S. Krajči (ed.): ITAT 2018 Proceedings, pp. 28–34
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Matej Gallo, Luboš Popelínský, and Karel Vaculík

ITAT

To text summarization by dynamic graph mining

Matej Gallo, Luboš Popelínský, and Karel Vaculík

KD Lab FI MU Brno
popel@fi.muni.cz

Abstract: We show that frequent patterns can contribute to the quality of text summarization. Here we focus on single-document extractive summarization in English. Performance of the frequent patterns based model obtained with DGRMiner yields the most relevant sentences of all compared methods. Two out of three proposed methods outperform other methods if compared on ROUGE data.

1 Introduction

Extractive summarization assigns a score to each text unit (phrase, sentence, paragraph, passage) and then picks the most informative ones to form a summary called extract. The selected sentences are used verbatim [1, 25].

Graph representation is a common way for representing data in automatic text summarization tasks. We introduce a new method for single-document extractive summarization in English language where a text is represented as a dynamic graph and each sentence corresponds to a dynamic graph snapshot, i.e. the graph in a partivular time. E.g. the first sentence is the oldest snapshot while the last sentence of a text is the newest one. This method is based on the principle of mining frequent patterns from such a dynamic graph and using the resulting patterns as indicators of sentence importance.

The structure of this text is following. In Section 2. we introduce DGRMiner, a tool for mining in dynamic graphs. Section 3. describes the algorithm. In Sections 4 we briefly introduce the data used for evaluation. Sections 5 and 6 explain a way of summarization evaluation and sentence scoring. Section 7 displays the main results of text summarization by means of frequent patterns mined from dynamic graphs. Related work is a contents of Section 8. We conclude with concluding Section 9.

2 DGRMiner

We used DGRMiner tool to extract these patterns from the dynamic graphs. DGRMiner was proposed in [26] for mining frequent patterns that capture various changes in dynamic graphs in the form of predictive rules. The found predictive graph rules express what way the graph changes. The rules capture patterns such as addition, deletion, and transformation of the subgraph. The use of relative timestamps for rules allows for mining general patterns while including time information simultaneously. To ensure that only significant rules are extracted, the algorithm incorporates measuring support and confidence. While support expresses what portion of the graph is affected by the rule, confidence measures the occurrence frequency of a specific change, given that a particular pattern was observed. Time abstraction is an extension to the DGRMiner that allows us to analyse broader class of dynamic graphs. The abstraction lies in the use of signum function on relative timestamps. All the negative timestamps become -1, all the positive timestamps become 1 and the timestamp of 0 remains 0. DGRMiner allows for two types of abstraction. One affects only timestamps of vertices and should be used when most changes are caused by edges and vertices remain static. The second one also affects the edges and is useful when there are too few patterns with exact timestamps.

3 Method

The initial idea was to take a text as a temporal (i.e. dynamic) graph where each sentence represent a graph snapshot at a particular time and tokens (a lemma together with a part-of-speech tag) were nodes and edges connected all pairs of tokens. The label of an edge was equal to a number of appearances of this pair of tokens. In the following subsections we describe particular steps of the algorithm.

3.1 Transforming Text to graphs

For pre-processing we employed CoreNLP [15]. CoreNLP provides a set of natural language analysis tools: POS tagger, named entity recognition, parser, coreference resolution system, sentiment analysis, bootstrapped pattern learning, and open information extraction. We used four annotators: sentence split, tokenize, lemma, and POS.

Using the coreNLP package for R, we split the text into sentences. In every iteration we removed the stop words from a sentence, lemmatize the remaining words and assigned the POS tags. If a lemma-tag pair was not already in the graph, we added it in and created edges between all words in a same sentence. Otherwise, we only notified the DGRMiner that a particular word had appeared again. We parametrized context c. Therefore, in each step we modelled at most c sentences.

We will show it on simple example that contains a single sentence.

```
<sentence position = " 9 " labelers
```

```
= "1, 2 , 3, 4">The great thing about
Aspen is that it has at least one of
each of the really useful stores that
you need, sellingstuff at regular
prices. </sentence>
```

After removing stop words and labeling words with a lemma-tag pair, we receive a graph `t # 9`.

```
t # 9
an 74 great#ADJ
an 75 thing#NOUN
an 11 Aspen#NOUN
an 76 least#ADJ
an 77 one#NUM
an 78 really#ADV
an 79 useful#ADJ
an 66 store#NOUN
an 5 need#VERB
ae u 48 5 11 [need#VERB]-[Aspen#NOUN]
ae u 432 5 66 [need#VERB]-[store#NOUN]
ae u 433 5 67 [need#VERB]-[sell#VERB]
ae u 434 5 74 [need#VERB]-[great#ADJ]
ae u 435 5 75 [need#VERB]-[thing#NOUN]
ae u 436 5 76 [need#VERB]-[least#ADJ]
ae u 437 5 77 [need#VERB]-[one#NUM]
```

In the first column, `an, ae` stand for "add a node" and "add an edge" respectively.

3.2 Pattern Mining

Then we applied the DGRMiner on the data from previous step to obtain predictive patterns, i.e. rules that describe frequent (or rare) changes between past and future graphs – sentences. We received two types of patterns: frequent single-vertex patterns corresponding to nodes and rare patterns corresponding to edges. We modified the support parameter to change the threshold on frequency of observed patterns. When the support parameter was set too low, the DGRMiner considered a pattern anything that appeared at least once in the text – every word, every possible word combination. By setting the confidence parameter in DGRMiner to 0, the DGRMiner assigned confidence (as discussed in section 2) to each extracted pattern. We also played with time abstraction which allowed us to ignore the preset context c as discussed in section 2. Therefore, patterns were observed in sentences that were more than $c-2$ sentences apart. Although this gave us more patterns, many of them were uninformative. An example of observed single-vertex pattern is "+supplies#NOUN". This pattern indicates that the noun *supplies* appeared at least n times in the text, where the n is determined by the support parameter. An example of multi-vertex pattern is [store#NOUN]-[sell#VERB], which tells us that the noun *store* is frequently closely accompanied by the verb *sell*. The proximity of these two words is given by context c and the frequency is given by the support parameter.

4 Data

To assess the performance of our method we used the Blog summarization dataset [10, 11, 23]. It consists of 100 posts annotated in XML that were randomly chosen from two blogs (half from each blog), Cosmic Variance and Internet Explorer Blog. Each of the four human summarizers picked approximately 7 sentences to form 4 reference summaries in total. We manually restored apostrophes for shortened forms of to be and to have verbs, and in possessive nouns. Punctuation within sentences was omitted as the coreNLP sentence split annotator often wrongly split the sentences in the middle. We decided not to use CNN Dataset, nor SUMMAC dataset mentioned in [9] because the provided reference summaries were not extracted, verbatim sentences. This would require us to manually find the best matching sentence from within the document.

5 Semi-automatic evaluation

Evaluating summaries on sentence level can be done semi-automatically by measuring content overlap with precision, recall, and F1 measure. An extracted sentence is considered acceptable if the same sentence was extracted in a reference summary. This process cannot be fully automatized because reference summaries are created by human judges. Other semi-automatic evaluation methods used nowadays are: ROUGE, PYRAMID and BASIC ELEMENTS. We will discuss only the ROUGE method [25, 14] as it was used in this work.

5.1 ROUGE

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measure is based on the BLEU metrics used in machine translation tasks. The idea is to compare the differences between the distribution of words in the candidate summary and the distribution of words in the reference summaries. Given h reference summaries and a candidate summary they are split into n-grams to calculate the intersection of n-grams between the references and the candidate. This process is illustrated in figure 1.

Given its correlation with manual judgments ROUGE almost seems to have become a standard. [13] reports Pearson coefficient for the most commonly used variations (ROUGE-2 and ROUGE-SU4 [14]) at a value of between 0.94 and 0.99. Generally, the ROUGE-n is calculated from the co-occurrences of n-grams between the candidate and reference summaries as shown by formula 1 in [25]:

$$\text{ROUGE-}n = \frac{\sum_{n\text{-grams}} \in \{\text{Sum}_{\text{can}} \cap \text{Sum}_{\text{ref}}\}}{\sum_{n\text{-grams}} \in \text{Sum}_{\text{ref}}} \qquad (1)$$

where the numerator is the maximum number of co-occurrences of n-grams in both reference and candidate

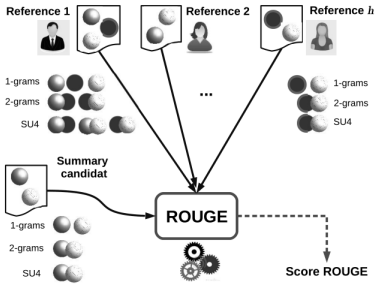

Figure 1: The basic idea of ROUGE for evaluation of summaries [25]

summary and the denominator is the total sum of the number of n-grams present in the reference summaries.

The ROUGE-SUγ is an adaptation of ROUGE-2 using skip units (SU) of a size $\leq \gamma$. SU4 considers the bigrams and the bigrams SU to be arbitrary in a maximum window length of $\gamma = 4$ words. The number of bigrams with an arbitrary size of length γ is given by formula 2:

$$\text{Count}(k,n) = C\binom{n}{k} - \sum_{0}^{k-\gamma}(k-\gamma); \gamma \leq 1 \qquad (2)$$

where n is the n-gram length and k is the sentence length in words.

The ROUGE metrics are not flawless. Firstly, they have a problem with the representation of content. Secondly, they will not consider chains of words such as "MUNI" $\neq$ "Masaryk University" and "FI" $\neq$" Faculty of Informatics". A study [22] found that the system can be tricked into generating a summary with high ROUGE score.

6 Sentence Scoring

In this step we used the observed patterns as indicators to score the sentences. The final score was obtained according to formula 3 as a sum of single-vertex and multi-vertex scores:

$$\text{Score}(s) = \text{Score}_{single}(s) + \text{Score}_{multi}(s) \qquad (3)$$

Three different metrics were implemented to calculate single-vertex score.

The Jaccard coefficient (**JAC**) as given by formula 4, expresses the overlap of two sets. In our case, between the set of patterns P and the set of words in the sentence S.

$$\text{Score}_J(s) = \frac{w_{sum}(S \cap P)}{w_{sum}(S \cup P)} \qquad (4)$$

where w_{sum} assigns weight to every element of the set and then sums over them. A question arises regarding what weight we should assign to non-indicators. Empirically, we received the best results for the value of 0.001.

The frequency method (**FRQ**) as given by formula 5, is the simplest of the three. For a sentence s and a set of patterns P the score is calculated as weighted average.

$$\text{Score}_F(s) = \sum_{p \in P} c(p)w(p) \qquad (5)$$

where $c(p)$ is the frequency of the pattern in the sentence and $w(p)$ is its associated weight.

The density method (**DEN**) as given by formula 6, counts the patterns in a sentence and normalizes by the length of the sentence. This method is parameter-free.

$$\text{Score}_D(s) = \frac{|\, S \cap P \,|}{\text{length}(s)} \qquad (6)$$

For multi-vertex patterns we tested frequency and density methods. The only difference between the methods is that instead of length of sentences we use the number of all possible word pair combinations $\binom{|S|}{2}$.

We built the summary in a greedy fashion. In every iteration we picked the highest scoring sentence. Patterns observed in the sentence were penalized by a parameter λ. The scores were recomputed and the process continued until a desired number of sentences was picked or until all the sentences were used. For every method we discovered the optimal value of λ. We tuned the parameter on 10% of the entire dataset. The optimal values for λ are presented in the tables 2 and 1. The ordering of sentences in the final summary maintains the relative ordering in the original document.

7 Results

We evaluated the model using the ROUGE-1 metric, which is recommended for short summaries [14]. Every blog post was compared against four reference summaries as described in chapter 4. The results of all three methods can be seen in tables 1, 2 and 3. The first three columns denote whether time abstraction on both vertices and edges is used, whether the patterns were used to score sentences, and whether the initial weights were determined by the confidence of DGRMiner for the given pattern, respectively. The last three columns correspond to the ROUGE-1 metrics – precision (what portion of sentences we selected were part of the reference summary), recall (what portion of sentences in reference summary we extracted), and f1 measure (harmonic mean of precision and recall).

The FRQ method marked the most accurate sentences among all the presented algorithms as can be seen from figure 2. JAC method picked fewer correct sentences than FRQ method but still more than any traditional approach. Both FRQ and DEN methods ranked first in terms of precision.

The frequency-based method incorporating patterns with no confidence weighting (identified as FRQ) achieved the highest recall, the highest f1 score, and the highest

binall	patterns	weights	precision	recall	f1 score
F	F	F	0.643	0.694	0.664
F	T	F	0.643	0.686	0.659
F	T	T	0.641	0.684	0.657
T	F	F	0.641	0.683	0.657
F	F	T	0.640	0.691	0.660
T	T	F	0.640	0.678	0.654
T	T	T	0.640	0.678	0.654
T	F	T	0.639	0.681	0.655

Table 1: Results for blog summarization dataset using jaccard method with parameters $\lambda = 0.55$, $w_0 = 0.001$

binall	patterns	weights	precision	recall	f1 score
F	T	F	0.645	0.702	0.668
F	F	T	0.645	0.700	0.667
F	F	F	0.644	0.701	0.665
F	T	T	0.643	0.693	0.662
T	F	F	0.642	0.691	0.661
T	T	F	0.642	0.691	0.661
T	F	T	0.642	0.690	0.660
T	T	T	0.642	0.688	0.660

Table 2: Results for blog summarization dataset using frequency method with parameter $\lambda = 0.20$

binall	patterns	precision	recall	f1 score
F	T	0.651	0.514	0.562
F	F	0.650	0.514	0.562
T	T	0.649	0.516	0.563
T	F	0.649	0.512	0.562

Table 3: Results for blog summarization dataset using density method

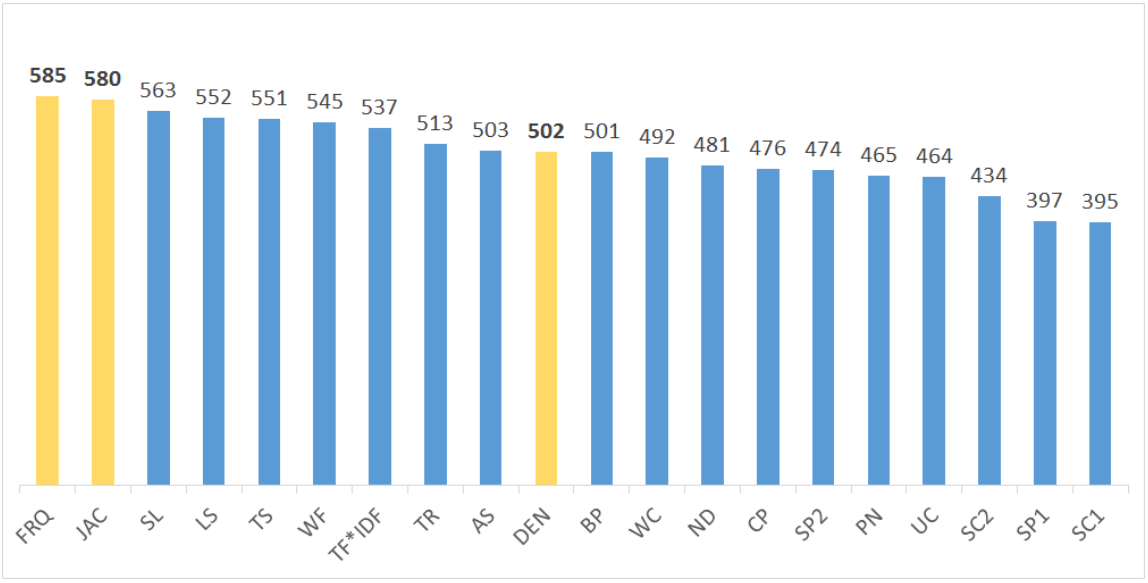

Figure 2: Comparison of number of correctly chosen sentences – using blog summarization dataset (our algorithms are displayed yellow)

number of correctly chosen sentences. The highest precision was achieved by density-based method incorporating patterns with no confidence weighting (identified as DEN). We chose these two models and the highest scoring Jaccard method for comparison together with algorithms presented in article [9]. We could see that FRQ behaves similarly to *Word frequency (WF)* algorithm proposed in the paper. This is not surprising, because the single vertex-frequent patterns correspond to words that appear at least n-times in the text. Where n is determined by the support parameter in the DGRMiner tool as described in section 3.2. The multi-vertex patterns improved the performance of the summarizer in density models and in one case (the highest scoring model) in frequency model.

8 Related work

Kupiec et al. introduce in their work [12] method inspired by Edmundson's [7]. They approached the summarization as a statistical classification problem. A Bayes classifier was trained to estimate the probability that a given sentence would be included in the summary. They used 6 discrete features (presented in order of importance): paragraph feature (the position of sentence in paragraph s), fixed-phrase feature (the sentence contains a phrase from a list), sentence length cutoff feature (threshold $u_1 = 5$, length$(s) > u_1$), thematic word feature (the presence of thematic terms), uppercase word feature (the presence of words in capital letters). The best results were obtained using the first three features.

Aone et al. [2] built on Kupiec's work [12] and expanded the feature set of their system, called DIMSUM, with signature terms, which indicate key concepts for a given document. Another advantage over [12]'s system is the use of multi-word phrases – statistically derived collocation phrases (e.g. "omnibus bill", "crime bill", "brady bill") and associated words ("camera" and "obscura", "Columbia River" and "gorge"), as well as the use of WordNet [16] to identify possible synonyms of found signature terms. He applied a shallow discourse analysis to resolve co-references and maintain cohesion – only name aliases were resolved such as UK to United Kingdom.

Osborne in his work [18] disagrees with the traditional assumption of feature independence and shows empirically that the maximum entropy (MaxEnt) model produces better extracts than the naïve Bayes model with similarly optimized prior appended to both models. Unlike naïve Bayes, MaxEnt does not make unnecessary feature independence assumptions. Let c be a binary label (binary: part of summary or not), s the item we are interested in labeling, f_i the i-th feature, and ω_i the corresponding feature weight.

Hidden Markov Models (HMM), similar to MaxEnt, have weaker assumptions of independence. There are three types of dependencies: positional dependence, feature dependence, and Markovity dependence. A first-order Markov model allows modeling these dependencies. Conroy and O'leary [6] use a joint distribution for the features set, unlike the independence-of-features assumption used in naïve Bayesian methods. The HMM was trained using five features: position of the sentence in the document (number of states); number of terms in a sentence, and likeliness of the sentence terms given the document terms.

Summarizer NetSum presented by Svore et al. [24] uses an artificial neural network (ANN) called RankNet to rank the sentences. RankNet is a pair-based neural network algorithm for ranking a set of inputs. It is trained on pairs of sentences (S_i, S_j), such that the ROUGE score for S_i should be higher than S_j. Pairs are only generated in a single document, not across documents. The cost function for RankNet is the probabilistic cross-entropy cost function. Training is performed using a modified version of back-propagation algorithm for two-layer networks, which is based on optimizing the cost function by gradient descent. The system significantly outperforms the standard baseline in the ROUGE-1 measure. No past system could outperform the baseline with statistical significance.

A system for generating product category-based (topic-based) extractive summarization was proposed by [4, 5]. The collection of 45 news items corresponding to various products are pre-processed using standard techniques: tokenization, stopword removal, stemming. The final corpus contains around 1500 features and is represented by a bag-of-words VSM based on these features. To identify the topics, the news items about specific categories of products are segregated into separate clusters using K-Means and then an extractive summary is generated from each of these topical clusters. The K number of cluster is determined by a Self-Organizing Map (SOM).

Chakraborti and Dey [5] assigned a score to the entire summary as a single unit. The total summary score (TSS) is taken as a combination of cosine similarity between centroid of corpus and the summary as a whole; relative length of the summary; and redundancy penalty. To maximize TSS constrained by the number of lines in summary ($\tau = 5-7\%$), they opted for quick Artificial Bee Colony optimization, a global optimization technique, for sentence selection. The summary with the highest score is then chosen.

In Muresan's paper [17], a system called GIST-IT used for email-summarization task is discussed. First, noun phrases (NPs) are extracted as they carry the most contentful information. Subsequently, machine learning is used to select the most salient NPs. A set of nine features, divided into three categories (head of the NP, whole NP, combination of head and modifiers of NP) were used: head of the NP TF*IDF, position of first occurrence (focc) of the head in text, TF*IDF of entire NP, focc of entire NP, length of the NP in words, length of the NP in characters, position of the NP in the sentence, position of the NP in the paragraph, and combination of the TF*IDF scores of head of the NP and its modifiers.

To find the most salient NPs, three machine learning al-

gorithms were applied: decision trees (axis-parallel trees – C4.5 and oblique trees – OC1), rule induction (production rule – C4.5rules and propositional rules – RIPPER), and decision forests (DFC using information gain ratio).

Muresan claims that shallow linguistic filtering applied to NPs improved the classifiers accuracy [17]. The filtering consisted of four steps: grouping inflectional variants, removing unimportant modifiers, filtering stopwords, and removing empty nouns.

A modification of PageRank algorithm called LexRank is used to weight sentences. The undirected graph of sentences is constructed from symmetrical similarities (modified cosine measure). The score of each vertex s is calculated iteratively until the values of the vertices have not been modified by more than $\varepsilon = 0.0001$. LexRank algorithm is used as a component of the MEAD [8].

Unlike LexRank, TextRank uses the similarities of edges to weight the vertices. The score of each sentence s_i is calculated iteratively until convergence is reached.

As the graph is constructed from inter-sentence similarity measure, the choice of the method for sentence-weighting has significant impact. One approach is to use a bag-of-words to represent the sentences. The similarity is obtained by calculating the cosine similarity weighted by inverse document frequency [1] between their vectorial representations. Another approach suggests using word overlap between sentences instead. The weak point of all similarity measures that use words (cosine similarity, word overlap, longest common subsequence) is the dependency on the lexicon of the document. The solution to this is its combining with similarity measure based on chains of characters. Therefore, sentences that do not share a single word but contain a number of words that are close morphologically can be compared [25].

Patil [19] proposed a new graph-based model called SUMGRAPH. First the text is pre-processed (stemmed) and represented as a VSM with TF*IDF weights. He then computes pair-wise cosine similarities and subtracts the values from 1 to obtain dissimilarities. The resulting matrix of intra-sentence dissimilarities is then used to model the document as graph. The vertices represent the sentences and edges are weighted by intra-sentence dissimilarities.

The novel idea is the use of link reduction technique known as Pathfinder Network Scaling (PFnet) [21, 20] to scale the graph. PFnet models the human aspects of semantic memory. The centrality of a sentence and its position in a document are used to compute the importance of sentences. Four different centrality measure were tested and closeness centrality showed to perform best. Finally, the sentences are ranked according to their importance and first n highest-scoring sentences were picked.

The approaches based on similarity graphs solely model the similarity between pairs of sentences with no clear representation of word relations. Therefore, it is not clear if they adequately cover all topical information. The hypergraph-based approach is to remedy this problem by capturing the high-order relations between both sentences and words. [3] proposed a hypergraph-based model for generic summarization based on [27]'s hypergraph-based model for query-focused summarization. The ranking uses a semi-supervised approach to order sentences. They model the words as vertices and sentences as hyperedges and then approach the problem as a random walk over hyperedges.

9 Conclusion

We showed that frequent patterns can contribute to the quality of text summarization. The comparison supports our statement that the performance of our frequent patterns based model is comparable to the simpler word frequency method and yielded the most relevant sentences of all compared methods. Our methods outperformed other methods in precision but lacked in recall. We attribute the similarity to word frequency method to inadequate graph representation – instead of interconnecting all the words within a sentence, suggest connecting them according to the parse tree. Another consideration is to use a graph mining tool that searches for more specific types of patterns than DGR-Miner.

Acknowledgments

This work has been supported by Faculty of Informatics, Masaryk University. We would like to thank to ITAT reviewers for their suggestions and comments.

References

[1] Charu C. AGGARWAL and ChengXiang ZHAI. *Mining Text Data*. Springer, New York, 2012.

[2] Chinatsu AONE, James GORLINSKY, and Bjornar LARSEN. A trainable summarizer with knowledge acquired from robust nlp techniques. *Advances in Automatic Text Summarization*, 71, 1999.

[3] Abdelghani BELLAACHIA and Mohammed AL-DHELAAN. Multi-document hyperedge-based ranking for text summarization. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1919–1922, New York, 2014 [cit. 2016-05-10]. ACM.

[4] Swapnajit CHAKRABORTI. Multi-document text summarization for competitor intelligence: A methodology based on topic identification and artificial bee colony optimization. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1110–1111, New York, 2015 [cit. 2016-05-10]. ACM.

[5] Swapnajit CHAKRABORTI and Shubhamoy DEY. Product news summarization for competitor intelligence using topic identification and artificial bee colony optimization. In *Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems*, pages 1–6, New York, 2015 [cit. 2016-05-10]. ACM.

[6] John M. CONROY and Dianne P. O'LEARY. Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 406–407, New York, 2001 [cit. 2016-05-10]. ACM.

[7] Harold P. EDMUNDSON. New methods in automatic extracting. *J. ACM [online]*, 16(2):264–285, april 1969 [cit. 2016-05-10].

[8] Günes ERKAN and Dragomir R. RADEV. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res. [online]*, 22(1):457–479, december 2004 [cit. 2016-05-11].

[9] Rafael FERREIRA, Luciano CABRAL, and Rafael D. LINS. Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications [online]*, 40(14):5755 – 5764, october 2013 [cit. 2016-05-10].

[10] Meishan HU, Aixin SUN, and Ee-Peng LIM. Comments-oriented blog summarization by sentence extraction. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pages 901–904, New York, 2007 [cit. 2016-05-10]. ACM.

[11] Meishan HU, Aixin SUN, and Ee-Peng LIM. Comments-oriented document summarization: Understanding documents with readers' feedback. In Sung-Hyon MYAENG, Douglas W. OARD, Fabrizio SEBASTIANI, Tat-Seng CHUA, and Mun-Kew LEONG, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 291–298, New York, 2008 [cit. 2016-05-10]. ACM.

[12] Julian KUPIEC, Jan PEDERSEN, and Francine CHEN. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, New York, 1995 [cit. 2016-05-10]. ACM.

[13] Chin-Yew LIN. Rouge: a package for automatic evaluation of summaries. In Stan SZPAKOWICZ Marie-Francine MOENS, editor, *Workshop Text summarization Branches Out (ACL '04) [online]*, pages 74–81, Barcelona, 2004 [cit. 2016-05-10]. ACL.

[14] Petr MACHOVEC. Automatická sumarizace textu [online]. Master's thesis, Masaryk University, Faculty of Informatics, Brno, 2015 [cit. 2016-05-10].

[15] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[16] George A. MILLER. Wordnet: A lexical database for english. *Commun. ACM [online]*, 38(11):39–41, november 1995 [cit. 2016-05-12].

[17] Smaranda MURESAN, Evelyne TZOUKERMANN, and Judith L. KLAVANS. Combining linguistic and machine learning techniques for email summarization. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning - Volume 7*, pages 19:1–19:8, Stroudsburg, 2001 [cit. 2016-05-10]. Association for Computational Linguistics.

[18] Miles OSBORNE. Using maximum entropy for sentence extraction. In *Proceedings of the ACL'02 Workshop on Automatic Summarization*, Stroudsburg, 2002 [cit. 2016-05-10]. Association for Computational Linguistics.

[19] Kaustubh PATIL and Pavel BRAZDIL. Text summarization: Using centrality in the pathfinder network. *Int. J. Comput. Sci. Inform. Syst [online]*, 2:18–32, 2007 [cit. 2016-05-12].

[20] Roger W. SCHVANEVELDT, editor. *Pathfinder Associative Networks: Studies in Knowledge Organization*. Ablex Publishing Corp., Norwood, 1990.

[21] Roger W. SCHVANEVELDT, D.W. DEARHOLT, and F.T. DURSO. Graph theoretic foundations of pathfinder networks. *Computers & mathematics with applications [online]*, 15(4):337–345, 1988 [cit. 2016-05-11].

[22] Jonas SJÖBERGH. Older versions of the rougeeval summarization evaluation system were easier to fool. *Inf. Process. Manage.*, 43(6):1500–1505, november 2007 [cit. 2016-05-12].

[23] Dr. Aixin SUN. Comments-oriented document summarization, 2015. http://www.ntu.edu.sg/home/axsun/datasets.html, visited 2016-05-20.

[24] Krysta M. SVORE, Lucy VANDERWENDE, and Christopher J.C. BURGES. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 448–457, Prague, 2007 [cit. 2016-05-10]. Association for Computational Linguistics.

[25] Juan-Manuel TORRES-MORENO. *Automatic Text Summarization*. ISTE, London, 2014.

[26] Karel Vaculík and Lubomír Popelínský. Dgrminer: Anomaly detection and explanation in dynamic graphs. In Knobbe-A. Soares C. Papapetrou P. Boström, H., editor, *Advances in Intelligent Data Analysis XV - 15th International Symposium, IDA 2016*, pages 308–319, Neuveden, 2016. Springer.

[27] Wei WANG, Furu WEI, Wenjie LI, and Sujian LI. Hypersum: Hypergraph based semi-supervised sentence ranking for query-oriented summarization. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1855–1858, New York, 2009 [cit. 2016-05-10]. ACM.