Automatic Text Summarization using a Machine Learning Approach

Joel Larocca Neto Alex A. Freitas Celso A. A. Kaestner

Pontifical Catholic University of Parana (PUCPR)
Rua Imaculada Conceicao, 1155
Curitiba - PR. 80.215-901. BRAZIL
{joel, alex, kaestner}@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex

Abstract. In this paper we address the automatic summarization task. Recent research works on extractive-summary generation employ some heuristics, but few works indicate how to select the relevant features. We will present a summarization procedure based on the application of trainable Machine Learning algorithms which employs a set of features extracted directly from the original text. These features are of two kinds: statistical – based on the frequency of some elements in the text; and linguistic – extracted from a simplified argumentative structure of the text. We also present some computational results obtained with the application of our summarizer to some well known text databases, and we compare these results to some baseline summarization procedures.

1 Introduction

Automatic text processing is a research field that is currently extremely active. One important task in this field is *automatic summarization*, which consists of reducing the size of a text while preserving its information content [9], [21]. A *summarizer* is a system that produces a condensed representation of its input's for user consumption [12].

Summary construction is, in general, a complex task which ideally would involve deep natural language processing capacities [15]. In order to simplify the problem, current research is focused on extractive-summary generation [21]. An extractive summary is simply a subset of the sentences of the original text. These summaries do not guarantee a good narrative coherence, but they can conveniently represent an approximate content of the text for relevance judgement.

A summary can be employed in an *indicative* way – as a pointer to some parts of the original document, or in an *informative* way – to cover all relevant information of the text [12]. In both cases the most important advantage of using a summary is its reduced reading time. Summary generation by an automatic procedure has also other advantages: (i) the size of the summary can be controlled; (ii) its content is determinist; and (iii) the link between a text element in the summary and its position in the original text can be easily established.

In our work we deal with an automatic trainable summarization procedure based on the application of machine learning techniques. Projects involving extractive summary generation have shown that the success of this task depends strongly on the use of heuristics [5], [7]; unfortunately few indicatives are given of how to choose the relevant features for this task. We will employ here statistical and linguistic features, extracted directly and automatically from the original text.

The rest of the paper is organized as follows: section 2 presents a brief review of the text summarization task; in section 3 we describe in detail our proposal, discussing the employed set of features and the general framework of trainable summarizer; in section 4 we relate the computational results obtained with the application of our proposal to a reference document collection; and finally, in section 5 we present some conclusions and outline some envisaged research work.

2 A review of text summarization

An automatic summarization process can be divided into three steps [21]: (1) in the *preprocessing step* a structured representation of the original text is obtained; (2) in the *processing step* an algorithm must transform the text structure into a summary structure; and (3) in the *generation step* the final summary is obtained from the summary structure.

The methods of summarization can be classified, in terms of the level in the linguistic space, in two broad groups [12]: (a) shallow approaches, which are restricted to the syntactic level of representation and try to extract salient parts of the text in a convenient way; and (b) deeper approaches, which assume a semantics level of representation of the original text and involve linguistic processing at some level.

In the first approach the aim of the preprocessing step is to reduce the dimensionality of the representation space, and it normally includes: (i) *stop-word* elimination – common words with no semantics and which do not aggregate relevant information to the task (e.g., "the", "a") are eliminated; (ii) *case folding*: consists of converting all the characters to the same kind of letter case - either upper case or lower case; (iii) *stemming*: syntactically-similar words, such as plurals, verbal variations, etc. are considered similar; the purpose of this procedure is to obtain the stem or radix of each word, which emphasize its semantics.

A frequently employed text model is the vectorial model [20]. After the preprocessing step each text element – a sentence in the case of text summarization – is considered as a N-dimensional vector. So it is possible to use some metric in this space to measure similarity between text elements. The most employed metric is the cosine measure, defined as $\cos\theta = (<x.y>) / (|x| \,.\, |y|)$ for vectors x and y, where $(<,>)$ indicates the scalar product, and $|x|$ indicates the module of x. Therefore maximum similarity corresponds to $\cos\theta = 1$, whereas $\cos\theta = 0$ indicates total discrepancy between the text elements.

The evaluation of the quality of a generated summary is a key point in summarization research. A detailed evaluation of summarizers was made at the TIPSTER Text Summarization Evaluation Conference (SUMMAC) [10], as part of an effort to standardize summarization test procedures. In this case a reference summary collection was provided by human judges, allowing a direct comparison of the performance of the systems that participated in the conference. The human effort to elaborate such summaries, however, is huge. Another reported problem is that even in the case of human judges, there is low concordance: only 46 % according to Mitra [15]; and more importantly: the summaries produced by the same human judge in different dates have an agreement of only 55 % [19].

The idea of a "reference summary" is important, because if we consider its existence we can objectively evaluate the performance of automatic summary generation procedures using the classical Information Retrieval (IR) *precision* and *recall* measures. In this case a sentence will be called *correct* if it belongs to the reference summary. As usual, *precision* is the ratio of the number of selected correct sentences over the total number of selected sentences, and *recall* is the ratio of the number of selected correct sentences over the total number of correct sentences. In the case of fixed-length summaries the two measures are identical, since the sizes of the reference and the automatically obtained extractive summaries are identical.

Mani and Bloedorn [11] proposed an automatic procedure to generate reference summaries: if each original text contains an author-provided summary, the corresponding size-K reference extractive summary consists of the K most similar sentences to the author-provided summary, according to the cosine measure. Using this approach it is easy to obtain reference summaries, even for big document collections.

A Machine Learning (ML) approach can be envisaged if we have a collection of documents and their corresponding reference extractive summaries. A trainable summarizer can be obtained by the application of a classical (trainable) machine learning algorithm in the collection of documents and its summaries. In this case the sentences of each document are modeled as vectors of features extracted from the text. The summarization task can be seen as a two-class classification problem, where a sentence is labeled as "correct" if it belongs to the extractive reference summary, or as "incorrect" otherwise. The trainable summarizer is expected to "learn" the patterns which lead to the summaries, by identifying relevant feature values which are most correlated with the classes "correct" or "incorrect". When a new document is given to the system, the "learned" patterns are used to classify each sentence of that document into either a "correct" or "incorrect" sentence, producing an extractive summary. A crucial issue in this framework is how to obtain the relevant set of features; the next section treats this point in more detail.

3 A trainable summarizer using a ML approach

We concentrate our presentation in two main points: (1) the set of employed features; and (2) the framework defined for the trainable summarizer, including the employed classifiers.

A large variety of features can be found in the text-summarization literature. In our proposal we employ the following set of features:

(a) Mean-TF-ISF. Since the seminal work of Luhn [9], text processing tasks frequently use features based on IR measures [5], [7], [23]. In the context of IR, some very important measures are term frequency (TF) and term frequency $\times$ inverse document frequency (TF-IDF) [20]. In text summarization we can employ the same idea: in this case we have a single document d, and we have to select a set of relevant sentences to be included in the extractive summary out of all sentences in d. Hence, the notion of a collection of documents in IR can be replaced by the notion of a single document in text summarization. Analogously the notion of document – an element of a collection of documents – in IR, corresponds to the notion of sentence – an element of a document – in summarization. This new measure will be called term frequency × inverse sentence frequency, and denoted TF-ISF(w,s) [8].The final used feature is calculated as the mean value of the TF-ISF measure for all the words of each sentence.

(b) Sentence Length. This feature is employed to penalize sentences that are too short, since these sentences are not expected to belong to the summary [7]. We use the normalized length of the sentence, which is the ratio of the number of words occurring in the sentence over the number of words occurring in the longest sentence of the document.

(c) Sentence Position. This feature can involve several items, such as the position of a sentence in the document as a whole, its the position in a section, in a paragraph, etc., and has presented good results in several research projects [5], [7], [8], [11], [23]. We use here the percentile of the sentence position in the document, as proposed by Nevill-Manning [16]; the final value is normalized to take on values between 0 and 1.

(d) Similarity to Title. According to the vectorial model, this feature is obtained by using the title of the document as a "query" against all the sentences of the document; then the similarity of the document's title and each sentence is computed by the cosine similarity measure [20].

(e) Similarity to Keywords. This feature is obtained analogously to the previous one, considering the similarity between the set of keywords of the document and each sentence which compose the document, according to the cosine similarity.

For the next two features we employ the concept of text cohesion. Its basic principle is that sentences with higher degree of cohesion are more relevant and should be selected to be included in the summary [1], [4], [11], [15].

(f) Sentence-to-Sentence Cohesion. This feature is obtained as follows: for each sentence s we first compute the similarity between s and each other sentence s' of the document; then we add up those similarity values, obtaining the raw value of this feature for s; the process is repeated for all sentences. The normalized value (in the range [0, 1]) of this feature for a sentence s is obtained by computing the ratio of the raw feature value for s over the largest raw feature value among all sentences in the document. Values closer to 1.0 indicate sentences with larger cohesion.

(g) Sentence-to-Centroid Cohesion. This feature is obtained for a sentence s as follows: first, we compute the vector representing the centroid of the document, which is the arithmetic average over the corresponding coordinate values of all the sentences of the document; then we compute the similarity between the centroid and each sentence, obtaining the raw value of this feature for each sentence. The normalized value in the range [0, 1] for s is obtained by computing the ratio of the raw feature value over the largest raw feature value among all sentences in the document. Sentences with feature values closer to 1.0 have a larger degree of cohesion with respect to the centroid of the document, and so are supposed to better represent the basic ideas of the document.

For the next features an approximate argumentative structure of the text is employed. It is a consensus that the generation and analysis of the complete rethorical structure of a text would be impossible at the current state of the art in text processing. In spite of this, some methods based on a surface structure of the text have been used to obtain good-quality summaries [23], [24]. To obtain this approximate structure we first apply to the text an agglomerative clustering algorithm. The basic idea of this procedure is that similar sentences must be grouped together, in a bottom-up fashion, based on their lexical similarity. As result a hierarchical tree is produced, whose root represents the entire document. This tree is binary, since at each step two clusters are grouped. Five features are extracted from this tree, as follows:

(h) Depth in the tree. This feature for a sentence *s* is the depth of *s* in the tree.

(i) Referring position in a given level of the tree (positions 1, 2, 3, and 4). We first identify the path form the root of the tree to the node containing *s*, for the first four depth levels. For each depth level, a feature is assigned, according to the direction to be taken in order to follow the path from the root to *s*; since the argumentative tree is binary, the possible values for each position are: left, right and none, the latter indicates that *s* is in a tree node having a depth lower than four.

(j) Indicator of main concepts. This is a binary feature, indicating whether or not a sentence captures the main concepts of the document. These main concepts are obtained by assuming that most of relevant words are nouns. Hence, for each sentence, we identify its nouns using a part-of-speech software [3]. For each noun we then compute the number of sentences in which it occurs. The fifteen nouns with largest occurrence are selected as being the main concepts of the text. Finally, for each sentence the value of this feature is considered "true" if the sentence contains at least one of those nouns, and "false" otherwise.

(k) Occurrence of proper names. The motivation for this feature is that the occurrence of proper names, referring to people and places, are clues that a sentence is relevant for the summary. This is considered here as a binary feature, indicating whether a sentence *s* contains (value "true") at least one proper name or not (value "false"). Proper names were detected by a part-of-speech tagger [3].

(l) Occurrence of anaphors. We consider that anaphors indicate the presence of non-essential information in a text: if a sentence contains an anaphor, its information content is covered by the related sentence. The detection of anaphors was performed in a way similar to the one proposed by Strzalkowski [22]: we determine whether or not certain words, which characterize an anaphor, occur in the first six words of a sentence. This is also a binary feature, taking on the value "true" if the sentence contains at least one anaphor, and "false" otherwise.

(m) Occurrence of non-essential information. We consider that some words are indicators of non-essential information. These words are speech markers such as "because", "furthermore", and "additionally", and typically occur in the beginning of a sentence. This is also a binary feature, taking on the value "true" if the sentence contains at least one of these discourse markers, and "false" otherwise.

The ML-based trainable summarization framework consists of the following steps:

1. We apply some standard preprocessing information retrieval methods to each document, namely stop-word removal, case folding and stemming. We have employed the stemming algorithm proposed by Porter [17].
2. All the sentences are converted to its vectorial representation [20].
3. We compute the set of features described in the previous subsection. Continuous features are discretized: we adopt a simple "class-blind" method, which consists of separating the original values into equal-width intervals. We did some experiments with different discretization methods, but surprisingly the selected method, although simple, has produced better results in our experiments.

4. A ML trainable algorithm is employed; we employ two classical algorithms, namely C4.5 [18] and Naive Bayes [14]. As usual in the ML literature, we employ these algorithms trained on a training set and evaluated on a separate test set.

The framework assumes, of course, that each document in the collection has a reference extractive summary. The "correct" sentences belonging to the automatically produced extractive summary are labeled as "positive" in classification/data mining terminology, whereas the remaining sentences are labeled as "negative". In our experiments the extractive summaries for each document were automatically obtained, by using an author-provided non-extractive summary, as explained in section 2.

4 Computational results

As previously mentioned, we have used two very well-known ML classification algorithms, namely Naive Bayes [14] and C4.5 [18]. The former is a Bayesian classifier which assumes that the features are independent from each other. Despite this unrealistic assumption, the method presents good results in many cases, and it has been successfully used in many text mining projects. C4.5 is a decision-tree algorithm that is frequently employed for comparison purposes with other classification algorithms, particularly in the data mining and ML communities.

We did two series of experiments: in the first one, we employed automatically-produced extractive summaries; in the second one, manually-produced summaries were employed. In all the experiments we have used a document collection available in the TIPSTER document base [6]. This collection consists of texts published in several magazines about computers, hardware, software, etc., which have sizes varying from 2 Kbytes to 64 Kbytes. Due to our framework, we used only documents which have an author-provided summary, and a set of keywords. The whole TIPSTER document base contained 33,658 documents with these characteristics. A subset of these documents was randomly selected for the experiments to be reported in this section.

In the first experiment, using automatically-generated reference extractive summaries, we employed four text-summarization methods, as follows:

(a) Our proposal (features as described in section 3) using C4.5 as the classifier;

(b) Our proposal using Naive Bayes as the classifier.

(c) First Sentences (used as a baseline summarizer): this method selects the first *n* sentences of the document, where *n* is determined by the desired compression rate, defined as the ratio of summary length to source length [12], [21]. Although very simple, this procedure provides a relatively strong baseline for the performance of any text-summarization method [2].

(d) Word Summarizer (WS): Microsoft's WS is a text summarizer which is part of Microsoft Word, and it has been used for comparison with other summarization methods by several authors [1], [13]. This method uses non-documented techniques to perform an "almost extractive" summary from a text, with the summary size specified by the user.

The WS has some characteristics that are different from the previous methods: the specified summary size refers to the number of characters to be extracted, and some sentences can be modified by WS. In our experiments due to these characteristics a direct comparison between WS and the other methods is not completely fair: (i) the summaries generated by WS can contain a few more or a few less sentences than the summaries produced by the other methods; (ii) in some cases it will not be possible to compute an exact match between a sentence selected by WS and an original sentence; in these cases we ignore the corresponding sentences.

It is important to note that only our proposal is based on a ML trainable summarizer; the two remaining methods are not trainable, and were used mainly as baseline for result comparison.

The document collection used in this experiment consisted of 200 documents, partitioned into disjoints training and test sets with 100 documents each. The training set contained 25 documents of 11 Kbytes, 25 documents of 12 Kbytes, 25 documents of 16 Kbytes, and 25 documents of 31 Kbytes. The average number of sentences per document is 129.5, since there are in total 12,950 sentences in the training set. The test set contained 25 documents of 10 Kbytes, 25 documents of 13 Kbytes, 25 documents of 15 Kbytes, and 25 documents of 28 Kbytes. The average number of sentences per document is 118.6, since there are in total 11,860 sentences in the test set.

Table 1 reports the results obtained by the four summarizers. We consider compression rates of 10 % and 20 %. The performance is expressed in terms of precision / recall values, expressed in percentage (%), and the corresponding standard deviations are indicated after the "±" symbol. The best obtained results are shown in boldface.

Table 1. Results for training and test sets composed by automatically-produced summaries

Summarizer	Compression rate: 10%		Compression rate: 20%	
	Precision	Recall	Precision	Recall
Trainable-C4.5	22.36 ± 1.48		34.68 ± 1.01	
Trainable-Bayes	**40.47 ± 1.99**		**51.43 ± 1.47**	
First-Sentences	23.95 ± 1.60		32.03 ± 1.36	
Word-Summarizer	26.13 ± 1.21	34.44 ± 1.56	38.80 ± 1.14	43.67 ± 1.30

We can draw the following conclusions from this experiment: (1) the values of precision and recall for all the methods are significantly higher with the rate of 20% than with the compression rate of 10%; this is a expected result, since the larger the compression rate, the larger the number of sentences to be selected for the summary, and then the larger the probability that a sentence selected by a summarizer matches with a sentence belonging to the extractive summary; (2) the best results were obtained by our trainable summarizer with Naive Bayes classifier for both compression rates; using the same features, but with the C4.5 as classifier, the obtained results were poor: the results are similar to the First-Sentences and Word Summarizer baselines.

The latter result offers us an interesting lesson: most research projects on trainable summarizers focus on the proposal of new features for classification, trying to produce more and more elaborate statistics-based or linguistics-based features, but they usually employ a single classifier in the experiments. Normally "conventional" classifiers are used. Our results indicate that researchers should concentrate their attention in the study of more elaborate classifiers, tailored for the text-summarization task, or at least evaluate and select the best classifier among the conventional ones already available.

In the second experiment we employ in the test step summaries manually produced by a human judge. We emphasize that in the training phase of our proposal we have used the same database of automatically-generated summaries employed in the previous experiment.

The test database was composed of 30 documents, selected at random from the original document base. The manual reference summaries were produced by a human judge – a professional English teacher with many years of experience –specially hired for this task. For the compression rates of 10 % and 20% the same four summarizers of the first experiment were compared. The obtained results are presented in Table 2.

Table 2. Results for training set composed by automatically-produced summaries and test set composed by manually-produced summaries

Summarizer	Compression rate: 10%		Compression rate: 20%	
	Precision	Recall	Precision	Recall
Trainable-C4.5	24.38 ± 2.84		31.73 ± 2.41	
Trainable-Bayes	**26.14 ± 3.32**		**37.50 ± 2.29**	
First-Sentences	18.78 ± 2.54		28.01 ± 2.08	
Word-Summarizer	14.23 ± 2.17	17.24 ± 2.56	24.79 ± 2.22	27.56 ± 2.41

Here again the best results were obtained by our proposal using the Naive Bayes algorithm as classifier. Similar to the previous experiment, results for 20% of compression were superior to the results produced with 10% of compression.

In order to verify the consistency between the two experiments we have compared the manually-produced summaries and the automatically-produced ones. We considered here the manually-produced summaries as a reference, and we calculated the precision and recall for the automatically produced summaries of the same documents. Obtained results are presented in Table 3. These results are consistent with the ones presented by Mitra [15], and indicate that the degree of dissimilarity between a manually-produced summary and an automatically-produced summary in our experiments is comparable to the dissimilarity between two summaries produced by different human judges.

Table 3. Comparison between automatically-produced and manually-produced summaries

	Precision / Recall
Compression rate: 10%	30.79 ± 3.96
Compression rate: 20%	42.98 ± 2.42

5 Conclusions and future research

In this work we have explored the framework of using a ML approach to produce trainable text summarizers, in a way which was proposed a few years ago by Kupiec [7]. We have chosen this research direction because it allows us to measure the results of a text summarization algorithm in an objective way, similar to the standard evaluation of classification algorithms found in the ML literature. This avoids the problem of subjective evaluation of the quality of a summary, which is a central issue in the text summarization research.

We have performed an extensive investigation of that framework. In our proposal we employ a trainable summarizer that uses a large variety of features, some of them employing statistics-oriented procedures and others using linguistics-oriented ones. For the classification task we have used two different well known classification algorithms, namely the Naive Bayes algorithm and the C4.5 decision tree algorithm. Hence, it was possible to analyze the performance of two different text-summarization procedures. The performance of these procedures was compared with the performance of two non-trainable, baseline methods.

We did basically two kind of experiments: in the first one we considered automatically-produced summaries for both the training and test phases; in the second experiment we used automatically-produced summaries for training and manually-produced summaries for testing. In general the trainable method using Naive Bayes classifier significantly outperformed all the baseline methods.

An interesting finding of our experiments was that the choice of the classifier (Naive Bayes versus C4.5) strongly influenced the performance of the trainable summarizer. We intend to focus mainly on the development of a new or extended classification algorithm tailored for text summarization in our future research work.

6 References

1. Barzilay, R. ; Elhadad, M. Using Lexical Chains for Text Summarization. In Mani, I.; Maybury, M. T. (eds.). In *Proceedings of the ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, Association of Computional Linguistics (1997).
2. Brandow, R.; Mitze, K., Rau, L. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management* 31(5) (1994) 675-685
3. Brill, E. A simple rule-based part-of-speech tagger. In *Proceedings of the Third Conference on Applied Comp. Linguistics.* Assoc. for Computational Linguistics (1992)
4. Carbonell, J. G.; Goldstein, J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR-98* (1998)
5. Edmundson, H. P. New methods in automatic extracting. *Journal of the Association for Computing Machinery* 16 (2) (1969) 264-285
6. Harman, D. Data Preparation. In Merchant, R. (ed.). *The Proceedings of the TIPSTER Text Program Phase I.* Morgan Kaufmann Publishing Co. (1994)
7. Kupiec, J. ; Pedersen, J. O.; Chen, F. A trainable document summarizer. In *Proceedings of the 18th ACM-SIGIR Conference, Association of Computing Machinery* (1995) 68-73
8. Larocca Neto, J.; Santos, A. D.; Kaestner, C.A.; Freitas, A.A.. Document clustering and text summarization. *Proc. of 4th Int. Conf. Practical Applications of Knowledge Discovery and Data Mining (PADD-2000)* London: The Practical Application Company (2000) 41-55

9. Luhn, H. The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2(92) (1958) 159-165
10. Mani, I.; House, D.; Klein, G.; Hirschman, L.; Obrsl, L.; Firmin, T.; Chrzanowski, M.; Sundheim, B. *The TIPSTER SUMMAC Text Summarization Evaluation.* MITRE Technical Report MTR 98W0000138. The MITRE Corporation (1998)
11. Mani, I.; Bloedorn, E. Machine Learning of Generic and User-Focused Summarization. In *Proceedings of the Fifteenth National Conference on AI (AAAI-98)* (1998) 821-826
12. Mani, I. *Automatic Summarization*. J.Benjamins Publ. Co. Amsterdam Philadelphia (2001)
13. Marcu, D. Discourse trees are good indicators of importance in text. In Mani., I.; Maybury, M. (eds.). *Adv. in Automatic Text Summarization*. The MIT Press (1999) 123-136
14. Mitchell, T. *Machine Learning*. McGraw-Hill (1997)
15. Mitra, M.; Singhal, A.; Buckley, C. Automatic text summarization by paragraph extraction. In *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*. Madrid (1997)
16. Nevill-Manning, C. G. ; Witten, I. H. Paynter, G. W. et al. KEA: Practical Automatic Keyphrase Extraction. *ACM DL 1999* (1999) 254-255
17. Porter, M.F. An algorithm for suffix stripping. *Program 14*, 130-137. 1980. Reprinted in: Sparck-Jones, K.; Willet, P. (eds.) *Readings in Information Retrieval*. Morgan Kaufmann (1997) 313-316
18. Quinlan, J. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Sao Mateo California (1992)
19. Rath, G. J. ; Resnick A. ; Savvage R. The formation of abstracts by the selection of sentences. *American Documentation* 12 (2) (1961) 139-141
20. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24,* 513-523. 1988. Reprinted in: Sparck-Jones, K.; Willet, P. (eds.) *Readings in I.Retrieval*. Morgan Kaufmann (1997) 323-328
21. Sparck-Jones, K. Automatic summarizing: factors and directions. In Mani, I.; Maybury, M. *Advances in Automatic Text Summarization*. The MIT Press (1999) 1-12
22. Strzalkowski , T.; Stein, G.; Wang, J.; Wise, B. A Robust Practical Text Summarizer. In Mani, I.; Maybury, M. (eds.), *Adv. in Autom. Text Summarization.* The MIT Press (1999)
23. Teufel, S.; Moens, M. Argumentative classification of extracted sentences as a first step towards flexible abstracting. In Mani, I.; Maybury M. (eds.). *Advances in automatic text summarization.* The MIT Press (1999)
24. Yaari, Y. Segmentation of Expository Texts by Hierarchical Agglomerative Clustering. *Technical Report*, Bar-Ilan University Israel (1997)