurrence be more than 2 (in big size inputs this number should be increased) the important word occurred in the text would reduce to the bigger n-gram. Finally we return the most important n-gram where the important words are in them.

## IV. EXPERIMENTAL RESULTS

Although in text summarization there is no unique way to summarize a text and it is a difficult task even for the human to reach a concise form of a text, but researchers prepared some gold test set for evaluating the methods on them. We have used Zamanifar dataset for testing the algorithm on the Persian language and a BBC News Greene and Cunningham (2006) dataset which contains BBC news and their summaries for evaluating the work in the English language. We have calculated the accuracy by Rough measurement Equation 5. It

should be mentioned that doc2vec involves some randomness because of the negative sampling that is being used in its implementation, a different set of negative examples would be tried in each call so the result of the inferred vector from doc2vec model would be different in each call. To overcome this and measure the accuracy of the algorithm precisely, we have run the algorithm 10 times on each text and calculated the average of these 10 results. We compare the result of each two datasets separately in Persian and English languages to the pure TextRank algorithm. We have also set the ratio of summaries to 0.2, 0.5, and 0.8, respectively; and compare the results on each language which has been illustrated in Figure(2) and Figure(3).

$$Rough\_2 = \frac{\sum_{s \in \{RefSummaries\}} \sum_{bigrams_i \in S} \min(count(i, x), count(i, S))}{\sum_{s \in \{RefSummaries\}} \sum_{bigrams_i \in S} count(i, S)} \quad (5)$$

Results from the Persian language are illustrated in Figure(2). The best result is the highest rough score obtained from 10 times of running the algorithm on each text. And in the table you can see the average of the best score on 59 documents in Zamanifar dataset for the Persian language, the average score is averaged over 10 times running the algorithm, in the table averaging over average score in 59 document is brought and then it is compared to the result of the TextRank algorithm on 59 documents.

Figure(3), demonstrates the results of the algorithm on the English language, there is 417 document in the politic folder of BBC news dataset Greene and Cunningham (2006) and the Rough measurement calculated on them, so the results are averaging over 417 documents.

The average of 10 runs on each document on average has comparable and better accuracy than TextRank The best summary among 10 times running the algorithm has the highest average on all of the experiments. This makes us conclude that the result of combining semantic in graph-based text summarization, TextRank, would be helpful to increase the performance and accuracy of the summaries in TextRank algorithm. Moreover, we can conclude that by having a better model of each language, higher accuracy would be obtained.

Due to the lack of gold standard set for keyword extraction and topic clustering, we experimentally observed that our methods generate meaningful results.

## V. FUTURE WORKS AND DISCUSSION

Text summarization is a difficult task which even humans could not reach a unique summary from a text. For decades, researchers in extractive text summarization tried to select the most important sentences from the text which contain useful information by weighing and ranking the sentences based on some statistical assumptions and neglect the semantics of the words in sentences. In the proposed method we can observe from the results that including semantic in summaries can improve the accuracy. In this work, we have employed doc2vec

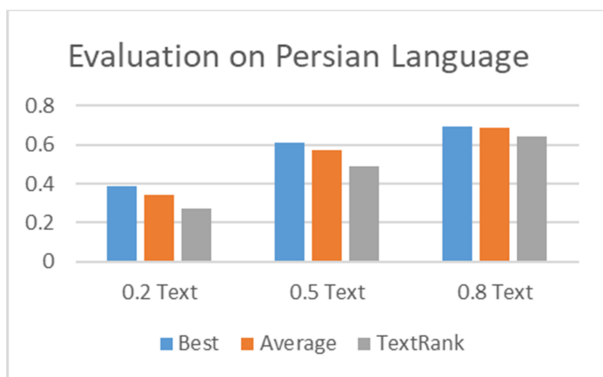


Fig. 2. Evaluation on Persian Language

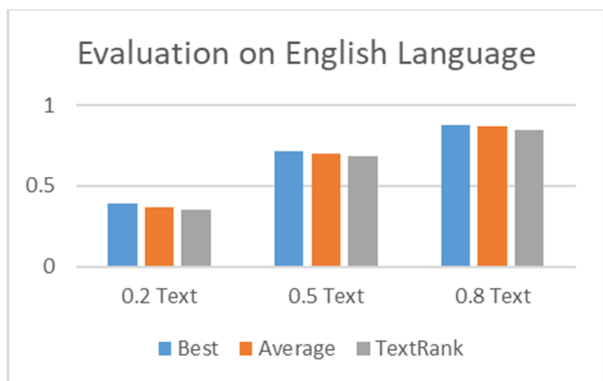


Fig. 3. Evaluation on English Language

models for learning semantic. For future work, we can say that by having better preprocessed and more data relating to the context of input texts, we can train more accurate models which leads to better results. hence, in future works, we can train these models on a bigger and cleaner dataset for reaching better accuracy. Also, we can train and adopt other word embeddings to include semantic in the statistical algorithms. Moreover, In the current work, we have used TextRank since it performs better among other graph-based algorithms in text summarization. Also, we can inject the semantic of texts into the other available graph-based statistical text summarization algorithms, such as hit and evaluate the results of that algorithms, too.

#### REFERENCES

AleAhmad, A., Amiri, H., Darrudi, E., Rahgozar, M., and Oroumchian, F. (2009). Hamshahri: A standard persian text collection. *Knowledge-Based Systems*, 22(5):382–387.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Gambhir, M. and Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Gauvain, L., Bengio, Y., and Schwenk, H. (2006). Neural probabilistic language models. *Innovations in Machine*.

Greene, D. and Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Harris, Z. (1954). Distributional structure. *word*, 10 (2-3): 146–162. reprinted in fodor, j. a and katz, jj (eds.), readings in the philosophy of language.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 170–173.

Mihalcea, R. (2005). Language independent extractive summarization. In *ACL*, volume 5, pages 49–52.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Pal, A. R. and Saha, D. (2014). An approach to automatic text summarization using wordnet. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 1169–1173. IEEE.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Saranyamol, C. and Sindhu, L. (2014). A survey on automatic text summarization. *Int. J. Comput. Sci. Inf. Technol*, 5(6):7889–7893.

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Sehgal, S., Kumar, B., Rampal, L., Chaliya, A., et al. (2018). A modification to graph based approach for extraction based automatic text summarization. In *Progress in Advanced Computing and Intelligent Engineering*, pages 373–378. Springer.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised

learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.