Graph Ranking on Maximal Frequent Sequences for Single Extractive Text Summarization

Yulia Ledeneva[1], René Arnulfo García-Hernández[1] and Alexander Gelbukh[2]

[1] Universidad Autónoma del Estado de México
Unidad Académica Profesional Tianguistenco
Instituto Literario #100, Col. Centro, Toluca, 50000, Estado de México
yledeneva@yahoo.com, renearnulfo@hotmail.com
[2] Centro de Investigación en Computación, Instituto Politécnico Nacional, 07738, DF, Mexico
www.Gelbukh.com

Abstract. We suggest a new method for the task of extractive text summarization using graph-based ranking algorithms. The main idea of this paper is to rank Maximal Frequent Sequences (MFS) in order to identify the most important information in a text. MFS are considered as nodes of a graph in term selection step, and then are ranked in term weighting step using a graph-based algorithm. We show that the proposed method produces results superior to the-state-of-the-art methods; in addition, the best sentences were found with this method. We prove that MFS are better than other terms. Moreover, we show that the longer is MFS, the better are the results. If the stop-words are excluded, we lose the sense of MFS, and the results are worse. Other important aspect of this method is that it does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, and languages.

1 Introduction

A summary of a document is a short text that communicates briefly the most important information from this document. The text summarization tasks can be classified into single-document and multi-document summarization. In single-document summarization, the summary of only one document is to be built, while in multi-document summarization the summary of a whole collection of documents is built. In this work, we have experimented only with single-document summaries, as a future work we apply this idea to multi-document summarization.

The text summarization methods can be classified into abstractive and extractive methods. An abstractive summary is an arbitrary text that describes the contexts of the source document. Abstractive summarization process consists of "understanding" the original text and "re-telling" it in fewer words. Namely, an abstractive summarization method uses linguistic methods to examine and interpret the text and then to find new concepts and expressions to best describe it by generating a new shorter text that conveys the most important information from the original document. While this may seem the best way to construct a summary (and this is how human beings do it), in

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

real-life setting immaturity of the corresponding linguistic technology for text analysis and generation currently renders such methods practically infeasible.

An extractive summary, in contrast, is a selection of text parts (phrases, sentences, paragraphs, etc.) from the original text, usually presented to the user in the same order—i.e., a copy of the source text with most text parts omitted. An extractive summarization method only decides, for each sentence, whether it should be included in the summary. The resulting summary reads rather awkward; however, simplicity of the underlying statistical techniques makes extractive summarization an attractive, robust, language-independent alternative to more "intelligent" abstractive methods. In this paper, we consider extractive summarization.

A typical extractive summarization method consists in several steps, at each of them different options can be chosen. We will assume that the text parts of selection are sentences. Therefore, final goal of the extractive summarization process is *sentence selection*. One of the ways to select the appropriate sentences is to assign some numerical measure of usefulness of a sentence for the summary and then select the best ones; the process of assigning these usefulness weights is called *sentence weighting*. One of the ways to estimate the usefulness of a sentence is to sum up usefulness weights of individual terms of which the sentence consists; the process of estimating the individual terms is called *term weighting*. For this, one should decide what the terms are: for example, they can be words; deciding what objects will count as terms is the task of *term selection*. Different extractive summarization methods can be characterized by how they perform these tasks [1].

There are a number of scenarios where automatic construction of such summaries is useful. For example, an information retrieval system could present an automatically built summary in its list of retrieval results, for the user to decide quickly which documents are interesting and worth opening for a closer look—this is what Google models to some degree with the snippets shown in its search results. Other examples include automatic construction of summaries of news articles or email messages to be sent to mobile devices as SMS; summarization of information for government officials, executives, researches, etc., and summarization of web pages to be shown on the screen of a mobile device, among many others.

The main proposal consists in detecting Maximal Frequent Sequences, and ranks them using a graph-based algorithm. The main contribution of this paper is the proposal of using MFS as nodes of a graph in term selection step, and the second contribution is using a graph-based algorithm in sentence weighting step.

The paper is organized as follows. Section 2 summarizes the state-of-the-art text summarization and graph-based methods. In Section 3, a graph-based algorithm is presented. Section 4 describes Maximal Frequent Sequences. In Section 5, a new method is presented. The experimental setting is described, and some conclusions are discussed in Section 6. Section 7 concludes the paper.

2 Graph-Based Algorithm

Recently, graph-based algorithms are applied successfully to different Natural Language Processing tasks. For example, a linear time graph-based soft clustering

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

algorithm was introduced for Word Sense Induction [2]. Given a graph, vertex pairs are assigned to the same cluster if either vertex has maximal affinity to the other. Clusters of varying size, shape, and density are found automatically making the algorithm suited to tasks such, where the number of classes is unknown and where class distributions may be skewed.

Other example of such applications consists of quantifying the limits and success of extractive summarization systems across domains [3]. The topic identification stage of single-document automatic text summarization across four different domains: newswire, literary, scientific and legal documents. The summary space of each domain is explored using an exhaustive search strategy, and finds the probability density function (pdf) of the ROUGE score distributions for each domain. Then this pdf is used to calculate the percentile rank of extractive summarization systems. The results introduce a new way to judge the success of automatic summarization systems and bring quantified explanations to questions such as why it was so hard for the systems to date to have a statistically significant improvement over the lead baseline in the news domain.

In [4], a hybrid graph-based method was presented annotating relationship maps with cross-document Structure Theory [5], and using network metrics [6]. It helped for Portuguese multi-document summarization.

Graph is a data structure that permits to model the meaning and structure of a cohesive text of many text-processing applications in a natural way. Particularly relevant in this paper is the application of random walks to text processing, as done in TextRank system [7]. TextRank has been successfully applied to three natural language processing tasks [8]: document summarization [3; 7], word sense disambiguation [9], and keyword extraction, and text classification [10] with results competitive with those of state-of-the-art methods. The strength of the model lies in the global representation of the context and its ability to model how the co-occurrence between features might propagate across the context and affect other distant features. The description of TextRank is given below.

2.1 Text Representation using Graphs

Graph representation. A text represented with a graph, interconnects words or other parts of a text with meaningful relations.

Depending on the application, nodes in the graph can be parts of a text of various sizes and characteristics. For example, words, ngrams, collocations, entire sentences, complete documents, etc. Note that the vertices can belong to different categories in the same graph.

To draw an edge between two vertices of a graph is done in a way of connection, which represent, for example, lexical or semantic relations, measures of text cohesiveness, contextual overlap, membership of a word in a sentence, etc.

Algorithm. After determining the type and characteristics of the elements added to the graph, the main algorithm of the ranking algorithms consists of the following steps [11]:

– Identify text units that best define the task, and add them as vertices in the graph.

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

– Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.

– Apply a graph-based ranking algorithm to find a ranking over the nodes in the graph. Iterate the graph-based ranking algorithm until convergence. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

2.2 Graph-Ranking Algorithms

The basic idea implemented by a random-walk algorithm is that of "voting" or "recommendation." When one vertex links to another one, it votes for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting a vote determines how important the vote itself is; this information is also taken into account by the ranking algorithm.

A random-walk algorithm called PageRank [Bri98] has been recently found successful in several text-processing applications such as text summarization and word sense disambiguation.

Given a directed graph $G = (V, E)$ with the set of vertices V and the set of edges E, where E is a subset of V x V. For a given vertex V_a, let In(V_a) be the set of vertices that point to it (predecessors), and let Out(V_a) be the set of vertices that vertex V_a points to (successors). The PageRank score associated with the vertex V_a is defined using a recursive function that integrates the scores of its predecessors:
We describe below two graph-based ranking algorithms:

$$S(V_a) = (1 - d) + d \times \sum_{V_b \in In(V_a)} \frac{S(V_b)}{|Out(V_b)|}, \quad (1)$$

where d is a parameter set between 0 and 1.

The score of each vertex is recalculated upon each iteration based on the new weights that the neighboring vertices have accumulated. The algorithm terminates when the convergence point is reached for all the vertices, meaning that the error rate for each vertex falls below a pre-defined threshold.

This vertex-scoring scheme is based on a random-walk model, where a walker takes random steps on the graph, with the walk being modelled as a Markov process. Under certain conditions (namely, that the graph should be aperiodic and irreducible), the model is guaranteed to converge to a stationary distribution of probabilities associated with the vertices in the graph. Intuitively, the stationary probability associated with a vertex represents the probability of finding the walker at that vertex during the random walk, and thus it represents the importance of the vertex within the graph.

PageRank [Bri98] is perhaps one of the most popular ranking algorithms, which was designed as a method for Web link analysis. Unlike other ranking algorithms,

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

PageRank integrates the impact of both incoming and outgoing links into one single model, and therefore it produces only one set of scores:

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

In matrix notation, the PageRank vector of stationary probabilities is the principal eigenvector for the matrix A_{row}, which is obtained from the adjacency matrix A representing the graph, with all rows normalized to sum to 1: $P = A^T_{row}P$.

A ranking process starts by assigning arbitrary values to each node in the graph, followed by several iterations until convergence below a given threshold is achieved. Convergence is achieved when the error rate for any vertex in the graph falls below a given threshold, where the error rate of a vertex V_i is approximated with the difference between the scores computed at two successive iterations: $S^{k+1}(V_i) - S^k(V_i)$ (usually after 25-35 iteration steps). After running the algorithm, a score is associated with each vertex, which represents the "importance" (*rank*) of the vertex within the graph. Note that for such iterative algorithms, the final value obtained for each vertex is not affected by the choice of the initial value; only the number of iterations to convergence may be different.

Undirected Graphs: Although traditionally applied on directed graphs, algorithms for node activation or ranking can be also applied to undirected graphs. In such graphs, convergence is usually achieved after a larger number of iterations, and the final ranking can differ significantly compared to the ranking obtained on directed graphs.

Weighted Graphs: When the graphs are built from natural language texts, they may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices V_i and V_j as a weight w_{ij} added to the corresponding edge that connects the two vertices. Consequently, we introduce new formulae for graph-based ranking that take into account edge weights when computing the score associated with a vertex in the graph, e.g.

$$PR^W(V_i) = (1 - d) + d \times \sum_{V_j \in In(V_i)} w_{ij} \frac{PR^W(V_j)}{\sum_{V_k \in Out(V_j)} w_{jk}} \quad (3)$$

3 Maximal Frequent Sequences

An ngram is a sequence of *n* words. We say that an ngram occurs in a text if these words appear in the text in the same order immediately one after another. For example, a 4-gram (ngram of length 4) *words appear in the text* occurs once in the

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

previous sentence, while *appear immediately after another* does not (these words do not appear on adjusting positions), neither does *the text appear in* (order is different).

The definition of ngram depends on what one considers words. For example, one can consider capitalized (*Mr. Smith*) and non-capitalized (*a smith*) words as the same word or as different words; one can consider words with the same morphological stem (*ask*, *asked*, *asking*), the same root (*derive*, *derivation*), or the same meaning (*occur*, *appear*) as the same word; one can omit the stop-words (*the*, *in*) when counting word positions, etc. Say, one can consider that in our example sentence above there occur the ngrams *we say* (capitalization ignored), *word appear* (plural ignored), *appear text* (*in the* ignored). This can affect counting the ngrams: if one considers *occur* and *appear* as equivalent and ignores the stop-words, then in our example sentence the bigram *appear text* occurs twice.

We call an ngram frequent (more accurately, β-frequent) if it occurs more than β times in the text, where β is a predefined threshold. Frequent ngrams—we will also call them frequent sequences (FSs)—often bear important semantic meaning: they can be multiword expressions (named entities: *The United States of America*, idioms: *kick the basket*) or otherwise refer to some idea important for the text (*the President's speech*, *to protest against the war*).

Our hypothesis is that FSs can express ideas both important and specific for the document. This can be argued in terms of *tf-idf* (term frequency—inverse document frequency, a notion well known in information retrieval [12]). On the one hand, the idea expressed by an FS is important for the document if it repeatedly returns to it (high term frequency). On the other hand, the corresponding idea should be specific for this document, otherwise there would exist in the language a single word or at least an abbreviation to express it (high inverse document frequency). It is important to note that this argument does not apply to 1-grams, i.e., single words. Therefore, we do not consider 1-grams as ngrams in the rest of this paper.

An ngram can be a part of another, longer ngram. All ngrams contained in an FS are also FSs. However, with the arguments given above one can derive that such smaller ngrams may not bear any important meaning by their own: e.g., *The United States of America* is a compound named entity, while *The United* or *States of America* are not. Exceptions like *The United States* should not affect much our reasoning since they tend to be synonymous to the longer expression, and the author of the document would choose one or another way to refer to the entity, so they should not appear frequently both in the same document.

FSs that are not parts of any other FS are called Maximal Frequent Sequences (MFSs) [13, 14]. For example, in the following text

... Mona Lisa is the most beautiful picture of Leonardo da Vinci ...
... Eiffel tower is the most beautiful tower ...
... St. Petersburg is the most beautiful city of Russia ...
... The most beautiful church is not located in Europe ...

the only MFS with $\beta = 3$ is *is the most beautiful*, while the only MFS $\beta = 4$ is *the most beautiful* (it is not an MFS with $\beta = 3$ since it is not maximal with this β). As this example shows, the sets of MFSs with different thresholds do not have to, say, contain one another.

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

One of our hypotheses was that only MFSs should be considered as bearing important meaning, while non-maximal FSs (those that are parts of another FS) should not be considered. Our additional motivation was cost vs. benefit considerations: there are too many non-maximal FSs while their probability to bear important meaning is lower. In any case, MFSs represent all FSs in a compact way: all FSs can be obtained from all MFSs by bursting each MFS into a set of all its subsequences. García [13] proposed an efficient algorithm to find all MFSs in a text, which we also used to efficiently obtain and store all FSs of the document.

The notions of FSs and MFSs are closely related to that of repeating bigrams; see Section 5. This set is conceptually simpler, but for computational implementation, MFSs could be more compact.

4 Proposed Method

In this section, the proposed method is presented.

TextRank. We use a graph-based ranking algorithm *TextRank* to find a ranking over the nodes in the graph. Iterate the graph-based ranking algorithm until convergence. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

- *Vertices*. We propose to use MFSs as vertices of a graph (see Section 3).
- *Nodes*. Relations that connect MFSs are term weighting relations such as (1) frequency of MFSs in a text: *f*, (2) length of MFS: l, and its presence as 1 or its absence as 0 (see Section 3).
- *Configuration of algorithm (TextRank)*: for this task, the goal is to rank MFSs, and therefore a *vertex* is added to the graph for each MFS in the text. To draw *nodes* between vertices, we are defining a term weighting relation, where "term weighting" can be defined in various ways. In the experiments realized in this paper, we use a term weighting described in below (in term weighting step). Such a relation between two sentences can be seen as a process of recommendation: a sentence that addresses certain concepts in a text, gives the reader a recommendation to refer to other sentences in the text that address the same or similar concepts. The resulting graph is highly connected, with a weight associated with each edge, and thus we use again the weighted version of the graph algorithms.

Term selection. We experiment with MFSs and other term selection options derived from them. Namely, we considered the following variants of term selection**:**

- *M*: the set of all MFSs, i.e., an ngram $m \in M$ if it is an MFS with some threshold β (recall that MFSs are of 2 words or longer and $\beta \geq 2$).[1] In the example from

[1] In practice, we only considered the MFSs with the thresholds $\beta = 2$, 3, and 4, since MFSs with higher thresholds were very rare in our collection, except for those generated by stop-words.

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

Section 3, M = {*is the most beautiful*, *the most beautiful*}. Also, we denote by M_2 the set of all MFSs with $\beta = 2$.

– *W*: single words (unigrams) from elements of *M*. Namely, a word $w \in W$ iff there exists an MFS $m \in M$ such that $w \in m$.
In our example, W = {*is*, *the*, *most*, *beautiful*}.

The set *W* are naturally derived from the notion of MFS and at the same time can be efficiently calculated.

Optionally, stop-words were eliminated at the pre-processing stage; in this case our MFSs could span more words in the original text, as explained in Section 4.

Term weighting. Different formulae were considered containing the following values:

– *f*: frequency of the term in MFSs, i.e., the number of times the term occurs in the text within some MFS. In our example, *f*(*is*) = 3 since it occurs 3 times in the text within the MFS *is the most beautiful*. If the term itself is an MFS, then this is just the frequency of this term in the text (e.g., for *M*, *f* is the same as term weight in Section 5; for *W* and *N* it is not). Under certain realistic conditions (MFSs do not intersect in the text, words do not repeat within one MFS) *f* is the number of times the term occurs in the text as part of a repeating bigram. In our example, *f*(*is*) = 3 since it occurs 3 times in a repeating bigram *is the* (and one time in a non-repeating context *church is not*).

– *l*: the maximum length of a MFS containing the term. In our example, *l*(*is*) = 4 since it is contained in a 4-word MFS *is the most beautiful*.

– 1: the same weight for all terms.

Sentence weighting. The sum of the weights of the terms contained in the sentence was used. For sentence selection, the following options were considered:

– best: sentences with greater weight were selected until the desired size of the summary (100 words) is reached. This is the most standard method.

– *k*best+first: *k* best sentences were selected, and then the first sentences of the text were selected until the desired size of the summary was reached. This was motivated by a hard-to-beat baseline mentioned in Section 2: only very best sentences according to our weighting scheme could prove to be above this baseline.

5 Experimental Setting and Results

Main algorithm. We have conducted several experiments to verify our hypotheses formulated in the previous section. In each experiment, we followed the standard sequence of steps:

– *TextRank algorithm*: we use undirected version of PageRank;
– *Term selection*: decide which features are to be used to describe the sentences;
– *Term weighting*: decide how the importance of each feature is to be calculated;

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

– *Sentence weighting*: decide how the importance of the features is to be combined into the importance measure of the sentence;
– *Sentence selection*: decide which sentences are selected for the summary.

Test data set. We used the DUC collection provided [15]. In particular, we used the data set of 567 news articles of different length and with different topics. Each document in the DUC collection is supplied with a set of human-generated summaries provided by two different experts. While each expert was asked to generate summaries of different length, we used only the 100-word variants.

Evaluation procedure. We used the ROUGE evaluation toolkit [16], which was found to correlate highly with human judgments [17]. It compares the summaries generated by the program with the human-generated (gold standard) summaries. For comparison, it uses *n*-gram statistics. Our evaluation was done using *n*-gram (1, 1) setting of ROUGE, which was found to have the highest correlation with human judgments, namely, at a confidence level of 95%.

Experiment 1. We conducted this experiment in two phases: first, we tried sentence weighting using option *best* and then option *k*best+first. In each experiment, we test two term selection options *M* and *W*. We excluded stop-words. The results are shown in Table 1.

Table 1. Results for different term selection, term weighting and sentence selection options using the proposed method (options: *M*, *W*, *excluded, best,* and *kbest+first*).

Term Selection		Term weighting	Sentence Weighting	Results		
Terms	Stop-words			Recall	Precision	F-measure
W	*excluded*	f	1*best+first*	47.603	47.518	47.543
			2*best+first*	47.718	47.621	47.652
M	*excluded*	l	1*best+first*	47.783	47.699	47.724
			2*best+first*	48.212	48.088	48.132
		f	1*best+first*	47.797	47.712	47.737
			2*best+first*	**48.211**	**48.093**	**48.134**
M	excluded	1	*best*	46.668	48.337	47.474
		f		48.009	47.757	47.865
		f^2		48.056	47.801	47.910
		l		48.025	47.773	47.881
		l^2		48.058	47.812	47.917
		$f \times l$		48.060	47.810	47.916
		f^l		48.079	47.831	47.937
W	excluded	1	*best*	47.682	47.604	47.626
		f		**48.659**	**48.324**	**48.473**
		f^2		48.705	48.235	48.451

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

We started our experiment from modifying term weighting parameters for the term selection scheme *W* with the term weighting option *f*, which showed good performance in the first experiment; and then we tried the term selection options *M* with the term weighting option 1 and the option *f*. We use *kbest+first* option for the sentence-weighting, see the first part of Table 1. The last line of the first part of the table represents the best result from the first part of Table 1. The best results are highlighted in boldface.

In the second part of Table 2, we change the sentence selection option using *best* option. In addition, we tried more term weighting option related to *f*. First, we tried the term selection scheme *M*, because of the better results obtained from the first part of the table, and then we tried the term selection *W*.

Term selection *W* gave a better result than *M*. Finally, with the best combinations obtained from the first two experiments, are highlighted in boldface and underlined.

Experiment 2. In the second part of experiments (see Table 2), we included stop-words and tried sentence weighting using *best* and *k*best+first options.

Table 2. Results for different term selection, term weighting and sentence selection options using the proposed method (options: *M*, *W*, *included, best,* and *kbest+first*).

Term Selection	Term weighting	Sentence Weighting	Results		
			Recall	Precision	F-measure
W	*f*	1*best+first*	47.694	47.612	47.635
		2*best+first*	47.870	47.761	47.798
M	*l*	1*best+first*	47.711	47.623	47.650
		2*best+first*	48.064	47.923	47.976
	f	1*best+first*	47.738	47.649	47.676
		2*best+first*	**48.148**	**48.016**	**48.065**
M	1	*best*	47.484	49.180	48.283
	f		48.803	48.533	48.626
	f^2		48.746	48.482	48.572
	l		**48.823**	**48.577**	**48.658**
	l^2		48.741	48.518	48.587
	$f \times l$		48.796	48.529	48.620
	f^l		48.716	48.497	48.564
W	1	*best*	47.529	47.483	47.489
	f		**48.821**	**48.424**	**48.604**
	f^2		48.784	48.322	47.489

Taking into account the best combination obtained from the first experiment, we tried different sentence selection variants including stop-words; see Table 2. From Table 1, we knew that term selection scheme *M* with stop-words removed, gave the best results with other parameters fixed (term weighting, sentence weighting, and sentence selection); see the first part of Table 1. Therefore, we started from modifying

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

these parameters for the same term selection scheme; see the first part of Table 2. The last line of the first part of the table represents the best result from the upper first part of Table 1. The best results are highlighted in boldface.

Then we tried the term selection option *W* with the term weighting option 1 and the options related to *f*, which showed good performance in the first experiment. The results are shown in the first below part of Table 2. Term selection *M* gave a better result than *W*. The option for term weighting *l* represents the best result from Table 2. The best results are highlighted in boldface and underlined.

Conclusion 1. In this conclusion, we discuss the best term selection option. We show in Table 3 the comparison of the best results for the term selection options of *M* and *W* from Table 2. The best result was obtained with MFSs (option *M*): it means that the proposed method ranks better on MFSs (option *M*) than words derived from MFSs (option *W*).

In [1] was shown that *W* are better than *M* for single text extractive summarization using options *W*, *f*, *best* and *W*, *f*, 1*best*+*first*. Thus, we conclude that the proposed method benefits MFSs (option *M*).

Table 3. Comparison for different term selection, term weighting and sentence selection options using the proposed method (options: *M*, *W*, *included, best,* and *kbest+first*). Comparison of the best results using the term selection option of *M* and *W* from the Table 2.

Term Selection	Term Weighting	Sentence Weighting	Results		
			Recall	Precision	F-measure
M	*f*	2*best*+*first*	**48.148**	**48.016**	**48.065**
W	*f*	2*best*+*first*	47.870	47.761	47.798
M	*l*	*best*	**48.823**	**48.577**	**48.658**
W	*f*	*best*	48.821	48.424	48.604

Conclusion 2. In this conclusion, we discuss the best term weighting options from Table 2. We take the best results from Table 2, and compose Table 4. The best result was obtained with the term weighting option *l*: length of the corresponding MFSs (option *M*); it means that the longer MFSs the better for single text summarization. In addition, it means that the proposed method benefits the length of MFS.

Table 4. Comparison for different term selection, term weighting and sentence selection options using the proposed method (options: *M*, *W*, *included, best,* and *kbest+first*). Comparison of the term weighting options *l* and *f*.

Term Selection	Term Weighting	Sentence Weighting	Results		
			Recall	Precision	F-measure
M	***L***	*best*	**48.823**	**48.577**	**48.658**
W	*F*	*best*	48.821	48.424	48.604
M	*F*	2*best*+*first*	48.148	48.016	48.065
W	*F*	2*best*+*first*	47.870	47.761	47.798

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

Conclusion 3. In this conclusion, we discuss the state-of-the-art methods that use different pre-processing options, see Table 5. The results are getting better, if for options with the term selection option *M*, stop-words are included. Contrary to the option *W*, the results are getting better, if stop-words are excluded. For the reason that the stop-words do not try any sense, when we eliminate stop-words, the results are getting better because the words with meaning are kept.

The option *M* considers MFSs (multiword expressions [20]), so when we exclude stop-words from multiword expressions, the sense is lost and the results are worsened.

Table 5. Comparison of the state-of-the-art methods that use different pre-processing options (sentence selection options: *best* and *kfirst*+*best*).

Method	Term Selection		Term weight-ing	Results		
	Terms	Stop-words		Recall	Precision	F-measure
Related work [1]	*W*	Included	*f*	46.523	48.219	47.344
Related work [1]	*W*	Excluded	*f*	46.576	48.278	47.399
Related work [1]	*W*	Excluded	*f*	46.536	48.230	47.355
Related work [1]	*W*	Excluded	*f*	46.622	48.407	47.486
Related work [1]	*W*	Excluded	*f*	46.788	48.537	47.634
Pre-processing [21]	*M*	Excluded	*1*	46.266	47.979	47.094
Pre-processing [21]	*M*	excluded stemming	*1*	46.456	48.169	47.285
Pre-processing [21]	*M*	included stemming	*1*	46.508	48.233	47.343
Proposed	*W*	Excluded	*f*	**48.659**	**48.324**	**48.473**
Proposed	*M*	Included	*1*	**48.823**	**48.577**	**48.658**

Conclusion 4. One can observe from [1] that any *k*best+first sentence selection option not outperformed any combination that used the standard sentence selection scheme, with bigger *k* always giving better results—that is, only the slightest correction to the baseline deteriorate it. See the comparison of MFS1 and MFS2 [1]. For the proposed method, the result with the option *kbest*+*first* is better than with the option *best*.

It is very important result because no one state-of-the-art-method could beat baseline configuration described in Section 5: only the very best sentences according to our weighting scheme might prove to be above this baseline. Therefore, the proposed method could beat this baseline configuration. The proposed method finds better sentences than baseline (baseline for the configuration of news article was not

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

possible to improve until now because the structure of news article where the first sentences are the most important). Observe in Table 6 that the result with options **Proposed**: *M, l, best* is better than **Proposed**: *M, l,* 1*best+first* and **Proposed**: *M, l,* 2*best+first*, contrary to *W, f, best* not outperformed *W, f,* 1*best+first* and *W, f,* 2*best+first.*

See in Table 1 and 2 details for more comparison. For example, *W, f, best* also are better than *W, f, kbest+first.*

Table 6. Comparison of methods to show the difference between *best* and *kbest+first* options.

Comparison	Method	Recall	Precision	F-measure
Related work [1]	*W, f, best*	44.609	45.953	45.259
	W, f, 1best+first	46.576	48.278	47.399
	W, f, 2best+first	46.158	47.682	46.895
Proposed	*M, l, best*	48.823	48.577	48.658
	M, f, 1best+first	47.711	47.623	47.650
	M, f, 2best+first	48.064	47.923	47.976

Conclusion 5. The comparison to the state-of-the-art methods is given in Table 7. We group methods depending on additional information were used for that methods: none, order of sentences, pre-processing, clustering. Even though the proposed method does not use any additional information, outperforms the other methods.

The best overall result is for the proposed method that does not use the additional information for generating text extractive summaries. It means that this method is completely independent (domain independent and text position independent). In addition, this method is language independent because does not use any lexical information, and can be used for different languages.

Table 7. Comparison of the results with other state-of-the-art methods.

Additional info used	Method	Recall	Precision	F- measure
None	Baseline: *random*	37.892	39.816	38.817
	TextRank: [7]	45.220	43.487	44.320
	MFS: [19]	44.609	45.953	45.259
	Proposed: *M, l, best*	**48.823**	**48.577**	**48.658**
Order of sentences	Baseline: *first*	46.407	48.240	47.294
	MFS: [19]	46.576	48.278	47.399
	Proposed: *M, f, 2best+first* (without pre-processing)	**48.148**	**48.016**	**48.065**
Pre-processing	TextRank: [A]	46.582	48.382	47.450
	TextRank: [3]	47.207	48.990	48.068
	Proposed: *W, f, best*	**48.659**	**48.324**	**48.473**
Clustering	MFS (*k*-best) [19]	47.820	47.340	47.570
	MFS (EM-5) [18]	47.545	48.075	47.742

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

Table 8 shows the results according to their relevance. Considering that Topline is the best obtained result and Baseline: *random* as the worse obtained result.

Table 8. Comparison of the results with other state-of-the-art methods (F-measure and significance).

Method	F- measure	Significance
Baseline: *random*	38.817	0%
TextRank: [7]	44.320	26.47%
MFS: [19]	45.259	31.28%
Baseline: *first*	47.294	40.78%
MFS: [19]	47.399	41.29%
TextRank: [A]	47.450	41.53%
MFS (*k*-best) [19]	47.570	42.11%
MFS (EM-5) [18]	47.742	42.94%
Proposed: *M, f, 2best+first* (without pre-processing)	**48.065**	**44.47%**
TextRank: [3]	48.068	46.20%
Proposed: *M, l, best*	**48.658**	**47.35%**
Topline [20]	59.600	100%

6 Future Work

As a future work, this method can be applied on different Natural Language Processing Tasks such as Word Sense Disambiguation, Text Classification, Collocation Extraction, and others. We believe that in the future the proposed method can improve various state-of-the-art methods and contribute with even better results.

In particular, we plan to extend the notion of MFS to that of syntactic n-gram [22, 23], and extend our method to multi-document summarization, especially in the context of social networks [24, 25].

References

1. Yulia Ledeneva, Alexander Gelbukh, René García Hernández. Terms Derived from Frequent Sequences for Extractive Text Summarization. LNCS 4919, pp. 593-604, Springer, 2008.
2. David Hope and Bill Keller. MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction. CICLing 2013, Part I, Springer LNCS 7816, pp. 368-381, 2013.
3. Hakan Ceylan, Rada Mihalcea, Umut Ozertem, Elena Lloret and Manuel Palomar, Quantifying the Limits and Success of Extractive Summarization Systems Across Domains, Proc. of the North American Chapter of the ACL (NACLO 2010), Los Angeles, 2010.
4. Ribaldo, R.; Akabane, A.T.; Rino, L.H.M.; Pardo, T.A.S. Graph-based Methods for Multi-document Summarization: Exploring Relationship Maps, Complex Networks and Discourse Information. In the Proceedings of the 10th International Conference on Computational Processing of Portuguese (LNAI 7243), pp. 260-271. Portugal. 2012.

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.

5. Maziero, E.G. and Pardo, T.A.S. Automatic Identification of Multi-document Relations. In the (on-line) Proceedings of the PROPOR 2012 PhD and MSc/MA Dissertation Contest, pp. 1-8. April 17-20, Coimbra, Portugal. 2012.
6. Antiqueira, L., Oliveira Jr., O.N., Costa, L.F., Nunes, M.G.V.: A Complex Network Approach to Text Summarization. Information Sciences 179 (5), 584-599. 2009.
7. Mihalcea, R. Random Walks on Text Structures. CICLing 2006, LNCS, vol. 3878, pp. 249–262, Springer-Verlag 2006.
8. Rada Mihalcea and Dragomir Radev, Graph-based Natural Language Processing and Information Retrieval , Cambridge University Press, 2011.
9. Ravi Sinha and Rada Mihalcea, Unsupervised Graph-based Word Sense Disambiguation, book chapter in "Current Issues in Linguistic Theory: Recent Advances in Natural Language Processing", N. Nicolov, R. Mitkov (Eds.), John Benjamins Publishers, 2009.
10. Hassan, S., Mihalcea, R., Banea, C. Random-Walk Term Weighting for Improved Text Classification. IEEE International Conference on Semantic Computing (ICSC 2007), Irvine, CA, 2007.
11. Mihalcea, R., Tarau, P. TextRank: Bringing Order into Texts, in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain, 2004.
12. Salton G., Buckley, C. Term-weighting approaches in automatic text retrieval. Information processing & Management, 24:513–523, 1988.
13. García-Hernández, R.A., Martínez-Trinidad J. F., Carrasco-Ochoa J. A. A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text, CIARP'2004, LNCS vol. 3287 Springer-Verlag 2004. pp. 478–486.
14. García-Hernández, R.A., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A. A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection, CICLing-2006, LNCS vol. 3878 Springer-Verlag 2006. pp. 514–523.
15. DUC. Document understanding conference 2002; www-nlpir.nist.gov/projects/duc.
16. Lin C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of Workshop on Text Summarization of ACL, Spain, 2004.
17. Lin C.Y., Hovy E. Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics. In Proceedings of HLT-NAACL, Canada, 2003.
18. Yulia Ledeneva, René García Hernández, Romyna Montiel Soto, Rafael Cruz Reyes, and Alexander Gelbukh. EM Clustering Algorithm for Automatic Text Summarization. LNAI 7094, Part I, pp. 305–315, Springer, 2011.
19. Romyna Montiel Soto, René García Hernández, Yulia Ledeneva, Rafael Cruz Reyes. Comparación de Tres Modelos de Representación de Texto en la Generación Automática de Resúmenes. Procesamiento del Lenguaje Natural, vol. 43, pp. 303-311, 2009.
20. Yulia Ledeneva, PhD. Thesis: Automatic Language-Independent Detection of Multiword Descriptions for Text Summarization, Mexico: National Polytechnic Institute, 2008.
21. Yulia Ledeneva. Effect of Preprocessing on Extractive Summarization with Maximal Frequent Sequences. LNAI 5317, pp. 123-132, Springer-Verlag, ISSN 0302-9743, 2008.
22. Grigori Sidorov. Syntactic Dependency Based N-grams in Rule Based Automatic English as Second Language Grammar Correction. International Journal of Computational Linguistics and Applications 4(2), 2013, pp. 169–188.
23. Grigori Sidorov. Non-continuous Syntactic N-grams. Polibits, vol. 48, pp. 67–75, 2013.
24. Nibir Nayan Bora. Summarizing Public Opinions in Tweets. International Journal of Computational Linguistics and Applications 3(1), 2012, pp. 41–55.
25. Alexandra Balahur, Mijail Kabadjov, Josef Steinberger. Exploiting Higher-level Semantic Information for the Opinion-oriented Summarization of Blogs. International Journal of Computational Linguistics and Applications 1(1–2), 2010, pp. 45–59.

15th International Conference on Intelligent Text Processing and Computational Linguistics, CICLing 2014, Part II. Lecture Notes in Artificial Intelligence, N 8404, 2013, pp. 466–480. See http://link.springer.com/chapter/10.1007/978-3-642-54903-8_39.