MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Text Summarization by Machine Learning

BACHELOR'S THESIS

Matej Gallo

Brno, Spring 2016

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Matej Gallo

Advisor: doc. RNDr. Lubomír Popelínský, Ph.D.

Acknowledgement

I would like to extend my gratitude to my supervisor doc. RNDr. Lubomír Popelínský, Ph.D. for offering valuable advice and suggestions. I would also like to thank RNDr. Karel Vaculík for all the help with R programming language and insights into DGRMiner tool.

Abstract

This thesis discusses the state-of-the-art algorithms, focusing only on those that use machine learning approach, such as neural network based NetSum, or graph representation, such as LexRank. We also present the means of evaluating summaries using ROUGE metrics. A part of the work is experimental implementation of a method that takes advantage of frequent patterns extracted from text. We evaluate and compare the performance of our methods to traditional methods found in automatic text summarization.

Keywords

extractive summarization, machine learning, graph mining, dynamic graphs, frequent patterns

Contents

List of Tables

List of Figures

1 Introduction

Recently, the amount of documents published on the Internet has reached such tremendous magnitude that simply skim-reading them manually would be rather a time-consuming task. Automatic text summarization is actually the method that (TS) enables to create short and concise summaries of a single or a set of documents.

There are two approaches to TS in computational linguistics: extractive and abstractive. While extractive approaches select a subset of sentences from the source document based on statistical and linguistic characteristics such as keywords, sentence position, and frequency, abstractive approaches understand texts as a whole and subsequently generate their accurate and succinct abstracts by means of natural language generation techniques. The latter approach focuses on sentence flow and vague terms minimization [1–4].

Methods based on graph theory and machine learning are thriving. According to this concept, each document that is subject to analysis is projected as a graph, where vertices are assimilated into sentences and edges represent intra-sentence similarity. The similarity is defined as topic overlap, a situation when words are present in both sentences. To the resulting weighted graph, a scoring algorithm is applied, of which the most commonly used ones are LexRank and TextRank. The chosen algorithm assigns the most significant vertices that are used in the final summary [1–4]. The authors of [5] introduced a novel idea of representing the document as a hypergraph. The authors of [6] introduced and compared the effectiveness of their new system SumGraph to traditional graph-based methods with promising results.

Machine learning methods create models based on the characteristics of the extracted text in order to classify the sentences of the text as either being part of a summary or not, or assigning them a real value. The higher the value, the higher the importance of the sentence. Amongst the most successful methods belong naïve Bayes classifier, AdaBoost and KNN (k-nearest neighbors). The features that best indicate whether a sentence is a part of a summary or not, are sentence centrality, sentence position, and named entities [7].

The goal of this thesis is to provide an overview of the state-of-the-art graph-based and machine learning methods for the task of

TS. We also aim to implement our own method based on mining frequent patterns from dynamic graphs using the graph mining tool – DGRMiner. The implementation will be written in R programming language, which is often used for statistical computations and it is currently the most used language in machine learning. We will evaluate the effectiveness of this approach to 17 summarizers using word frequency, similarity, position and other features. To test the systems we will use a blog summarization dataset, consisting of 100 blogs and ROUGE-1 metric commonly used for evaluating TS and machine learning tools.

Two of our models outperformed all other compared methods in terms of precision and the number of correctly chosen sentences. The results also showed that the extracted frequent patterns corresponded to words with more than one appearance within text. And thus, they performed similarly to methods based on word frequency. More complex patterns did not consistently improve the performance of our summarizer. We attribute this fact to improper graph representation or the generality of extracted patterns.

2 Preliminaries

In this chapter, we define the main terms related to natural language processing (NLP) theory that will appear throughout our work.

Not only in TS, but also in NLP a **corpus** represents the source of knowledge about a specific language. Namely, it is a large set of structured collection of documents annotated with linguistic markers such as part of speech (POS), and metainformation such as position of words in the sentence and position of sentences in a document. Corpora are used to analyze languages as well as to verify linguistic theories. Nowadays we often use corpus to train machine learning algorithms. They are capable of extracting the necessary attributes and statistical data for a particular task. Some literature has misused the terminology and refers to corpus as simple collections of unstructured documents [1].

The quality of a summary is closely linked to cohesion and coherence. According to [1], cohesion is located at the linguistic level of the sentence. The use of semantically related terms (Brno–city), co-reference (Jane–she), ellipsis ("I can summarize the text; you can summarize the text too.") and conjuctions ("and", "but") helps to improve the cohesion of a text. Coherence, on the other hand, is located at the higher level of semantics. To produce a coherent text, we must avoid contradictions and/or redundancy in the text. Currently, there is no effective way how to evaluate cohesion and coherence in summaries and the task remains an open problem.

Unresolved **anaphoric references** are another common problem in text summarization. The term anaphora describes a linguistic phenomenom when a text unit derives its meaning from a previously mentioned part of the text (the antecedent). Some literature states that anaphora can also derive its meaning from succeeding parts of text. The process of matching anaphora with its corresponding antecedent is called **anaphora resolution**. In a sentence *I read a book. It was interesting*, the anaphoric pronoun *it* refers to *book* [1, 8, 9].

A **co-reference** is a linguistic phenomenon when two or more lexical units share a reference. In the sentence *I wish I hadn't listened* the pronouns *I* refer to the same referent in real world; they are co-referential. However, in *He wishes he hadn't listened* the pronouns don't necessarily have to refer to the same person [1, 8, 9].

3 Types of Summaries

Automatic summarization systems construct short summaries, that retains the most important ideas of the original text. Each of the summaries can be described by three principles [10]: the summary should be produced from single or multiple documents, it should preserve important information, and it should be short. The community disagree on any attempt to make a formal definition [1].

Summaries are categorized according to their function, number of documents for summarization, genre of document, type of summary, type of summarizer, context, target audience, and supported media types [1, 2, 11].

Extractive summarization is the most common approach to TS problem [2], which focuses on the very nature of the summary content. The system 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. This results in a lack of cohesion or coherence, unresolved co-references or discourse relations. In chapter 4 we constrain ourselves to extractive systems [1, 2, 4].

Abstractive summarization tries to understand the text (just like humans) by identifying the key concepts and then generates a summary (abstract) conveying the same information. Compared with the extractive approach the goal is to generate a grammatical summary while focusing on the form. The use a language model increases the level of understanding by employing strategies to improve coherence. In practice, however, the poor performance of existing natural language text generators affects the quality of generated sentences. For this reason, extractive summarization is preferred. Abstractive summarization is closely related to paraphrasing. Applying sentence fusion or compression on extract can be considered a first step to forming a more comprehensible abstract. [1–3].

The first attempts at automatic summary dealt with **single-document summarization**, where the system received one document only on input. Later on, researchers moved on to **multi-document summarization**. The latter approach was inspired by the Internet due to high information redundancy. The system receives multiple documents on input and since there is a possibility of choosing redundant sentences (each from different input), it must be able to detect sentences of the same topic (via clustering or similarity measure). A major issue is coherence as the sentences might contain conflicting information, especially if the documents are from different periods [1, 2, 10]. Recently, there has been a decline in studies on automatic single-document summarization, partly because the task might be counter-intuitively more difficult than multi-document summarization itself [12].

We term a summary **indicative** if it determines the main topics (resembles a table of contents). It includes metadata such as length of a document and writing style, but fails to provide factual information. It reduces the length of the original document by 90%. On the other hand, an **informative summary** also includes facts and information content and is generally longer – reduces the length by 70-80%. While informative summary can replace the source text, indicative summary can help us decide whether we want to read the document. The **evaluative summary** captures the opinion of the author on a given subject (review) [1–3].

By limiting the length of the final summary to a single sentence, we obtain a **headline summarization**. **Ultra-summarization** lies in the process of summarizing short texts, which is useful for mobile devices and platforms with limited screen space. Whereas, a **keyword summary** produces indicative words or phrases from the input [1, 3].

Generic summarization is the most common type of summary aimed towards a general audience. The system is oblivious to the genre or the domain of the document. The importance of a sentence is determined only from the input document alone. This kind of summary can provide enough information, possibly enabling the reader to avoid reading the entire document. Jones [13] argues that generic summarization is unnecessary and should always keep the purpose, intended reader and genre in mind [3].

Users provide additional information relevant to their interest along with the set of documents to obtain a **query-focused summary** (or **user-focused**, or **topic-focused** [11]). The query can be a keyword, phrase or an open-ended question. This type of summary can be especially helpful in search engines to produce snippets for suggested websites. The automatic summarizer takes the query as well as the suggested website on input and either returns relevant information or indicates how much the document relates to the query. A study showed that query-focused summaries require a combination of features or indicators to produce a quality summary [1, 11, 14].

An **update summary** is a multi-document summarization that can omit parts of information that the user is already familiar with. It can be used to track the development of events over time [1, 3].

In terms of input and output language, we differentiate between **monolingual**, **multilingual** and **cross-lingual** summarizers. While monolingual summarizers process one language with both input and output in the same language, multilingual summarizers process several languages again with both input and output in the same language. Cross-lingual summarizers process several languages, while the output is in different language from input. **Sublanguage summaries** have restricted vocabulary that is oriented towards a specific audience – non-native speakers or children [11].

According to the genre of document, we classify summaries as **news summaries**, **specialized summaries** (specialized domain such as law or science), **literary** (narrative documents), **encyclopedic** (Wikipedia) and **social networks** (blogs, forum posts and tweets) [1].

According to the type of summarizer, we classify summaries as **author summary** (reflects the author's point of view), **expert summary** (not author – expert in the field of the topic discussed, but not skilled at producing summaries) and **professional summary** (written by a professional summarizer who is not necessarily expert in the field of the topic) [1].

Tables, pictures and diagrams provide an efficient way of summarizing large amount of data. In **multimedia summarization** the input and/or output consist of a combination of different media types, including audio or video [1].

Currently, the scientific community is focusing mainly on researching multi-document (mono- or multilingual) summaries guided by a query, update summaries, and summaries of specialized domains [1].

4 Extractive Summarization Techniques

In this section we restrict ourselves to monolingual extractive summarization. We briefly introduce the basic approaches and describe state-of-the-art machine learning and graph-based approaches in greater detail.

Before a text can be summarized, it is necessary to pre-process it. Pre-processing is a form of dimensionality reduction by removing noise (e.g. uninformative words or conjugation). As a result of this reduction, the vectorization of documents becomes easy to manage. The pre-processing phase often involves: splitting the text into segments (phrases, sentences, paragraphs); splitting segments into words (tokenization); word normalization (lemmatization, stemming); stopword filtering; POS tagging; named entity recognition; extraction of terms and keywords; weighting terms (which depends on representation) [1, 15]. All of these steps are language-dependent (difficulty varies with language). If the language is unknown, we can identify it using a language identification module such as TreeTagger.

Virtually all summarizers perform three tasks: creating intermediate representation of the input, scoring sentences, and selecting a summary [4]. Figure 4.1 illustrates these tasks.

Intermediate representation is performed in order to identify key content. Topic representation approaches convert the text into a set of topics discussed in it, while the topics may be represented in the form of tables of words with corresponding weights, lexical chains, latent semantic analysis (LSA) or Bayesian topic models. The most widely used representation is called **the vector space model (VSM)** as defined by 4.1 in [1].

Definition 4.1. Vector Model *Let* $\{t_1, t_2, \ldots, , t_k\}$ *be the set of terms and* $\{D_1, D_2, \ldots, D_N\}$ *the set of documents. A document* D_i *is represented as a vector:*

$$\vec{D}_i = \langle w(D_i, t_1), w(D_i, t_2), \ldots, w(D_i, t_k) \rangle$$

where $w(D_i, t_j)$ *is the weight of a term* t_j *in the document* D_i*.*

By disregarding the word order, we obtain a **bag-of-words model** where every word is weighted by its importance in the document. There are three types of weighting $\omega_{i,j}$: binary (equals 1 if the term is

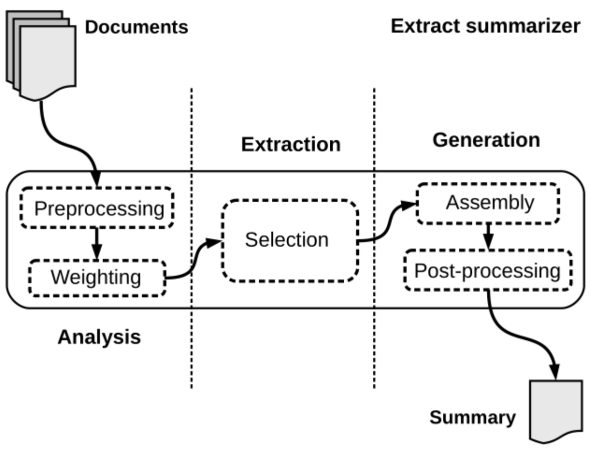

Figure 4.1: General architecture of an extraction-based single-document summarization system [1]

present in the sentence and 0 if not), frequency (number of occurrences of a term in a sentence) and corrective (frequency normalized by word distribution – TF*IDF). The result is a $S_{[\rho \times N]}$ matrix of ρ rows (sentences) and N columns (terms in the lexicon). Each sentence $\vec{s}_\mu$ is projected as a vector of terms weighted by their prominence in it.

$$S = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,N} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{\rho,1} & s_{\rho,2} & \cdots & s_{\rho,N} \end{pmatrix} ; \; s_{\mu,j} = \begin{cases} \omega_{\mu,j} & \text{when the word } j \in \vec{s}_\mu \\ 0 & \text{otherwise} \end{cases}$$

where each μ row contains the $\omega_{\mu,j}$ weighting of the word j in a $\vec{s}_\mu$ sentence [1].

Indicator representation takes advantage of indicators of importance such as sentence length, proper nouns, inclusion of numerical data in a sentence, location of a sentence in the document, presence of words in the title and cue phrases (a list of specific phrases marking a possibly significant sentence). Studies show that the most important indicator is position of sentence in the document and that the combination of cue words, title words and position performs better than the distribution of the frequency of keywords alone [1, 4].

In **graph representation**, the vertices (or nodes) represent units of text (words, sentences or documents) and the edges represent the relations between these units. We can express this relation either by a cosine similarity or word overlap. Instead of assigning weights to the edges we can express the relation in binary – vertices are connected if their similarity exceeds a preset threshold [1, 4].

A graph G can be directed forward, directed backward or undirected. In directed forward (backward) graph a sentence only recommends the sentences which succeed (precede) it in the text. The most recommended sentences are then extracted to form a summary. Forward graphs are suitable for film and fiction criticism, while backward graphs are more suited to journalistic articles. In undirected graphs a sentence can recommend any of the two sentences [1].

Given a particular representation, the TS system **scores** the **sentences**. For topic representation the score is related to the ability of the sentence to express the main topics. For the majority of indicator representation methods, machine learning is applied to combine the indicators and gain scores. In graph representation the weights are obtained by applying stochastic techniques. Most commonly, a stationary distribution is calculated by applying iterative random walk algorithm and the resulting probabilities corresponds to the importance of each vertex [1, 3, 4].

Lastly, the summarizer **selects summary sentences**. The summary is usually constrained by its length. In the *best n* approach, the top n highest scoring sentences are selected. To avoid redundancy we can penalize sentences that are similar to the sentences already chosen for summary. Maximal marginal relevance (MMR), as given by formula 4.1, maximizes relevant information and minimizes redundancy. Every time a new sentence is selected, the scores are recalculated as a linear combination of the original weight and its similarity with already chosen sentences.

$$\omega_{\text{MMR}}(s) = \arg\max_{s \in D \setminus \text{Sum}} [\lambda \underbrace{\text{sim}_1(s, Q)}_{\text{Relevance}} - (1-\lambda) \underbrace{\arg\max_{s_\mu \in \text{Sum}} \text{sim}_2(s, s_\mu)}_{\text{Redundancy}}] \tag{4.1}$$

where D is the set of sentences in the document, *Sum* is the set of sentences already selected for the summary, Q is a query (topic of the summary), λ is a penalization coefficient and $\text{sim}_i()$ is a function which returns the similarity of two sentences [1, 4].

In *global selection* approaches, the subset of sentences is subjected to constraints to maximize overall importance, minimize redundancy and maximize coherence [4].

According to [1, 11, 16] we can classify summarization algorithms in terms of understanding of text and complexity in text processing. Surface level (shallow) approaches are based mostly on term frequency and lexical semantics. The deep parsing approaches exploit discourse relations in sentence and assume at least a sentential semantics level of representation.

4.1 Early work

In the earliest works on automatic summarization Luhn [17] proposed the idea of using frequency and position of relevant words to measure the significance factor of a sentence. The idea was based on the fact that authors frequently repeat those words that they want to emphasize. The words were reduced to their word stem and because Luhn noticed that not all frequent words are relevant, he suggested the use of stopword filter. The remaining words were sorted in descending order of frequency. The significance factor reflects the occurrence of significant words in a sentence as well as the distance between them. All sentences are ranked based on this factor and the most significant ones are chosen for the summary.

In his work, Edmundson [18] described an extractive summarizer. He used the features from the previously mentioned paper and included two new features: cue phrases, and title (words appearing in title or headings had higher weight).

In contrast, Vanderwende's SumBasic [19] system uses true initial probabilities to compute the weights for relevant words. The sentence weight was computed as a weighted average of all significant words in a sentence. She also proposed the idea of modifying the probabilities of words in the already chosen sentences. According to [4] the use of raw frequencies is strongly influenced by document length.

The main issue with all the above methods is the reliance on stopwords. It is rather complicated to decide whether a word should be included in the stopword list or not. For this reason, two new methods were introduced: TF*IDF weighting and log-likelihood ratio (LLR) test [4]. Both methods were based on the idea that relevant words appear more frequently in the document to be summarized than in other documents. Compared to TF*IDF, the LLR test provides a way of setting a threshold to divide words into either significant ones or not.

4.2 Naïve Bayes

Kupiec et al. introduce in their work [20] method inspired by Edmundson's [18]. 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. The training set consisted of hand-selected summaries. 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 $u_1 = 5$, $\text{length}(s) > u_1$), 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. The minimum sentence length u_1 was empirically determined. Assuming the statistical independence of the features, the probability that a sentence s will be included in summary S given the k features $F_i; i = 1, 2, \ldots, k$ is calculated according to the formula 4.2.

$$P\left(s \in S | F_1, F_2, \ldots, F_k\right) = \frac{\prod_{i=1}^{k} P\left(F_i | s \in S\right) \cdot P\left(s \in S\right)}{\prod_{i=1}^{k} P\left(F_i\right)} \tag{4.2}$$

$p(s \in \text{Sum})$ is a constant, and the probabilities $p\left(F_i | s \in \text{Sum}\right)$ and $p\left(F_i\right)$ can be estimated from a training corpus.

The sentences received a score equal to their probability of being a part of summary and top n most likely sentences were extracted.

Aone et al. [16] built on Kupiec's work [20] and expanded the feature set of their system, called DimSum, with signature terms, which indicate key concepts for a given document. Another advantage over [20]'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 [21] 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.

4.3 Maximum Entropy

Osborne in his work [22] 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. The conditional log-linear model can be computed by formula 4.3

$$P\left(c \mid s\right) = \frac{1}{Z(s)} \exp\left(\sum_i \omega_i f_i(c,s)\right) \tag{4.3}$$

where $Z(s) = \sum_c \exp\left(\sum_i \lambda_i f_i(c,s)\right)$. The weights λ_i are trained using conjugate-gradient descent. Non-uniform priors were added because the model tends to over-select the "reject" label. Osborne uses the same prior in a naïve Bayes model. The classification of sentences was determined by formula 4.4

$$label(s) = \arg\max_{c \in C} P(c) \cdot P(s,c) \tag{4.4}$$

Osborne turned the task of selecting priors into an optimization task. The freedom of choosing any optimization function enabled him to control the trade-off between precision and recall. The proposed optimization function was a f_2 score, where $f_2 = \frac{2pr}{p+r}$. The feature set consisted of the following features: word pairs, sentence length, sentence position, shallow discourse features.

4.4 Hidden Markov Models

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 [23] 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.

The HMM structure, as presented in [23], had $2s + 1$ states, with s summary states and $s + 1$ non-summary states. Hesitation was allowed only in non-summary states, while skipping of states was allowed only in summary states. Depending on how the last two states are connected either $s - 1$ lead summary sentences and arbitrary many supporting sentences are retrieved; or exactly s summary sentences are retrieved by eliminating the cycle between the last two states. Both the transition matrix and initial distribution must be estimated using the training data.

The probability that sentence s corresponds to state i is given by $\gamma_s(i)$. Computing the probability that a sentence s is a summary sentence can be achieved by summing over all even i's (summary states) of $\gamma_s(i)$. To make the best summary n sentences with highest probability of being part of summary are selected [3, 23].

4.5 Artificial Neural Network

Summarizer NetSum presented by Svore et al. [12] 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 NetSum system is a two-layer neural network trained using RankNet, sped up using the framework of LambdaRank. The author tested the number of hidden nodes between 5 and 15, with error rate between 10^{-2} and 10^{-7}. Every sample consisted of label and feature vector. Labels corresponded to a ROUGE-1 score between a given sentence and the reference summary. The feature vector consisted of 10 features: is first sentence, sentence position, SumBasic score, SumBasic bigram score, title similarity score, average news query term score, news query term sum score, relative news query term sum score, average Wikipedia entity score, Wikipedia entity sum score.

The pre-processing step consisted of lowercasing every word and removing the punctuation. Stemming was ineffective. Wikipedia entities (titles) were disambiguated if they had landed on the same page.

The system significantly outperforms the standard baseline in the ROUGE-1 measure. No past system could outperform the baseline with statistical significance.

Enertex is a summarization system using ANN based on statistical physics – textual energy [24]. It's based on paradigms which are markedly different to Natural Language Processing [1]. The document is modelled as a neural network from which a textual energy is deduced. An algorithm is used in which the spin stages represent the positive or negative polarity of the words. This enables semantic orientation to be extracted to classify either positive or negative opinions in a corpus.

A magnetic system is composed of a set of N small magnets called spins (quantum-mechanical phenomenon). The simplest case, the

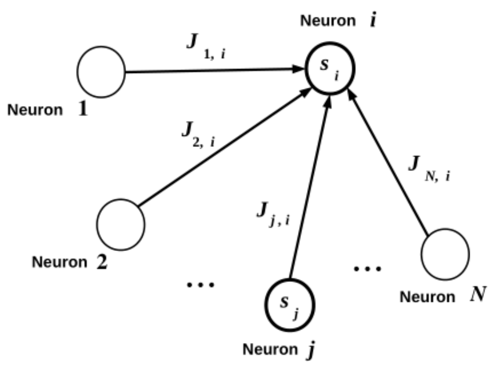

Figure 4.2: Field received by the neuron s_i, induced by the other $N-1$ neurons. $J_{i,j}$ represent synaptic weights. [1]

Ising model, is used by a variety of systems, which can be described by binary variables. A system of N binary units (neurons) can store $\rho = 1, \ldots, 2^N$ possible configurations (or patterns).

The spins correspond to neurons, which are linked by synaptic weights and which interact according to Hebb's learning rule. The learning rule transforms the configurations into attractors (local minimums) of the energy function. Energy varies according to the configuration of the system (activation states or inhibition of neurons).

Presenting a pattern to the network, each neuron receives a local field induced by the other neurons. This process is illustrated in figure 4.2. The neurons align themselves according to their field to retrieve the closest pattern associated with the presented pattern.

The document is first transformed into a VSM bag-of-words representation that produces a $S_{[\rho \times N]}$ matrix composed of ρ sentences (patterns or rows) and a lexicon of N terms (spins). The matrix can be defined as a set of ρ configurations of system, where it is possible to calculate energy. A sentence is therefore a chain of N spins [1].

In Enertex, the text undergoes standard pre-processing: stopword filtering, normalization of words (lemmatization and stemming) and representation of text in VSM. Then second module applies the spin model to compute the textual energy matrix from which a weight of a sentence (absolute energy) is calculated. The process is applied to all sentences and we receive a sorted list. Sentences with highest absolute energy are considered relevant sentences.

4.6 K-Means

A system for generating product category-based (topic-based) extractive summarization was proposed by [25, 26]. 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 [26] 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.

4.7 Decision Trees

In Muresan's paper [27], 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 algorithms 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 [27]. The filtering consisted of four steps: grouping inflectional variants, removing unimportant modifiers, filtering stopwords, and removing empty nouns.

4.8 LexRank

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 by formula 4.5 until the values of the vertices have not been modified by more than $\epsilon = 0.0001$. LexRank algorithm is used as a component of the MEAD [28].

$$\omega(s_i) = \frac{d}{N} + (1-d) \sum_{s_j \in In(s_i)} \frac{\mathrm{sim}(s_i, s_j)}{\sum_{s_k \in \mathrm{Out}(s_j)} \mathrm{sim}(s_k, s_j)} \omega(s_j) \tag{4.5}$$

where sim() is the cosine weighted by inverse document frequency, the damping factor $d = 0.85$ and N is the number of vertices [1].

4.9 TextRank

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 by formula 4.6.

$$\omega(s_i) = (1-d) + d \sum_{s_j \in In(s_i)} \frac{w_{j,i}}{\sum_{s_k \in \mathrm{Out}(s_j)} w_{j,k}} \omega(s_j) \tag{4.6}$$

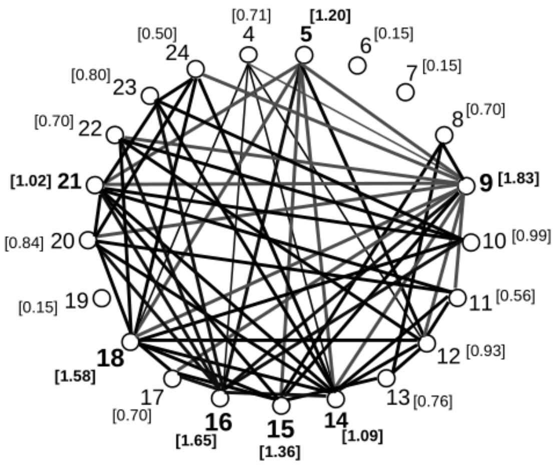

Figure 4.3: Example of a TextRank graph. The heaviest sentences are 9, 16, 18, 15, 5, 14 and 21 [1]

where the damping factor $d = 0.85$ and the weight $w_{j,i}$ between two sentences i and j is defined by formula 4.7

$$w_{j,i} = \frac{\sum_{w \in i,j} C_w^i + C_w^j}{\log |s_i| + \log |s_j|} \tag{4.7}$$

where C_w^i represents the occurrences of the word w in the sentence i [1]. An example of a TextRank graph is shown in figure 4.3.

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 [4] 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 [1].

4.10 SumGraph

Patil [6] 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) [29, 30] 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.

4.11 Hypergraph

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. [5] proposed a hypergraph-based model for generic summarization based on [31]'s hypergraph-based model for query-focused summarization. Figure 4.4 shows an example of hypergraph. 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.

A hypergraph is a generalization of a graph that relaxed the condition of edges being pair-wise and makes high-order relation explicitly represented. The hyperedges are weighted using density of topic sig-

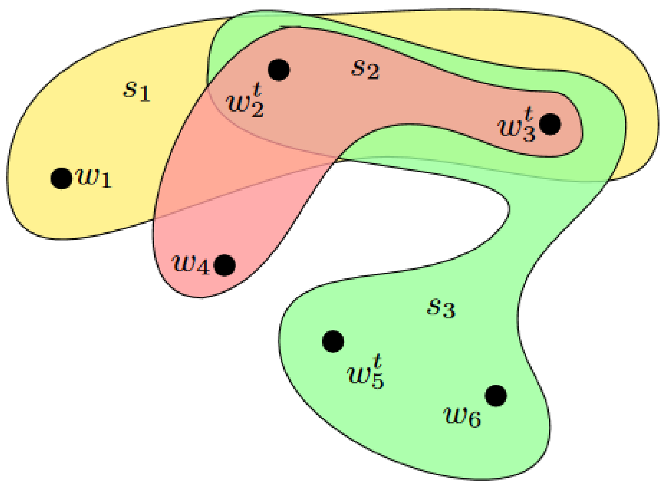

Figure 4.4: An example of hypergraph. s represents sentences, w represents words, and w_t represents topical words [5]

natures (log-likelihood ratio test) over total words in a sentence to favor descriptive sentences over long sentences given by formula 4.8:

$$w(e_i) = \frac{1 + \sum_{w \in e_i} R(w)}{\delta(e_i)^2} \tag{4.8}$$

where $\sum_{w \in e_i} R(w)$ is the number of topic signature words in sentence e_i, and $\delta(e_i)^2$ is the square of the length of the sentence. .

Traditionally, random walks are defined over vertices while transitioning is performed over edges. Since we rank hyperedges instead of vertices, a new random walk is used. It transitions between hyperedges by starting at a given hyperedge e_i, uniformly choosing vertex v and then choosing hyperedge e_j proportionally to hyperedge weight, and moves to that neighboring hyperedge.

After the sentences are ranked, they are ordered in descending order and the most central sentences (highest score) are choosen to be included in the summary. Maximal margin relevance is applied to eliminate redundancy.

5 Evaluating Document Summaries

Evaluating the quality of automatically produced summaries is subjective (there is no "perfect" summary), rather difficult and remains an open problem. Evaluation can be divided into two categories: extrinsic and intrinsic. Extrinsic methods evaluate summaries' performance related to its contribution to a specific task that the evaluation is dependent on (combining the results of summarizer with Question-Answer system). However, they are time-consuming, expensive and require careful planning.

Intrinsic evaluation involves human judges who either rate the summaries or create reference summaries. Intrinsic evaluation can be conducted manually, semi-automatically or automatically. The first two approaches rely on automatic summaries read and evaluated by human judges; and on reference summaries produced by human beings, respectively. This is, however, a major limitation due to subjectivity. Therefore, the final summaries will vary greatly in terms of content. A third new approach – automatic intrinsic methods – has emerged and is gradually increasing in popularity. The summary is evaluated in relation to the source document. Statistical diverge measures (Jensen-Shannon or Kullback-Leibler) applied to the probability distribution are prioritized via this approach [1].

To conduct an intrinsic evaluation of automatic summaries, a baseline is set out. The definition of baseline summary from [1]:

Definition 5.1. BASELINE SUMMARY *The baseline method carries out an elementary task of summarization. It enables systems to compare their results to the base results. It can be defined in several ways because the baseline depends on the task.*

The regular baseline methods to evaluate summaries are: random sentences, lead sentences, lead-end sentences, and lead words/chars. Despite the simplicity of the methods, these baselines are very difficult to beat. So much, that single-summarization task was dropped from the DUC Evaluation Workshops altogether [3]. In query-oriented summarization tasks, the summarization system MEAD was chosen as a baseline for DUC evaluations [1].

5.1 Manual evaluation

The human judges read through the generated summaries and score each one of them. Form of the summaries is scored based on linguistic criteria established by NIST: grammaticality, non-redundancy, referential clarity, focus, structure, and coherence. Content of the summaries is scored based on its content responsiveness and overall content quality. Scores are assigned according to a qualitative five-point scale (1 – lowest, 5 – human like). Manual evaluation is a good compromise for judging both the form and the content of summaries and is the key to the success of DUC/TAC workshops [1].

5.2 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 the Rouge method only [1, 2].

5.2.1 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 5.1.

Given its correlation with manual judgments Rouge almost seems to have become a standard. [32] reports Pearson coefficient for the most commonly used variations (Rouge-2 and Rouge-SU4 [2]) at a value of between 0.94 and 0.99. Generally, the Rouge-n is calculated from

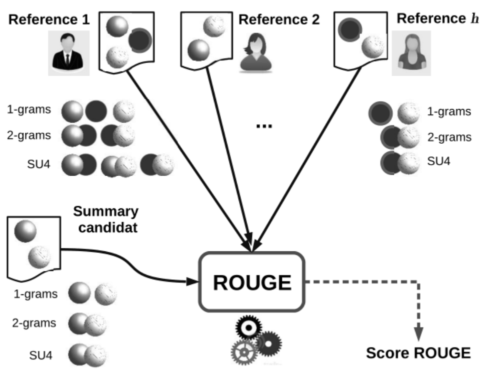

Figure 5.1: The basic idea of ROUGE for evaluation of summaries [1]

the co-occurrences of n-grams between the candidate and reference summaries as shown by formula 5.1 in [1]:

$$\text{Rouge-}n = \frac{\sum_{n\text{-grams}} \in \{\text{Sum}_{\text{can}} \cap \text{Sum}_{\text{ref}}\}}{\sum_{n\text{-grams}} \in \text{Sum}_{\text{ref}}} \tag{5.1}$$

where the numerator is the maximum number of co-occurrences of n-grams in both reference and candidate 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 5.2:

$$\text{Count}(k, n) = C\binom{n}{k} - \sum_{0}^{k-\gamma}(k-\gamma); \gamma \leq 1 \tag{5.2}$$

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 [33] found that the system can be tricked into generating a summary with high Rouge score.

6 Summarization Based on Frequent Patterns

As a part of this thesis we introduce a new method for single-document extractive summarization in English language. This method is based on the principle of mining frequent patterns from a text that is represented as a dynamic graph and using the resulting patterns as indicators of sentence importance.

6.1 DGRMiner

Data mining is an automated process that facilitates the discovery of data patterns. As described in chapter 4, graph representation is a common method for representing data in automatic text summarization tasks. We used DGRMiner tool to extract these patterns from the dynamic graphs. DGRMiner was proposed in [34] for mining frequent patterns that capture various changes in dynamic graphs in the form of predictive rules. It can be applied on both single dynamic graph and a set of dynamic graphs. 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. Therefore, a relative timestamp equals to 0 means current change and a timestamp of $-t$ denotes a change of t snapshots earlier. 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.

6.2 Dataset

To assess the performance of our method we used the Blog summarization dataset [35–37]. 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 [38] 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.

6.3 coreNLP

An R implementation of Stanford CoreNLP java library was used. 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. CoreNLP is highly flexible and extensible integrated framework. A CoreNLP pipeline can be run on a piece of plain text. For our paper, we used 4 annotators: sentence split, tokenize, lemma, and POS. CoreNLP supports English, Chinese, French, German and Spanish, although some tools might be available for certain languages only. Interfaces for running CoreNLP from other programming languages are also available in .NET, Perl, Python, R, Scala etc.

6.4 Text Pre-processing

In this step we pre-processed the text to create an input data for DGR-Miner. 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.

6.5 Pattern Mining

In this step we applied the DGRMiner on the data from previous step to obtain patterns. 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. No reliable way of determining the correct value of support was found. If the support parameter is set too low, the DGRMiner will consider a pattern anything that appeared at least once in the text – every word, every possible word combination. This results in an almost infinite run of the tool. By setting the confidence parameter in DGRMiner to 0, the DGRMiner assigned confidence (as discussed in section 6.1) to each extracted pattern. The last available parameter was time abstraction, which allowed us to ignore the preset context *c* as discussed in section 6.4. 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.

6.6 Sentence Scoring

In this step we used the observed patterns as indicators to score the sentences. The final score was obtained according to formula 6.1 as a sum of single-vertex and multi-vertex scores:

$$\text{Score}(s) = \text{Score}_{single}(s) + \text{Score}_{multi}(s) \tag{6.1}$$

Three different metrics were implemented to calculate single-vertex score. The Jaccard coefficient, as given by formula 6.2, expresses the overlap of two sets. In our case, between the set of patterns P and the set of words in the sentence S.

$$\text{Score}_J(s) = \frac{w_{sum}(S \cap P)}{w_{sum}(S \cup P)} \tag{6.2}$$

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, as given by formula 6.3, is the simplest of the three. For a sentence s and a set of patterns P the score is calculated as weighted average.

$$\text{Score}_F(s) = \sum_{p \in P} c(p)w(p) \tag{6.3}$$

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

The density method, as given by formula 6.4, counts the patterns in a sentence and normalizes by the length of the sentence. This method is parameter-free.

$$\text{Score}_D(s) = \frac{\mid S \cap P \mid}{\text{length}(s)} \tag{6.4}$$

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 *lambda*. We tuned the parameter on 10% of the entire dataset. The optimal values for λ are presented in the tables 6.2 and 6.1. The ordering of sentences in the final summary maintains the relative ordering in the original document.

6.7 Results

The FRQ method marked the most accurate sentences among all the presented algorithms as can be seen from figure 6.1. 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.

We evaluated the model using the ROUGE-1 metric, which is recommended for short summaries [2]. Every blog post was compared against four reference summaries as described in chapter 6.2. The results of all three methods can be seen in tables 6.1, 6.2 and 6.3. 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 frequency-based method incorporating patterns with no confidence weighting (identified as FRQ) achieved the highest recall, the highest f1 score, and the highest 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 [38]. From table 6.4 we can 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 6.5. The multi-vertex patterns improved the performance of the summarizer in density models and in one case (the highest scoring model) in frequency model.

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 6.1: 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 6.2: 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 6.3: Results for blog summarization dataset using density method

algorithm	precision	recall	f1 score
FRQ	**0.65**	**0.70**	**0.67**
DEN	**0.65**	**0.51**	**0.56**
Sentence centrality 1 (SC1)	0.65	0.50	0.56
JAC	**0.64**	**0.69**	**0.66**
Resemblance to the title (TR)	0.64	0.60	0.61
Cue-phrase (CP)	0.64	0.52	0.57
TF*IDF	0.63	0.75	0.68
Word frequency (WF)	0.63	0.72	0.67
Lexical similarity (LS)	0.63	0.71	0.66
TextRank score (TS)	0.63	0.68	0.65
Word co-occurrence (WC)	0.63	0.65	0.63
Bushy path (BP)	0.63	0.58	0.60
Proper noun (PN)	0.63	0.57	0.59
Inclusion of numerical data (ND)	0.63	0.54	0.58
Sentence position 2 (SP2)	0.63	0.52	0.56
Sentence length (SL)	0.62	0.76	0.68
Aggregate similarity (AS)	0.62	0.58	0.59
Upper case (UC)	0.61	0.58	0.59
Sentence position 1 (SP1)	0.60	0.46	0.51
Sentence centrality 2 (SC2)	0.60	0.46	0.51

Table 6.4: Comparison of results of ROUGE on blog summarization dataset.

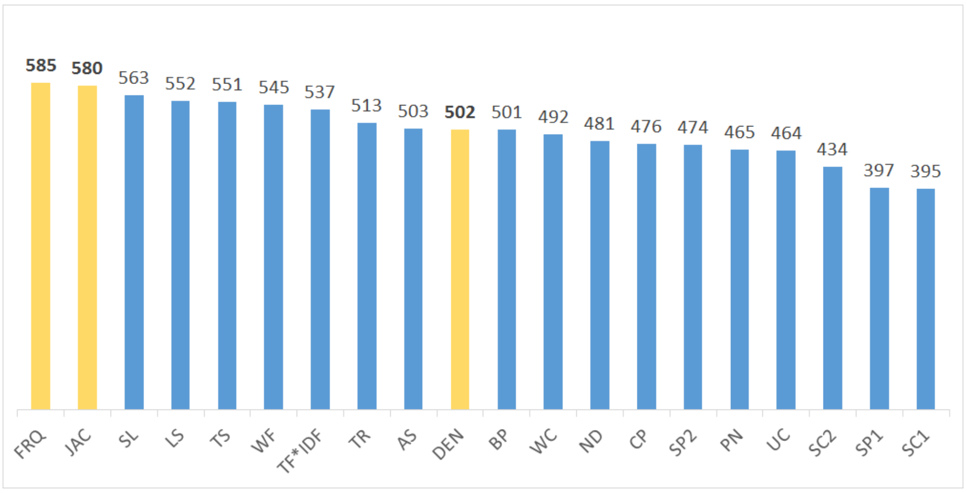

Figure 6.1: Comparison of number of correctly chosen sentences – using blog summarization dataset (our algorithms are displayed yellow)

7 Conclusion

In the first chapter we defined what a summary is and what types of summaries can be created. In the second chapter we discussed state-of-the-art machine learning and graph-based algorithms for solving the task of TS. In the last chapter, we introduced and compared our new method that is based on frequent patterns.

We showed that frequent patterns can contribute to the quality of TS. 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 DGRMiner.

Bibliography

[1] Juan-Manuel TORRES-MORENO. *Automatic Text Summarization*. London: ISTE, 2014. ISBN: 978-1-84821-668-6.

[2] Petr MACHOVEC. "Automatická sumarizace textu [online]". Master's thesis. Masaryk University, Faculty of Informatics, Brno, 2015 [cit. 2016-05-10]. URL: `http://is.muni.cz/th/359331/fi_m/`.

[3] Ani NENKOVA. "Automatic Summarization". In: *Foundations and Trends in Information Retrieval* 5 (2 2011).

[4] Charu C. AGGARWAL and ChengXiang ZHAI. *Mining Text Data*. New York: Springer, 2012, pp. 43–76. ISBN: 978-1-4614-3222-7.

[5] 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*. New York: ACM, 2014 [cit. 2016-05-10], pp. 1919–1922. ISBN: 978-1-4503-2598-1. URL: `http://doi.acm.org/10.1145/2661829.2662036`.

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

[7] Gabriel SILVA et al. "Automatic Text Document Summarization Based on Machine Learning". In: *Proceedings of the 2015 ACM Symposium on Document Engineering [online]*. New York: ACM, 2015 [cit. 2016-05-10], pp. 191–194. ISBN: 978-1-4503-3307-8. DOI: `10.1145/2682571.2797099`.

[8] David CRYSTAL. *A dictionary of linguistics and phonetics*. Sixth edition. Malden: Blackwell, 2008. ISBN: 978-1-405-15296-9.

[9] Ruslan MITKOV et al. "Coreference and anaphora: developing annotating tools, annotated resources and annotation strategies". In: *Proceedings of the Discourse, Anaphora and Reference Resolution Conference (DAARC2000)*. Ed. by Paul BAKER et al. Lancaster: Lancaster Univ., 2000 [cit. 2016-05-10], pp. 49–58. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.486.9249&rep=rep1&type=pdf`.

[10] Dipanjan DAS and André F. T. MARTINS. *A Survey on Automatic Text Summarization*. Tech. rep. Carnegie Mellon University, 2007

[cit. 2016-05-10]. URL: https://www.cs.cmu.edu/~afm/Home_files/Das_Martins_survey_summarization.pdf.

[11] Inderjeet MANI. *Automatic Summarization*. Amsterdam: John Benjamins Publishing Company, 2001. ISBN: 1588110605.

[12] 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)*. Prague: Association for Computational Linguistics, 2007 [cit. 2016-05-10], pp. 448–457. URL: http://research.microsoft.com/pubs/77563/emnlp_svore07.pdf.

[13] Karen S. JONES. "Automatic summarizing: factors and directions". In: *Advances in automatic text summarization* (1999 [cit. 2016-05-11]), pp. 1–12. URL: https://www.cl.cam.ac.uk/archive/ksj21/ksjdigipapers/summbook99.pdf.

[14] Chin-Yew LIN. "Training a Selection Function for Extraction". In: *Proceedings of the Eighth International Conference on Information and Knowledge Management*. New York: ACM, 1999 [cit. 2016-05-10], pp. 55–62. ISBN: 1-58113-146-1. URL: http://doi.acm.org/10.1145/319950.319957.

[15] Christopher D. MANNING and Hinrich SCHÜTZE. *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press, 1999. ISBN: 9780262133609.

[16] Chinatsu AONE, James GORLINSKY, and Bjornar LARSEN. "A trainable summarizer with knowledge acquired from robust NLP techniques". In: *Advances in Automatic Text Summarization* 71 (1999).

[17] Hans P. LUHN. "The Automatic Creation of Literature Abstracts". In: *IBM J. Res. Dev. [online]* 2.2 (Apr. 1958 [cit. 2016-05-11]), pp. 159–165. ISSN: 0018-8646. URL: http://dx.doi.org/10.1147/rd.22.0159.

[18] Harold P. EDMUNDSON. "New Methods in Automatic Extracting". In: *J. ACM [online]* 16.2 (Apr. 1969 [cit. 2016-05-10]), pp. 264–285. ISSN: 0004-5411. URL: http://doi.acm.org/10.1145/321510.321519.

[19] Lucy VANDERWENDE et al. "Beyond SumBasic: Task-focused Summarization with Sentence Simplification and Lexical Expansion". In: *Inf. Process. Manage. [online]* 43.6 (Nov. 2007 [cit. 2016-05-11]), pp. 1606–1618. ISSN: 0306-4573. URL: `http://dx.doi.org/10.1016/j.ipm.2007.01.023`.

[20] 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*. New York: ACM, 1995 [cit. 2016-05-10], pp. 68–73. ISBN: 0-89791-714-6. URL: `http://doi.acm.org/10.1145/215206.215333`.

[21] George A. MILLER. "WordNet: A Lexical Database for English". In: *Commun. ACM [online]* 38.11 (Nov. 1995 [cit. 2016-05-12]), pp. 39–41. ISSN: 0001-0782. URL: `http://doi.acm.org/10.1145/219717.219748`.

[22] Miles OSBORNE. "Using maximum entropy for sentence extraction". In: *Proceedings of the ACL'02 Workshop on Automatic Summarization*. Stroudsburg: Association for Computational Linguistics, 2002 [cit. 2016-05-10]. URL: `http://dx.doi.org/10.3115/1118162.1118163`.

[23] 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*. New York: ACM, 2001 [cit. 2016-05-10], pp. 406–407. ISBN: 1-58113-331-6. URL: `http://doi.acm.org/10.1145/383952.384042`.

[24] John J. HOPFIELD. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[25] 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*. New York: ACM, 2015 [cit. 2016-05-10], pp. 1110–1111. ISBN: 978-1-4503-3196-8. URL: `http://doi.acm.org/10.1145/2695664.2696073`.

[26] 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*. New York: ACM, 2015 [cit. 2016-05-10], pp. 1–6. ISBN: 978-1-4503-3738-0. URL: `http://doi.acm.org/10.1145/2811411.2811465`.

[27] 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*. Stroudsburg: Association for Computational Linguistics, 2001 [cit. 2016-05-10], 19:1–19:8. URL: `http://dx.doi.org/10.3115/1117822.1117837`.

[28] Günes ERKAN and Dragomir R. RADEV. "LexRank: Graph-based Lexical Centrality As Salience in Text Summarization". In: *J. Artif. Int. Res. [online]* 22.1 (Dec. 2004 [cit. 2016-05-11]), pp. 457–479. ISSN: 1076-9757. URL: `http://dl.acm.org/citation.cfm?id=1622487.1622501`.

[29] Roger W. SCHVANEVELDT, D.W. DEARHOLT, and F.T. DURSO. "Graph theoretic foundations of pathfinder networks". In: *Computers & mathematics with applications [online]* 15.4 (1988 [cit. 2016-05-11]), pp. 337–345.

[30] Roger W. SCHVANEVELDT, ed. *Pathfinder Associative Networks: Studies in Knowledge Organization*. Norwood: Ablex Publishing Corp., 1990, pp. 337–345. ISBN: 0-89391-624-2.

[31] Wei WANG et al. "HyperSum: Hypergraph Based Semi-supervised Sentence Ranking for Query-oriented Summarization". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. New York: ACM, 2009 [cit. 2016-05-10], pp. 1855–1858. ISBN: 978-1-60558-512-3. URL: `http://doi.acm.org/10.1145/1645953.1646248`.

[32] Chin-Yew LIN. "ROUGE: a package for automatic evaluation of summaries". In: *Workshop Text summarization Branches Out (ACL '04) [online]*. Ed. by Stan SZPAKOWICZ Marie-Francine MOENS. Barcelona: ACL, 2004 [cit. 2016-05-10], pp. 74–81. URL: `http://research.microsoft.com/en-us/people/cyl/was2004.pdf`.

[33] Jonas SJÖBERGH. "Older Versions of the ROUGEeval Summarization Evaluation System Were Easier to Fool". In: *Inf. Process. Manage.* 43.6 (Nov. 2007 [cit. 2016-05-12]), pp. 1500–1505. ISSN:

0306-4573. URL: `http://dx.doi.org/10.1016/j.ipm.2007.01.014`.

[34] Karel VACULÍK. "A Versatile Algorithm for Predictive Graph Rule Mining". In: *Proceedings of the 15th conference ITAT 2015*. Ed. by Jakub YAGHOB. Praha: CreateSpace Independent Publishing Platform, 2015 [cit. 2016-05-10], pp. 51–58. ISBN: 978-1515120650. URL: `http://ceur-ws.org/Vol-1422/51.pdf`.

[35] 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*. New York: ACM, 2007 [cit. 2016-05-10], pp. 901–904. ISBN: 978-1-59593-803-9. URL: `http://doi.acm.org/10.1145/1321440.1321571`.

[36] Meishan HU, Aixin SUN, and Ee-Peng LIM. "Comments-oriented Document Summarization: Understanding Documents with Readers' Feedback". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Ed. by Sung-Hyon MYAENG et al. New York: ACM, 2008 [cit. 2016-05-10], pp. 291–298. ISBN: 978-1-60558-164-4. DOI: `10.1145/1390334.1390385`.

[37] Dr. Aixin SUN. *Comments-Oriented Document Summarization*. `http://www.ntu.edu.sg/home/axsun/datasets.html`, visited 2016-05-20. 2015.

[38] Rafael FERREIRA, Luciano CABRAL, and Rafael D. LINS. "Assessing sentence scoring techniques for extractive text summarization". In: *Expert Systems with Applications [online]* 40.14 (Oct. 2013 [cit. 2016-05-10]), pp. 5755–5764. ISSN: 0957-4174. URL: `http://www.sciencedirect.com/science/article/pii/S0957417413002601`.