Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

# Automatic Name and Snippet Generation of Web pages with Unknown Content

Bachelor thesis

*Jonáš Amrich*

Study programme: Open Informatics
Specialisation: Computer and Information Science
Supervisor: Ing. Jan Šedivý, Csc.

Prague, May 2015

**Czech Technical University in Prague**
**Faculty of Electrical Engineering**

**Department of Cybernetics**

# BACHELOR PROJECT ASSIGNMENT

**Student:** Jonáš A m r i c h

**Study programme:** Open Informatics

**Specialisation:** Computer and Information Science

**Title of Bachelor Project:** Automatic Name and Snippet Generation of Web pages with Unknown Content

## Guidelines:

The information retrieval engines return a ranked list of URL for each search request. Each URL in the list is described by a title and a short descriptive snippet on text. The snippet is mainly generated from the URL content stored on the search server. Some of the pages do not allow robots to download the content, some of them are in graphics, some are written in JavaScript.

The task is to design and develop algorithms for automatically generating the title of pages with unknown content. Proceed in the following steps:

1. Review the state of the art.
2. Based on the state of the art analysis choose and implement one method.
3. Generate a testing sequence of web pages.
4. Run experiments and estimate the quality of title generation.
5. Summarize the results.

## Bibliography/Sources:

[1] Aggarwal, Charu C., Zhai, ChengXiang (Eds.): Mining Text Data, Springer 2012.
[2] Prabhakar Raghavan, Christopher D. Manning, and Hinrich Schütze: Introduction to information retrieval, Cambridge University Press, 2008.
[3] Pedregosa, Fabian, et al.: "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.
[4] Turpin, Andrew, et al.: "Fast generation of result snippets in web search."Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.
[5] Blair-Goldensohn, Sasha, et al.: "Phrase based snippet generation." U.S. Patent No. 8,010,539. 30 Aug. 2011.

**Bachelor Project Supervisor:** Ing. Jan Šedivý, CSc.

**Valid until:** the end of the summer semester of academic year 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
**Head of Department**

prof. Ing. Pavel Ripka, CSc.
**Dean**

Prague, February 5, 2015

**České vysoké učení technické v Praze**
**Fakulta elektrotechnická**

**Katedra kybernetiky**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

**Student:** Jonáš A m r i c h

**Studijní program:** Otevřená informatika (bakalářský)

**Obor:** Informatika a počítačové vědy

**Název tématu:** Automatické generování názvu a popisek webových stránek s neznámým obsahem

## Pokyny pro vypracování:

Algoritmy pro vyhledávání informací obvykle vrací na každý dotaz seřazený seznam URL. Každá URL adresa v seznamu je popsána názvem a krátkým popisným textem. Popisek je generován především z obsahu URL uložené na vyhledávacím serveru. Některé URL stránky nedovolují robotům stažení obsahu, některé jsou v grafice, některé jsou psány v jazyce JavaScript.

Úkolem je navrhnout a vyvinout algoritmy pro automatické generování titulů stránek s neznámým obsahem. Postupujte podle následujících kroků:

1. Proveďte rešerši stavu používaných technologií.
2. Na základě rešerše vyberte a implementujte jednu metodu.
3. Sestavte trénovací a testovací sekvence webových stránek.
4. Proveďte základní experimenty a odhadněte kvalitu vygenerovaných titulků.
5. Shrňte a okomentujte výsledky.

## Seznam odborné literatury:

[1] Aggarwal, Charu C., Zhai, ChengXiang (Eds.): Mining Text Data, Springer 2012.
[2] Prabhakar Raghavan, Christopher D. Manning, and Hinrich Schütze: Introduction to information retrieval, Cambridge University Press, 2008.
[3] Pedregosa, Fabian, et al.: "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.
[4] Turpin, Andrew, et al.: "Fast generation of result snippets in web search."Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.
[5] Blair-Goldensohn, Sasha, et al.: "Phrase based snippet generation." U.S. Patent No. 8,010,539. 30 Aug. 2011.

**Vedoucí bakalářské práce:** Ing. Jan Šedivý, CSc.

**Platnost zadání:** do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
**vedoucí katedry**

prof. Ing. Pavel Ripka, CSc.
**děkan**

V Praze dne 5. 2. 2015

# Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 22. 5. 2015

.....
Jonáš Amrich

# Contents

# List of tables



# List of figures

# Abstract

This thesis explores the problem of name and snippet generation of web pages based on their context rather than on their content. A growing number of web pages is short on text and rich on multimedia, or is highly interactive, or their content can not be downloaded for various reasons. But these web pages may still be valuable for users and search engines need to present their labels. In this thesis a survey of several approaches to this task is presented, the process of automatic generation of the synthetic content is proposed and the performance of proposed method is measured. Exemplary snippets are then generated.

# Abstrakt

Tato práce se zabývá problémem automatického generování názvů a popisek webových stránek, pomocí jejich okolí namísto jejich obsahu. Rostoucí počet webových stránek je namísto textem tvořen multimédii či je interaktivní, případně jejich obsah nemůže být stažen z různých důvodů. Tyto stránky ale mohou být pro uživatele důležité a vyhledávače potřebují uživatelům prezentovat jejich popisy. V této práci je zkoumána řada přístupů k této problematice a je představena metoda pro automatickou tvorbu názvů a popisků. Její úspěšnost je následně změřena a jsou vytvořeny ukázkové popisky.

# Acknowledgments

I would like to thank Ing. Jan Šedivý, Csc. for his support and guidance and to thank Ing. Tomáš Tunys for his helpful advice.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

# Chapter 1

# Introduction

## 1.1 Motivation

*Almost every article on the topic of modern text mining algorithm starts with the sentence "With the rapid growth of the world wide web, there is an increasing need of [the application of the algorithm]". So does this thesis:*

With the rapid growth of the world wide web, there is an increasing need of automatic labeling and summarization. The Internet nowadays contains more data than ever and the amount of information online grows like never before. According to a study by IDC [11], the amount of the information on the Internet is now about 10 000 exabytes[1]. In this amount of data, it is virtually impossible to find desired information simply by browsing the Web using hyperlinks. Hyperlinks are still the very essential part of the World Wide Web, and this thesis is based on their principles, but for the majority of people the first everyday interaction with the Web is through the search window.

The key principle of every search engine is to respond to user queries and present the overview of related documents, that should lead the user to one of them, which contains the answer to his query. The decision whether the user visits the document or not is based on the short information provided by the search engine — the title and the snippet, which is a short summary of the content of the document.

The process of generation of the snippets is tightly coupled with the way how search engines, or information retrieval engines in general works. First, the inverted index of documents is built. Next, when the user submits his query, documents are looked up in the index, ranked, and presented to the user on the search results page.

Those inverse indexes are generated by robots crawling the web and downloading the content of the web pages. But some web pages can not be downloaded for various reasons. They may not allow the robots to download the content or may be in a non-textual form, e.g. in graphics, video, flash or JavaScript. Also, the content of the page may be very sparse or just insufficient. But these web pages may still be valuable for the user and the search engine may find them important because of their off-site signals like PageRank. Under normal circumstances the title and snippet would be gathered from the web page; but for these pages we need to automatically generate the title or snippets using other information.

[1] 1 EB is $10^{18}$ bytes or 1 billion gigabytes

## 1.2 Problem definition

In our case, the task is to generate the title, which is usually few words long and should include the words that are most descriptive for the page and the snippet, which is a few sentences long extractive summary of the document. Both tasks must be completed without the access to the target page. This can be formulated as a special case of context based extractive summarization. Automatic text summarization in general is an well established and explored domain — the demand for it emerged naturally with the growth of the Web. Plenty of high quality articles about this topic can be found. The short overview of history of automatic summarization is presented in 2.2, however this thesis is not intended to provide a deep overview of all approaches. Such comprehensive overview of all kinds of summarization can be found in [19]; also the following text will use the notation presented there for most of the summarization problems.

The task is to utilize knowledge from the web structure and from other accessible web pages to generate the synthetic content. This is possible because of the very nature of the web seen as a collection of hypertext documents linking to each other (this hypothesis is examined in 2.4). The page with unknown content for which we would like to generate title or snippet is called the *target page*. Other pages on the web that link to the target page (e.g. they contain one or more hyperlinks pointing to the target page) are called *source pages*. Next, the set of all source pages for one target page is called its *context*.

The goal is to generate meaningful, discriminative title and snippet characterizing the document. Both meaningfulness and ability to characterize are in nature subjective metrics, and it would be another challenging task only to measure them (see [16]), even with proper gold standard[2]. Therefore, in this work, the real contents of target web pages are used as the gold standard. Various metrics are then used to evaluate the quality of generated content.

[2]Manually annotated test set

# Chapter 2

# State of the art analysis

The need for automatic summarization has emerged a long time before the Internet era. Foundations of the field were laid in Luhn's paper from 1958 [17]. Since then, with the rise of the World Wide Web, automatic summarization went through rapid development and new methods and approaches emerged. In this chapter I present the most important approaches from my point of view, which are preceded with a short classification of the field of summarization.

## 2.1 Types of summarization

### 2.1.1 Abstractive summarization

Abstractive summarization is the most natural way of reinterpreting the content of the document. The summary may contain phrases from the original document, but is overall written with it's author's own words. That requires full understanding not only of the meaning of the document, but also of the language and it's interpretation. Summaries written by humans also contain background knowledge about the topic of the document. In this time, automatic abstractive summarization is a difficult problem that relies on development in other fields like natural language generation. Nevertheless, several attempts were made in this field with promising results - see the work of Yves Petinot, Kathleen McKeown and Kapil Thadani [24].

### 2.1.2 Extractive summarization

Extractive summarization follows the hypothesis, that individual sentences that are present in the document (or in its context in our case) can be extracted and combined into one or more meaningful paragraphs. This approach simplifies the problem to finding relevant and important sentences. Nowadays, the majority of real world applications of summarization are extractive, as the difficulty of the task is in the boundaries of current technology. The extractive approach is the one that will be further examined in this thesis.

### 2.1.3 Query dependent summarization

Another possibility of classification of summarization techniques is whether the summaries are dependent on the users query or are static for each document. As defined by [28], query dependent (or query biased) summary should provide enough information for user to

make the decision whether to retrieve the proposed document or not — to get the answer for his query. Some approaches can be found in [29] or [20], however, query dependent summarization is beyond the scope of this thesis, but may be the subject of my future work.

## 2.2 Brief history

The very first paper that has shaped the process of automatic summarization was already mentioned - Luhn's "The Automatic Creation of Litarature Abstracts" [17]. His approach was extractive, following the hypothesis that most frequent words in the document are the most descriptive. However, some of these most common words are not descriptive at all, namely stopwords and words that are common for a particular domain. To address these caveats, he used high and low frequency thresholds to filter out unwanted words. Following that, he selected sentences with highest density of descriptive words and included these in a summary. The so-called "auto-abstract" was generated from individual significant sentences of the article. The mechanism of selection of significant sentences is essential for majority of future works in summarization and differences are mostly only in ranking systems. In Luhn's work, frequencies of words in the article and also relative position of the word in the sentence were combined into the significance factor. Words with frequencies between given thresholds were labeled as significant and used for sentence selection, that has also taken in account the proximity of words in the sentence. Next, whole sentences or their portions are combined into one auto-abstract, complete with list of most significant words. This relatively simple principle sets the baseline for future researchers. While the schema of ranking words or sentences by importance remains, methods of ranking still evolves.

With the foundations laid by single document summarization, methods for multiple documents summarization and web page summarization could be developed. Multiple documents summarization is usually applied on many documents on the same topic - the first systems were aggregating multiple news articles into one summary (see [18]).

Web page summarization utilizes knowledge about the structure of the web to produce summarization of web page with known or unknown content. Although the first approaches use the content of the page only, those may contain relevant and useful principles. In 2000, two interesting approaches were introduced, the OCELOT [5], relying on machine translation principles and the InCommonSense [2], leveraging knowledge from the structure of the web page.

## 2.3 Machine learning approaches

While the early approaches were based on statistical models with hand made features, methods using machine learning techniques emerged soon. In 1995, Kupiec et al. [14] used Bayesian classifier trained on 188 documents and corresponding human labeled summaries. Another approach to the summary generation using classification model can be found in [1], where various types of features were used and some of them were also expanded with word clusters built using local context analysis. With this features, a ranking algorithms based on logistic regression and boosting algorithms were trained. Both approaches were focused on single document summarization, opposed to the approach proposed in [20]. In this article, a method using Support Vector Regression is applied to

the problem of query-focused multi-document summarization, otherwise them method is still based on ranking and extraction of the sentences.

## 2.4 Context based summarization overview

The hypothesis behind context based summarization (where the context is defined as in 1.2) is that pages that are in source — target relationship share the same topic and contain some overlap information. This was proved in [9], where textual similarities of web pages in various relationships were measured. In this work, Davison has used the tf-idf vector representation of web pages and compared them using cosine similarity measure, which is defined as:

$$cos(D_i^{tf-idf}, D_j^{tf-idf}) = \frac{\sum_{k=0}^{t} w_{ik} \times w_{jk}}{\sqrt{\sum_{k=0}^{t} w_{ik}^2 \times \sum_{k=0}^{t} w_{jk}^2}} \tag{2.1}$$

Where documents $D_i$, $D_j$ are defined as in 3.2.

He has shown that the similarity of anchor text, or window of 20 terms before and after the anchor is about at least a magnitude better than random. Within his dataset, the chance of anchor text appearing on the target page is 65%, while the words surrounding the anchor has 51% chance. Also the overlap of linked pages shows very similar trend.

The hypothesis of context—target similarity was analyzed even before publication of the Davison's article. One of the interesting applications was presented by Attardi in [4], where he has shown that information from the context leads to better results in web page classification. In addition to classification, Glover et al.[12] has proposed method for naming clusters of documents based solely on the context: They used terms both from the full text of source pages, as from the window surrounding the anchor. Using the expected entropy loss, they ranked the terms and constructed a list of possible names of the category of web pages. In their case, every category was successfully named using the first or the second ranked term.

Proper use of the web page context for summarization was introduced by Delort et al. in [10], where two algorithms of which one was based only on the context were revealed. He described three essential kinds of issues, that any context-based summarizer has to deal with:

1. contextualization - selecting related pieces of information about the target document from the context and leaving out the unrelated

2. partiality - combining partial information or finding information that covers entire content of the target

3. topicality - finally, selecting information that describes the *content* of the target, opposed to information related to the target though not mentioning its content.

The framework for context based summarization was derived by Delort from these issues. First, the contextualization issue is faced: only sentences near the anchor are selected, then annotated using part-of-speech tagger and filtered using various rules. Some of the rules can be called syntactic - e.g. the sentence must contain a verb; some of the rules deal with the structure of the web pages - e.g. splits on lists and sections or limits for the number of links in the sentence. The partiality issue is then solved using the inclusion

degree of sentences. Details on the definition of the measure of inclusion are in chapter 3.5. The last one is the topicality issue, for which Delort proposes two algorithms - mixed approach and context-based approach.

### 2.4.1 Other web page contexts

The context of the web page does not have to be limited only to the hyperlinks representing web structure. One interesting approach was published by JT Sun et al. [27], that exploits clickthrough data from user's search engine queries. In their approach, the selection of significant words from the page content is determined by the clickthrough data, afterwards the process continues as a regular single page summarization. Others [21] use specific user generated labels - social bookmarks from services like YouTube or Amazon, that acts as a web page context that expands limited information from the web page content.

# Chapter 3

# Proposed approach

In this chapter, the method for automatic selection of the most descriptive terms from the web page's context and for extractive summarization dependent on these terms is proposed. It is based mainly on the method presented by Delort in [10], but is slightly modified in various aspects. The most noticeable difference is in use of regression model for ranking the terms. Similar approaches can be found in [1, 20].

The method consists of a few steps that are described below. At the beginning, dataset containing source documents, target documents and anchors describing their relations is provided. In the preprocessing phase, the context for each target document is built and a bag of words representation is generated. Then, top $n$ candidates for the most descriptive terms are pre-selected from the context. These are consequently ranked using pre-trained regression model. The training of the model is also described in the following text. Using the ranked terms, a pre-selection of sentences suitable to be included in the snippet is made. For each of these sentences a feature vector is computed and the sentences are sorted according to manually defined rules. At the end, the snippet is constructed using the subset of sorted sentences with different content.

## 3.1 Building the context

The dataset consists of crawled web pages, therefore some basic pre-processing is required. Exact pre-processing steps performed on the data are described in the next chapter 4.1.2. When every web page is represented as a set of paragraphs cleaned of HTML markup, the context of each target page can be built. Based on hypertext links, a set of source pages for every target page is gathered. All source web pages from the dataset are included in the context, even pages belonging to the same domain as the target (compared to Delort's approach [10], where these source pages are omitted).

Multiple representations of the page are then used, depending on the amount of text selected from the page. First, a representation that includes a fixed window of three paragraphs before and after the link is obtained. The second representation uses the whole content of every source page - this follows the assumption that when the context is large enough, the noisy parts of web pages, that are not informative about the target page, are not statistically significant. This assumption needs to be further examined; see the section 4.2.2 for a short review of the results.

## 3.2 Bag of words

In the next step, all documents are represented in a vector representation[26] following the bag-of-words model. In this representation, syntactic properties including word order are disregarded and only numbers of appearances of words are kept. Each document $D_i$ is represented by a sparse $t$-dimensional vector. The dimensionality of the vector $t$ is determined by a number of terms present in the dictionary $W$, which is usually built from the set of all documents (see 4.1.2 for detailed information about the dictionary).

$$D_i = \{w_{i1}, w_{i2}, ..., w_{it}\} \tag{3.1}$$

where $w_{ij}$ represents the weight of the term $j$ in document $i$. Two different weighting schemes are used for the next processing, term frequency and tf-idf, respectively. TF-IDF, or term frequency—inverse document frequency is a widely used weighting system and is defined as:

$$\begin{gathered} w_{ij} = tf_{ij} \cdot \log \frac{N}{\sum_{k=0}^{N} tf_{kj}} \\ \hat{w}_{ij} = \frac{w_{ij}}{|w_{ij}|} \\ D_i^{tf-idf} = \{\hat{w}_{i1}, \hat{w}_{i2}, ..., \hat{w}_{it}\} \end{gathered} \tag{3.2}$$

where $tf_{ij}$ is the frequency of term $W_j$ in the document $D_i$, and $N$ is the total number of documents.

With the vector representation of single documents, we can proceed to the representation of the context. For every target document $D_\tau \in \mathcal{D}$, a context vector $C_\tau$ is built according to these rules:

$$\begin{gathered} context(D_\tau) = \{D_i \in \mathcal{D}; D_i \text{ is source document for } D_\tau\} \\ C_\tau = \{\sum_{D_j \in context(D_\tau)} w_{j1}, \sum_{D_j \in context(D_\tau)} w_{j2}, ..., \sum_{D_j \in context(D_\tau)} w_{jt}\} \end{gathered} \tag{3.3}$$

After this step, four different vector representations of documents and contexts are present:

$$\begin{aligned} \mathcal{D} &= \{D_1, ..., D_N\} \\ \mathcal{D}^{tf-idf} &= \{D_1^{tf-idf}, ..., D_N^{tf-idf}\} \\ \mathcal{C} &= \{C_1, ..., C_T\} \\ \mathcal{C}^{tf-idf} &= \{C_1^{tf-idf}, ..., C_T^{tf-idf}\} \end{aligned} \tag{3.4}$$

where $T$ denotes total number of target documents.

## 3.3 Top $n$ words selection

For the ranking phase, it is more meaningful (and much faster) to rank only top $n$ pre-selected words, instead of whole dictionary present in the context. As we predict the most significant words in terms of tf-idf, it feels natural to use this metric for pre-selection too. My selection of top $n$ words from the context contains words with high both tf-idf and tf scores; precisely all unique words from the set of top 100 words with highest tf and top 100 words with highest tf-idf from sets $\mathcal{C}$ and $\mathcal{C}^{tf-idf}$ respectively.

## 3.4 Top $n$ words ranking

Ranking of the most descriptive terms is the key part of both title and snippet generation. This is done using the random forest algorithm, which is first trained using the sample data. The process of feature extraction, training and application of the model is described in this section.

### 3.4.1 Learning to rank

Learning to rank is a supervised or semi-supervised machine learning method, whose purpose is to order a set of items according to some given measure. In general, the goal of the LTR[1] model is to produce ranked permutation of given set based on information gathered from training data. There are three main approaches to LTR, as defined in [15]: *the pointwise approach*, *the pairwise approach* and *the listwise approach*.

The task fits best to the pointwise approach, where the input and output spaces are defined as follows: The input space contains the feature vector of a single document, in our case the feature vector of a single word (the feature vector is described below); and the output space contains the relevance label, which is in our case the importance of the word. The importance label can be represented by a non-ordered category (e.g. important / not important) or as a real number (higher number means more important word). Depending on the representation of the importance label and corresponding algorithm we talk about classification based approach and regression based approach.

Various classification and regression algorithms can be used, for a comprehensive list and benchmarks see [15]. As the difference in performance of the algorithms is not so significant, I did not perform additional experiments and selected the widely used Random forest algorithm.

### 3.4.2 Random forests

Random forests, as introduced by Leo Breiman [7] is an ensemble learning method[2] based on a collection of decision trees that vote for the final decision - in the case of classification a mode of the classes is used, in the regression case the mean of outputs is used as final decision. In following experiments both classification and regression approaches are compared.

[1] Learning to rank

[2] Ensemble methods combines multiple weak learners into one strong

### 3.4.3 Features

There are three different types of features extracted from the data: *context-wise* (document independent) *term features* which are computed for the whole context of the target document and does not take in account individual documents; *document-wise term features*, that are computed for each source document individually; and *source document features*, that are independent on the term and rather describe the source document in general. The number in square brackets denotes position in feature vector and may be used as a legend for the results section. For a given word $W_k$ in document $D_i$ from the context $C_\tau$ of target document $D_\tau$, the features are:

**Context-wise (document independent) term features:**

- Term frequency in the context vector $C_\tau$ [1.1]
- TF-IDF in the context vector $C_\tau^{tf-idf}$ [1.8]
- Term frequency[3] in anchor texts pointing to the $D_\tau$ [1.7]
- Term frequency[3] in titles of source documents [1.2]
- Term frequency[3] in html meta description [1.5]
- Term frequency[3] in html keywords [1.6]
- Term frequency[3] in query strings — the query strings that are frequently used to search the web page using *Seznam.cz* search engine [1.4]
- Term frequency in tokenized URL [1.3]
- Number of documents in context that contain the term [1.9]
- Number of documents in context that contain the term, normalized [1.10]

**Document-wise term signals:**

- Term frequency in document $D_i$ [2.1]
- TF-IDF in document $D_i^{tf-idf}$ [2.2]

**Source document signals:**

- Pagerank of the source document [3.1]
- Body size of the document including boilerplate code [3.2]
- Is document homepage (true for homepage documents like index.html, false otherwise) [3.3]

The features are listed in arbitrary order, for an overview of learned feature importances of the LTR model, see the chapter 4.2.

[3]To be correct, the term frequency here means sum of term frequencies in all source documents

### 3.4.4 Training sequence

As was mentioned before, the input space of the LTR model contains the feature vector of a single word present in the context. When training only on context-wise features, there is a single input vector for each word $W_j$ from the pre-selected words from the context $C_\tau$:

$$x_j = \left[\ l_j \mid f_{1.1,j} \quad f_{1.2,j} \quad \cdots \quad f_{1.n_1,j}\ \right] \tag{3.5}$$

When we have to take in account the document-wise features, each word $W_i$ must be represented in the training sequence exactly $n_d$ times, where $n_d$ is number of documents in $context(D_\tau)$ containing the word. For the context $C_\tau$ and word $W_j$, the set of input vectors consists of "simple" input vectors $x_j$ enhanced of document dependent features:

$$\left[\begin{array}{c|cccccccc} l_j & f_{1.1,j} & \cdots & f_{1.n_1,j} & f_{2.1,j,1} & \cdots & f_{2.n_2,j,1} & f_{3.1,j,1} & \cdots & f_{3.n_3,j,1} \\ l_j & f_{1.1,j} & \cdots & f_{1.n_1,j} & f_{2.1,j,2} & \cdots & f_{2.n_2,j,2} & f_{3.1,j,2} & \cdots & f_{3.n_3,j,2} \\ \vdots & & & & & & & & & \\ l_j & f_{1.1,j} & \cdots & f_{1.n_1,j} & f_{2.1,j,n_d} & \cdots & f_{2.n_2,j,n_d} & f_{3.1,j,n_d} & \cdots & f_{3.n_3,j,n_d} \end{array}\right] \tag{3.6}$$

where $f_{1.n,j}$ is a context-wise feature of word $W_j$ and $f_{2.n,j,i}$ and $f_{3.n,j,i}$ are a document-wise features of word $W_j$ dependent on document $D_i$.

The label $l_j$ depends on the selected approach. For the classification based approach to LTR, the output space contains only two categories: *important / not important*. This is determined by the presence of the word in the target document: $l_j = 1$ iff $w_{\tau,j} > 0$ otherwise 0. In the regression based approach, the label is a real number, in our case the tf-idf score of the word in the target document: $l_j = \hat{w}_{\tau,j}^{tf-idf}$.

## 3.5 Building the snippet

The snippet is composed of individual sentences extracted from the context of the target page. The process of sentence selection is build up on the output of the LTR algorithm, which is supposed to produce ranked list of important words. As the first step of the process, the sentences are preselected from the context according to the number of important words they contain. This threshold is set heuristically and serves only as mechanism for reducing the number of candidates for the next step. There are also few features gathered from sentences in this step:

- Sum of predicted tf-idf scores of important words
- Number of important words
- Distance from the anchor in the source document
- Length of the sentence
- Number of verbs in the sentence

Next, the sentences are ranked according to their relation to the target page. At the present time, the dataset is not annotated by hand, so the ranking of sentences is done using only heuristics instead of trained model. When the ordered list of sentences is constructed, the snippet is generated in a greedy way. To address the partiality issue, after the selection of the first sentence on the list, some remaining sentences are removed, based on their inclusion degree.

The inclusion degree of two sentences $S_i$ and $S_k$, is computed in a same fashion as in [10]:

$$I(S_i, S_k) = \frac{\sum_{j=0}^{t} w_{ij} \cdot w_{kj}}{\sum_{j=0}^{t} w_{ij}} \tag{3.7}$$

where the sentences are defined as $S_i = \{w_{i1}, w_{i2}, ..., w_{it}\}$ where individual words are in the boolean representation, which assigns 1 to the word which is present in the sentence and 0 otherwise.

The final snippet is generated with specific length, which is usually between two and five sentences. With the ranked set of sentences $\mathcal{S} = \{S_1, ..., S_{n\_sentences}\}$, the snippet $X$ is constructed according to this algorithm:

$X \leftarrow \{\}$
**for** 1 **to** target snippet length **do**
  $X \leftarrow X \cup S_1$
  $\mathcal{S} \leftarrow \{S_i \in \mathcal{S}; I(S_i, X) < 1\}$
**end for**

The algorithm takes the first sentence and removes it from the list, along with other sentences that can be expressed using the first sentence. The steps are repeated until the target length of the snippet is not reached, removing sentences from the list of candidates according their inclusion degree with all already selected sentences together.

# Chapter 4

# Experiments

## 4.1 Dataset

### 4.1.1 Original dataset

The dataset for my experiments was provided by *Seznam.cz*, for what I am very grateful as it is not easy to obtain data of this kind and size. However this brings some downsides - while the dataset contains features that directly affect the search results at *Seznam.cz*, the data can not be publicly released.

The original dataset contains more than 1 million documents and corresponding anchors, which represent a random sample of data crawled by the company. Both documents and anchors are represented in the protocol buffers format[13]. The documents went through basic preprocessing and their derivates were build: the content is split into pseudo-paragraphs (as these are not only html <p> elements) that are extended with various features. They include features for the whole page as pagerank, language and various others; and also features for individual paragraphs. One of them and one of the important ones is a label from the body text extraction algorithm, which is used to distinguish body text and boilerplate content, such as site-wide headers, menu or footer. This labeling and first preprocessing was done internally in *Seznam.cz*.

### 4.1.2 Preprocessing

Working with the whole dataset is impractical and computationally expensive, thus it was split into various smaller datasets differing both in document selection and preprocessing applied. It is common for all derived datasets, that only pages written in English were selected - although the proposed method is language independent[1], avoiding the complexity of Czech or multilingual documents makes things little bit easier.

As the first thing, the dictionary is built. It contains words from the paragraphs marked as body text that are between 2 and 25 characters long. The words with appearance in less than 5 documents were discarded, so were the words present in more than 80% documents. Also, the stopwords were not included in the dictionary, based on the stopwords list from the NLTK toolkit [6]. After this process the dictionary contains 847 743 tokens.

[1]There are actually few points where language specific methods are used, e.g. the stopwords list and the POS tagging, but only minor modifications are needed for extension to other languages

Next, the web pages were filtered according to their context. The list of target pages was built, which included pages with 100-2000 source pages. Only links that appear in the body text part of page were considered.

### 4.1.3 Derived datasets

From the pre-selection of documents, two datasets were constructed: *fullpage* and *partpage_p3*. Each of them uses different representation of page, according to the amount of text used from the page. The dataset labeled *fullpage* contains all paragraphs from the body text section of web page, contrary to the dataset labeled *partpage_p3*, which contains only the paragraph with the link and three paragraphs directly before and after it. As this may sound as a large window, keep in mind that the paragraphs are actually pseudo-paragraphs and they may be also headings, lists and other html elements. The mean size of the document is then 206 words for the *partpage_p3* and 1277 for the *fullpage* dataset.

Both datasets contain 1672 target documents and 444 291 source documents - the average is almost 266 source documents for one target. Due to the preprocessing, 192 of the target documents ended as blank. These documents were removed from the ranking.

## 4.2 Learning to rank

### 4.2.1 Features

Two different selections of features were made, as noted in 3.4.3. The models trained only on context features are labeled *f1*, the ones trained on both context-wise and document-wise features are labeled *f3*. For the models from the group *f1*, the number of training data was by orders of magnitudes smaller then for *f3*. For a comparison, there were 167 200 input vectors for the *fullpage-f1-rfr* model against 18 080 003 input vectors for the *fullpage-f3-rfr* model. The large number of input vectors was caused by expansion of the data with document-wise features, as presented in 3.6, and is one of the reasons of worse performance of later models, as it causes overfitting.

Of the training vectors, usually more than a half of them are positive examples, with target labels *important* or nonzero target tf-idf score (for the fullpage-f3-rfr the ratio of positive to negative training data is 70:30).

The feature importances of the learned model can be computed, which provides useful insights. In case of models I have used, the importance is computed as a mean decrease accuracy - more information about the implementation can be found in [8]. There is an interesting trend, that can be seen from the data in the tables 4.1 and 4.2: the regression based models use mainly tf-idf features, as they try to predict the tf-idf value of words in target document. On the other hand the classification models find the number of source documents containing the term (feature 1.9) as the most important, as their task is only to differentiate important and unimportant words. It also seems that the importance of document-wise features is fairly low, but this may be caused by the expansion of training vectors. The trend is similar for other classification and regression based models; the full list of features importances of most of the mentioned models can be found in A.

Table 4.1: Feature importances of *fullpage-f3-rfr*

| | | | | |
|---|---|---|---|---|
| **tf-idf [1.8]** | **# docs [1.10]** | **# docs [1.9]** | **tf [1.1]** | **tf/anchors [1.7]** |
| 0.48 | 0.09 | 0.09 | 0.08 | 0.08 |
| **tf/meta [1.5]** | **doc tf-idf [2.2]** | **body size [3.2]** | **tf/url [1.3]** | **tf/query [1.4]** |
| 0.04 | 0.03 | 0.02 | 0.02 | 0.02 |
| **pagerank [3.1]** | **tf/titles [1.2]** | **tf/keywords [1.6]** | **doc tf [2.1]** | **is hp [3.3]** |
| 0.02 | 0.01 | 0.01 | 0.01 | 0.00 |

Table 4.2: Feature importances of *fullpage-f1-rfc*

| | | | | |
|---|---|---|---|---|
| **# docs [1.9]** | **# docs [1.10]** | **tf [1.1]** | **tf-idf [1.8]** | **tf/meta [1.5]** |
| 0.26 | 0.24 | 0.20 | 0.16 | 0.04 |
| **tf/query [1.4]** | **tf/titles [1.2]** | **tf/keywords [1.6]** | **tf/url [1.3]** | **tf/anchors [1.7]** |
| 0.03 | 0.03 | 0.02 | 0.02 | 0.01 |

### 4.2.2 Models

Two different ranking models were compared: Random Forest Classifier and Random Forest Regressor. These were trained using data obtained in the previous step, when the whole dataset was split into train and test parts, in the 50:50 ratio. As the data was split on whole contexts (one target document and corresponding context is the elementary part of the dataset), the ratio can be $\pm 10\%$ off in terms of number of input vectors.

In addition to different datasets and models, I have also experimented with model parameters, mainly with number of trees and the maximum number of features considered for the best split. As the effect of different parameters was not so significant for the task, the evaluations were done with 60 decision trees in the forest for the *f1* models and with 30 trees for the *f3* models (the later is mostly due to computing requirements). Other parameters were left with the default values. The detailed comparison of models with different forest sizes can be found in table 4.3, where scores for *fullpage-f1-rfr* are presented.

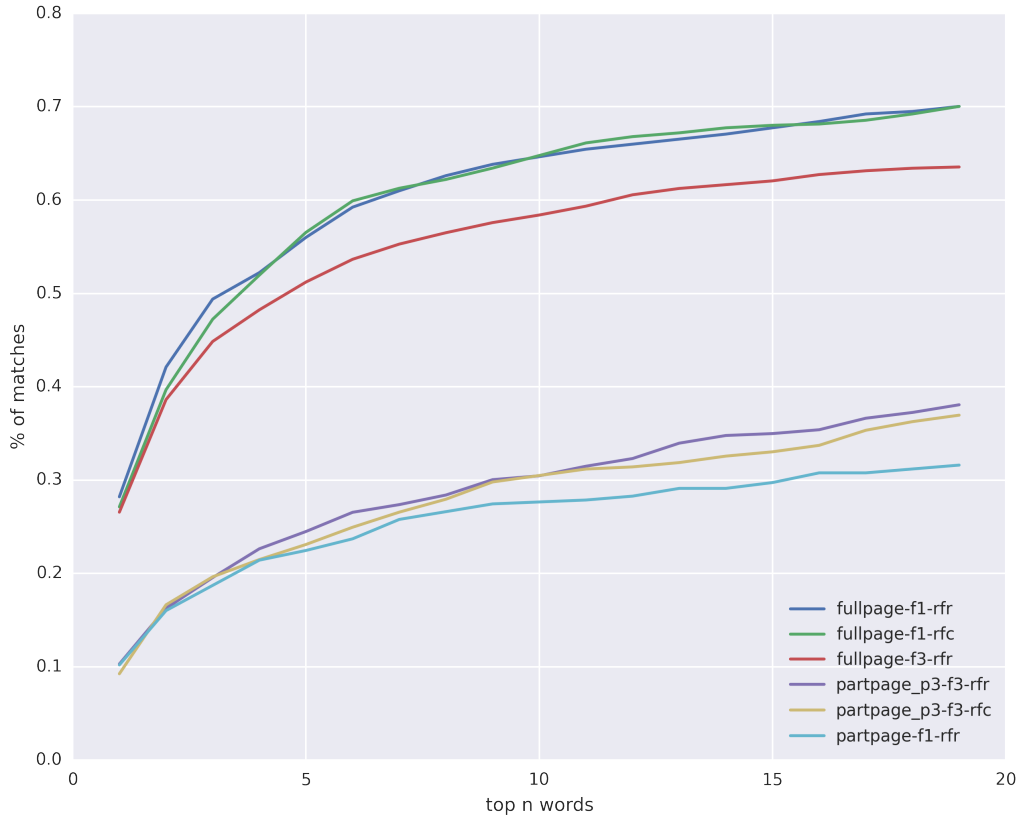
Table 4.3: Comparison of different numbers of trees for the *fullpage-f1-rfr* model

| | 10 | 30 | 60 | 100 | 500 |
|---|---|---|---|---|---|
| $R^2$ | 0.3042 | 0.3432 | 0.3537 | 0.3563 | 0.3616 |
| $MSE$ | 0.0019 | 0.0018 | 0.0017 | 0.0017 | 0.0017 |

The performance of models is measured using standard metrics for evaluation of random forests, as well using the metric more suitable to the application - the ability to predict the target word which is most descriptive for the page. The models output ranked list of words, from which the first one is selected as the prediction of the most descriptive word. This is then compared with the ordered words in the target document and a mean

position of the predicted word is computed. For the highest ranked model, the *fullpage-f1-rfr*, the predicted word is in the top five words on the page for 56% predictions. The full comparison can be seen in the figure 4.1.

Figure 4.1: Comparison of the performance of different ranking models




Two metrics that are also used are coefficient of determination ($R^2$) and mean squared error (MSE). If we mark the true labels from the dataset as $y_1, ..., y_n$, the predicted labels as $f_1, ..., f_n$ and the mean as $\bar{y} = \frac{1}{n}\sum_i y_i$, then the $R^2$ is defined as:

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{4.1}$$

And the mean squared error:

$$MSE = \frac{1}{n}\sum_i (f_i - y_i)^2 \tag{4.2}$$

The coefficient of determination $R^2$ computed according to the definition 4.1 can actually be a negative number. This is caused by high variance in the data, which is the case of the data from *partpage_p3-f3* dataset. The overview of $R^2$ and $MSE$ scores for both datasets and both feature sets is in the table 4.4.

For the classification based models, the mean accuracy score is used, which is defined as:

$$accuracy = \frac{1}{n}\sum_i 1 \text{ iff } f_i = y_i \text{ otherwise } 0 \tag{4.3}$$

The mean accuracy scores of the classification models can be seen in the table 4.5.

Table 4.4: Scores of regression models

| | fullpage-f3-rfr | fullpage-f1-rfr | partpage_p3-f3-rfr | partpage_p3-f1-rfr |
|---|---|---|---|---|
| $R^2$ | 0.1859 | 0.3537 | -0.3197 | 0.2242 |
| $MSE$ | 0.0036 | 0.0017 | 0.0079 | 0.0005 |

Table 4.5: Scores of classification models

| | fullpage-f3-rfc | partpage_p3-f3-rfc |
|---|---|---|
| mean accuracy | 0.7885 | 0.6096 |

## 4.3 Building the snippet

### 4.3.1 Sentence selection

For each target document, the list of top 15 most important words is predicted using the LTR algorithm (the *fullpage-f1-rfr* is used for the following experiments). From the context of the target, sentences are extracted from the body text part of each source document. This is done using the Punkt Sentence Tokenizer module from NLTK, that contains pre-trained model for English sentences. From these sentences, only those that contain at least three important words are selected to be included in the possible snippet. Sentences with more than three verbs are discarded and others are ranked according the sum of predicted tf-idf scores. The snippet is then generated using the algorithm 3.5.

### 4.3.2 Sample results

A selection of sample results was made by hand, to present some of the capabilities of the model. Apparently, the best results are achieved on target pages that represent a single entity, which could be seen on the example 4.6. Entities from the Wikipedia, items from e-commerce sites and articles focused on one topic belongs to the group of target sites where the snippet generation is successful. More examples could be found in the appendix A.3.

Table 4.6: Selected title and snippet generated by the *fullpage-f1-rfr* model.

| **URL**<br>titles<br>*Generated snippet* |
|---|
| **http://en.wikipedia.org/wiki/YLE**<br>film, channel, eurofighter, including, funding<br>*Number 3 was added later, when the channel was allocated the third nation-wide television channel and it generally became known as "Channel Three" – Finnish Broadcasting Company's Yle TV1 and Yle TV2 being the first two – and also to distinguishing it from the later MTV Finland. YLE, Finland's public broadcasting station, operates five television channels and thirteen radio channels in both national languages.* |

## 4.4 Implementation

The source code of the implementation of proposed method and a framework for running the experiments is available on GitHub [3]. The code is written mainly in python and makes use of various libraries, of which I would like to mention gensim [25], which was used for the pre-processing tasks and corpus representation; scikit-learn [22] which contains implementations of the random forest algorithms; NLTK [6] that contains models used for sentence tokenization and part of speech tagging; and I also appreciate the ipython[23] notebook for interactive python environment.

# Chapter 5

# Conclusion

In this thesis, various approaches to the problem of context based labeling and snippet generation were surveyed. A combination of two approaches was selected and implemented. The framework for the task was proposed and a comparison of two algorightms and various preprocessing tasks was performed, resulting in six different models. The influence of the amount of data gathered from each web page's context was examined; and the effects of different features to the selection of the most discriminative terms were studied.

The performance of these models was then measured and using the best model, exemplary snippets were generated. Due to the lack of labeled data, that could act as a gold standard for generated snippets, the evaluation of snippet quality will have to be done manually. Moreover, with labeled data, the use of a learning model for snippet generation would also be possible.

The context based content generation is a complex problem and as the topic is valued by *Seznam.cz*, the further research on the topic could be done. The learning model for snippet generation was already mentioned, but the other parts of the process can be explored more deeply, e.g. the semantic expansion for better coverage of title genaration, improved feature engineering and more.

# Appendix A

# Additional tables and figures

Table A.1: Feature importances of *partpage_p3-f3-rfr*

| | | | | |
|---|---|---|---|---|
| **tf [1.1]** | **tf-idf [1.8]** | **# docs [1.9]** | **tf/anchors [1.7]** | **# docs [1.10]** |
| 0.29 | 0.16 | 0.12 | 0.11 | 0.08 |
| **tf/meta [1.5]** | **tf/query [1.4]** | **tf/url [1.3]** | **tf/titles [1.2]** | **tf/keywords [1.6]** |
| 0.07 | 0.04 | 0.04 | 0.03 | 0.02 |
| **doc tf-idf [2.2]** | **doc tf [2.1]** | **body size [3.2]** | **pagerank [3.1]** | **is hp [3.3]** |
| 0.02 | 0.01 | 0.00 | 0.00 | 0.00 |

Table A.2: Feature importances of *partpage_p3-f3-rfc*

| | | | | |
|---|---|---|---|---|
| **# docs [1.9]** | **# docs [1.10]** | **tf [1.1]** | **tf-idf [1.8]** | **doc tf-idf [2.2]** |
| 0.20 | 0.19 | 0.14 | 0.13 | 0.09 |
| **tf/titles [1.2]** | **tf/keywords [1.6]** | **tf/query [1.4]** | **doc tf [2.1]** | **tf/url [1.3]** |
| 0.05 | 0.04 | 0.04 | 0.04 | 0.04 |
| **tf/meta [1.5]** | **body size [3.2]** | **pagerank [3.1]** | **tf/anchors [1.7]** | **is hp [3.3]** |
| 0.03 | 0.01 | 0.01 | 0.00 | 0.00 |

Figure A.1: Effect of size of the forest on the performance of ranking model. Shown on the *fullpage-f1-rfr* model.

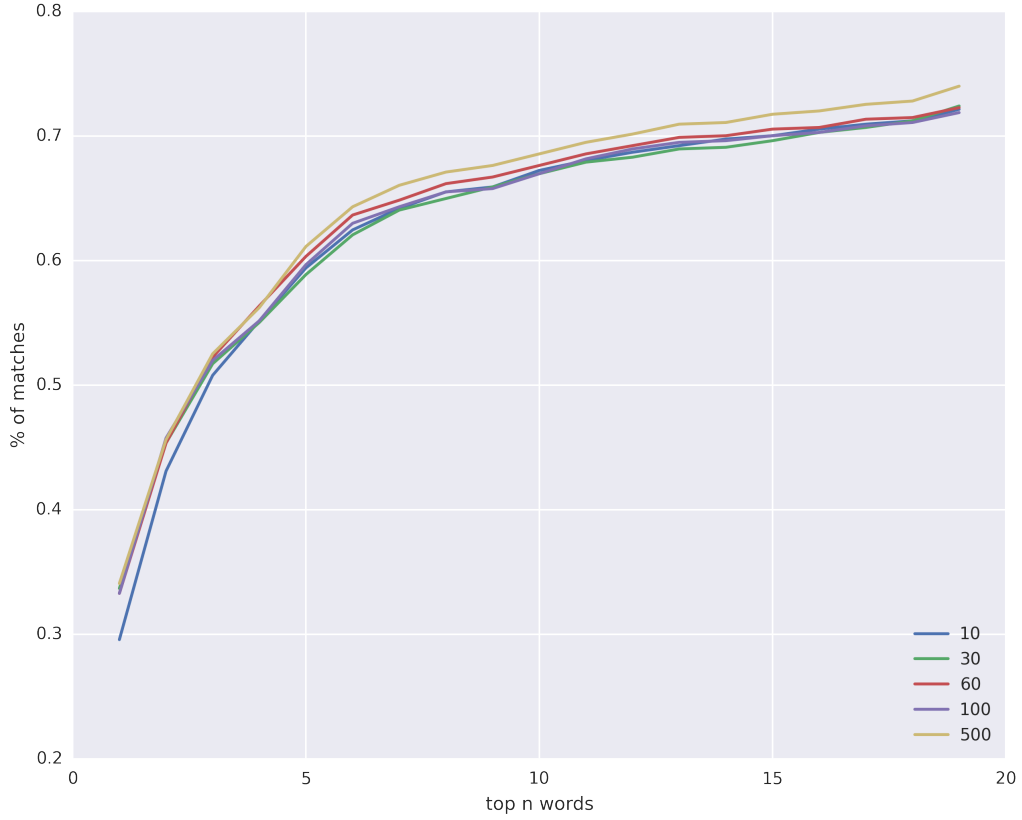


Table A.3: Selected titles and snippets generated by the *fullpage-f1-rfr* model.

| |
|---|
| **URL**<br>titles<br>*Generated snippet* |
| **http://wn.com/Nevada_Senate**<br>committee, arizona, hawaii, american, judiciary<br>*2014 Nevada State Senate District 4 Election Interview by Asian American Group Michele Fiore (Incumbent) http://www.VoteFiore.com This is a must see for all voters!!* |
| **http://therecipedaily.in/category/party-food/**<br>sausage, biscotti, pantry, bark, breakfast<br>*Recipe: Southwestern Butternut Squash Soup — Recipes from The Kitchn. SPONSORED POST: Recipe: Crunchy Chicken Salad with Grape-Nuts and Cranberries — Recipes from The Kitchn Sponsored by Grape-Nuts* |
| **http://www.masonlec.org/**<br>institute, papers, including, infrastructure, homepage<br>*Program Description: The Global Antitrust Institute (GAI) at the Law & Economics Center at George Mason University School of Law is a leading international platform for research and education that focuses on the legal and economic analysis of key antitrust issues confronting competition agencies and courts around the world. Moderator: James C. Cooper, Director, Research and Policy, Law & Economics Center and Lecturer in Law, George Mason University School of Law* |
| **http://www.dx.com/p/sewor-m113-2-men-s-pu-leather-band-self-winding-mechanical-analog-wristwatch-bla**<br>resin, keychain, leather, women, cree<br>*Sewor M113-2 Men's PU Leather Band Self-winding Mechanical Analog Wristwatch - Black + Silver* |

# Bibliography


[1] Massih R Amini, Nicolas Usunier, and Patrick Gallinari. Automatic text summarization based on word-clusters and ranking algorithms. In *Advances in Information Retrieval*, pages 142–156. Springer, 2005.

[2] Einat Amitay and Cécile Paris. Automatically summarising web sites: is there a way around it? In *Proceedings of the ninth international conference on Information and knowledge management*, pages 173–179. ACM, 2000.

[3] Jonáš Amrich. Repository for the code from this thesis on github. `https://github.com/JonasAmrich/SyntheticContent`.

[4] Giuseppe Attardi, Antonio Gulli, and Fabrizio Sebastiani. Automatic web page categorization by link and context analysis.

[5] Adam L Berger and Vibhu O Mittal. Ocelot: a system for summarizing web pages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 144–151. ACM, 2000.

[6] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees.* CRC press, 1984.

[9] Brian D Davison. Topical locality in the web. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–279. ACM, 2000.

[10] J-Y Delort, Bernadette Bouchon-Meunier, and Maria Rifqi. Enhanced web document summarization using hyperlinks. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 208–215. ACM, 2003.

[11] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. 2012.

[12] Eric J. Glover, Kostas Tsioutsiouliklis, Steve Lawrence, David M. Pennock, and Gary W. Flake. Using web structure for classifying and describing web pages. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 562–569, New York, NY, USA, 2002. ACM.

[13] Google. Protocol buffers. `https://developers.google.com/protocol-buffers/`.

[14] 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. ACM, 1995.

[15] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[16] Annie Louis and Ani Nenkova. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300, 2013.

[17] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[18] Kathleen R McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. Tracking and summarizing news on a daily basis with columbia's newsblaster. In *Proceedings of the second international conference on Human Language Technology Research*, pages 280–285. Morgan Kaufmann Publishers Inc., 2002.

[19] Ani Nenkova, Sameer Maskey, and Yang Liu. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*, page 3. Association for Computational Linguistics, 2011.

[20] You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47(2):227–237, 2011.

[21] Jaehui Park, Tomohiro Fukuhara, Ikki Ohmukai, Hideaki Takeda, and Sang-goo Lee. Web content summarization using social bookmarks: A new approach for social summarization. In *Proceedings of the 10th ACM workshop on Web information and data management*, pages 103–110. ACM, 2008.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

[24] Yves Petinot, Kathleen McKeown, and Kapil Thadani. Cluster-based web summarization. 2013.

[25] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[26] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[27] Jian-Tao Sun, Dou Shen, Hua-Jun Zeng, Qiang Yang, Yuchang Lu, and Zheng Chen. Web-page summarization using clickthrough data. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201. ACM, 2005.

[28] Anastasios Tombros and Mark Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 2–10. ACM, 1998.

[29] Andrew Turpin, Yohannes Tsegay, David Hawking, and Hugh E Williams. Fast generation of result snippets in web search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 127–134. ACM, 2007.