

Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

The Ranking and Comparison of Extracted Anchor Fragments

Master's Thesis

Masoud Allahyari
Born Feb 13, 1987 in Baghbahadoran

Matriculation Number 116246

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr. Norbert Siegmund

Submission date: July 2, 2018

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, July 2, 2018

.....
Masoud Allahyari

Abstract

It is an immensely challenging task for search engines to generate a suitable snippet, defined as a short description of a web page since the textual content on the Internet is constantly decreasing and being replaced by the hypermedia. This work aims to address the issue of web document summarization and investigate an alternative approach to improve the quality of snippets.

Our hypothesis is that the surrounded texts of a link could possibly be related to its target and could, therefore, be used to generate snippets. First, we will describe a pipeline which receives a link as input and retrieves several ranked lists of snippet candidate generated from surrounded texts (anchor fragments) of the links. consequently, we divide this pipeline into three major procedures. The first one finds and mines the text from the web documents. The second procedure tries to enhance the quality of the extracted texts and finally, the last procedure ranks the texts with different methods. Subsequently, we evaluate the results of this pipeline by conducting an online user study, which measures the satisfaction of participants on the results. Eventually, the effectiveness of each ranking method will be assessed based on participants votes.

Our findings include that (1) the most effective way of ranking fragments depends on the frequency of the similar fragments existing in different web pages, and (2) combining the fragments which have an identical part has a great impact on improving the quality of the fragments.

Contents

1	Introduction	1
2	Background and related works	5
2.1	Automatic text summarization techniques	5
2.1.1	Extractive text summarization	5
2.1.2	Abstractive text summarization	6
2.1.3	Web summarization	7
2.2	Summary	10
3	Proposed approach	11
3.1	The architecture of proposed system	11
3.2	Extracting anchor fragment	13
3.2.1	Surrounded text or paragraph	13
3.2.2	Extractor	15
3.3	Preprocessing anchor fragments	18
3.3.1	Features of meaningless fragments	19
3.3.2	Preprocessor	28
3.4	Indexing anchor fragments information	29
3.5	Ranking anchor fragments	29
3.5.1	Computing the similarity	30
3.5.2	Ranking methods	34
3.6	Summary	39
4	Evaluation	40
4.1	Common approaches	40
4.2	Experiment setup	41
4.2.1	Dataset	42
4.2.2	Implementation of approaches	42
4.2.3	Mturk setup	44
4.3	Evaluation Results and Discussion	51
4.3.1	Discussion	55

4.4 Summary	57
5 Conclusion and Future work	58
5.1 Contributions	58
A HIT Interface	61
Bibliography	63

*Life is like riding a bicycle. To keep your balance, you
must keep moving.*
– *Albert Einstein*

Acknowledgements

A very special gratitude goes out to Prof. Dr. Matthias Hagen and Wei-Fan Chen. Without their dedication and help, this work would have never been achieved. I am also grateful to the Web Information System group staff for giving me the opportunity to be part of their continuous ambitious research.

I would like to thank Prof. Dr. Benno Stein and Prof. Dr. Norbert Siegmund for accepting my work under their supervision.

My sincere thanks also goes to my dear friend Jale Babajani, for her advices and knowledge and suggestions. There is no way to express the thanks I need to.

I also thank my colleagues for being always around for help when it was needed. Namely: Payam Adineh, Milad Alshomary, Arefeh Bahrami, Alexander Bondarenko, Ehsan Fatehifar, Nikolay Kolyada , and Negin Yaghoubi Sharif. It was a fun and enjoyable working with you.

And finally, last but by no means least, I am grateful to my sibling, my mother and father, who have supported me along the way.

Thanks for all your encouragement!

Chapter 1

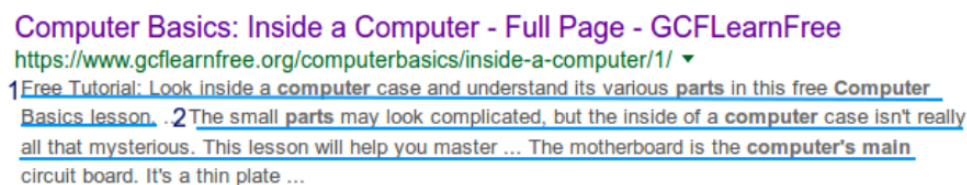
Introduction

Textual information is growing dramatically on the Internet and information retrieval systems have a more difficult task to help users find their desired information as fast as possible. Not only retrieving the ranked list of relevant documents for a user's query is one of the most important tasks of a search engine, also to present these results is extremely important. Making the results page informative enough helps users find the most relevant documents to the searched query faster. The search engines' results usually possess some parts of the document's content alongside with the URL and title of the document, which will give users a preview of the document. This part of document's content is called snippet. Each snippet can have a fixed or variable length which is usually two or three lines extracted automatically from the target document. There have been so far several approaches conducted to generate a snippet, however, there is yet no ideal solution. All of these mentioned approaches suffer from some issues, to describe which, first the approaches have been briefly introduced and then the issues will also be presented. Consequently, our solution will be introduced.

Automatic text summarizing techniques play an important role in search engines to help them select the useful parts of a document for generating snippets. Query-dependent and query-independent are two common approaches in automatic text summarization that search engines are using to generate snippets [18]. Most common query-independent approaches are dealing with identifying important parts of the documents, usually sentences. Furthermore, meta-data description and other external resources, such as web directories which are already generated by the authors or webmasters might be used in this approach [18]. It is also called context-based snippet. Query-dependent approaches generate the snippet based only on the query terms and not any other features of the document. This approach extracts the sentences containing query terms. In the next chapter, these two approaches will be explained

in more detail.

After having investigated some commercial search engines and also having read Google webmaster central blog¹, we found out that they mostly use the query-dependent approach and sometimes a combination of both methods. Usually, they see how relevant the meta-data description and the body text is to the query, and choose either one or a combination of the two. For example, in Figure 1.1 the first fragment is the meta-description and the rest is from the content of the same target page.



Computer Basics: Inside a Computer - Full Page - GCFLearnFree
<https://www.gcflearnfree.org/computerbasics/inside-a-computer/1/> ▼
1 Free Tutorial: Look inside a computer case and understand its various parts in this free Computer Basics lesson. **2** The small parts may look complicated, but the inside of a computer case isn't really all that mysterious. This lesson will help you master ... The motherboard is the computer's main circuit board. It's a thin plate ...

Figure 1.1: Snippet from query-dependant(1) and query-independent(2) approaches

However, these approaches do not always perform very well and might have some complications. For example, Figure 1.2 shows unfamiliar topics to the users cause them to enter a short query. As a result, a bad snippet is generated in the query-dependent approach. In addition, snippet generating methods cannot produce a good snippet when web pages contain less textual information. In these cases, search engines usually rely only on the external resources such as meta-data description or web directories. However, most of them only use one meta-data description for all pages of the website, while each page should have a different description. Web directory, on the other hand, could be considered as a good resource of a web page as well. Web-directory is an online list of websites which categorizes websites by their subjects. There is also a short description of the context of each website which has been written by humans instead of a software. DMOZ, for instance, was the most famous multilingual web directory. Based on a report from Google blog¹, Google used to use DMOZ's content when the description for a page was more useful than the site's meta description or content. But on 17th of March 2017, DMOZ was officially shut down, because AOL refused to further support the project. Thus, the search engines stopped following it. Furthermore, these web directories do not hold a fresh list of all the information on web pages.

As described above, external reliable resources might help solve the mentioned issues, as to that, we presume that anchor fragments can be considered

¹<http://webmasters.googleblog.com/2017/06/better-snippets-for-your-users.html>

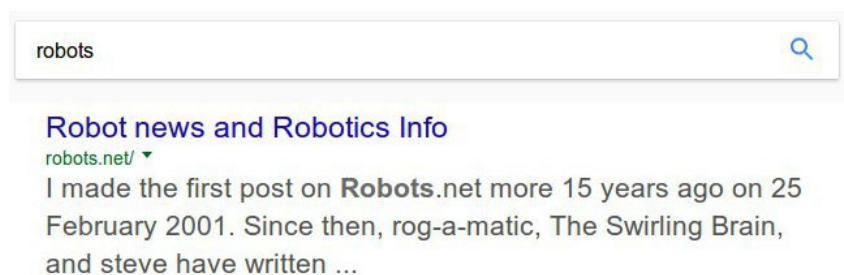


Figure 1.2: Example of a bad snippet: query is only one word and web page does not have meta-data description. That makes the search engine to take the first paragraph of web page and generate a bad snippet.

as another external resource to generate snippets. In this work, this idea will be investigated. What we call an anchor fragment is the sentences around an anchor text. Anchor text is the text of a link which points to the other web page. Usually, these anchor texts are relevant, descriptive or contextual information about the content of the target web page. However, anchor texts cannot be used in generating snippet because they are too short. In that regard, our hypothesis is that the sentences around the anchor text could be related to the web page Figure 1.3.

Software

Docker CE. Installation instructions for Windows, Mac and various Linux distros can be found [\[here\]](#). Select the *stable channel*, and follow the instructions for your particular platform.

Figure 1.3: Example of anchor fragment: “here” is *Anchor Text* which points to “[www.docs.docker.com/install](#)” and *Anchor Fragment* is “Docker CE. Installation instructions for Windows, Mac and various Linux distros can be found here. Select the stable channel, and follow the instructions for your particular platform.” which describes content of the pointed link.

Source: <https://www.uni-weimar.de>

In this thesis, we would show how to extract the anchor fragments, clean them and rank them. The main aim of this thesis is to investigate different methods to rank anchor fragments for each link and then evaluate them to see which one is better. A few studies have been published similar to this hypothesis, however, there is still considerable ambiguity in their approaches. That is why our main focus is not on generating a snippet but studying different methods to rank fragments and to select best of them. Afterward, these

methods will be evaluated to see which one generates better results. Each of these procedures is described in detail in the following chapters.

The next chapter gives an overview of the current status of research in automatic text summarization together with web summarization and then a review on similar approaches to our work. In chapter 3 our approach is redundantly described. First, we explain how to extract the anchor fragments and their challenges. Some of these fragments might contain noise and therefore not adequate to be considered as a snippet. In the next step, filtering and cleaning the fragments is described. And finally, all different ways to rank the fragments are shown. Our experiments and our approaches are discussed in chapter 4. For evaluating the results, we conducted a user study; the method alongside with its results are shown. In the last step, the future steps and conclusion are discussed in the last chapter.

Chapter 2

Background and related works

As pointed out previously, automatic text summarization plays an important role in generating snippets. There are numerous approaches in the field of automatic text summarization, but not many on the snippet generation. Firstly, in order to have a better understanding of automatic text summarization, we will introduce the theoretical background and some important research in that field. Then, different approaches to summarize web pages will be demonstrated. Furthermore, some works very similar to this thesis have been reviewed.

2.1 Automatic text summarization techniques

Automatic text summarization is not an easy task. To summarize a given text, It is vital to understand it properly and generate a shorter and self-contained text. Without sufficient linguistic knowledge, it is difficult for machine learning models to provide a qualified text like humans.

To approach this task, there are two main streams in the domain of automatic text summarization, namely *Extractive* and *Abstractive*. In extractive approaches, the generated text is a composition of important sentences in the original text. On the other hand, the output of abstractive approaches, in general, uses different words to rewrite it [2]. In the following, we will discuss the details and provide examples of this two main approaches.

2.1.1 Extractive text summarization

There are a considerable number of research conducted on this approach. The aim of all the approaches is to extract the important sentences from the original document to show the concept of the document. Luhn [1958] had proposed the first approach to extract important sentences according to the occurrences

of “significant” words in the document. Edmundson [1969], later on, described new methods for screening purposes. Luhn, in his approach, has focused on finding the most significant sentences based on the frequency of keywords; Edmundson’s approach calculates the weight of sentences based on three additional methods:

1. Cue Method: based on certain cue words in the cue dictionary.
2. Title Method: based on occurrences of certain words in the title and the headings of a text.
3. Location Method: based on the location of sentences appearing at the beginning of document or beginning of each paragraph.

Many other studies have been published for extracting significant sentences; such as location-based summarization. This approach assumes that the first and sometimes the last sentences of the first and last paragraph of a document could be good candidates for the summary of that document. Moreover, Abrae Os and Lopes [2002] suggested another method to retrieve most significant paragraphs when the text is long and it is not only on one topic. They used term frequency/inverse document frequency (TF/IDF) to select important paragraphs. Although these approaches seem to be promising, they have several drawbacks:

1. The generated summary might be incoherent since there are no language processing techniques used in it.
2. Long sentences, selected to build the summary, have some unnecessary parts which lead to wasting the space.
3. Sometimes the important information of a text is distributed throughout the document.
4. The pronouns lose their references in most cases.

2.1.2 Abstractive text summarization

Abstractive summarization methods, similar to humans, do not simply extract sentences, but they create a new shorter text which describes the most important information of the original text in a new way. In other words, they use natural language processing (NLP) to understand the text and generate a new shorter text, describing the most critical information of the original text. This

new text should be grammatical and easy to read for the users and it could contain some sentences that are not in the original text, which requires advanced language generation and compression techniques and redundancy removal [22].

According to Saggion and Poibeau [2013], abstractive summarization techniques can broadly be classified into two categories: *Structured-based* and *Semantic-based*. Different methods that use structured based approach are as follows: tree base method, template-based method, ontology-based method, lead and body phrase method and rule-based method. Similarly, methods using semantic-based approach are as follows: Multi-modal Semantic model, Information item based method, and semantic graph based method.

Multi-document summarization, has later been invented as a result of the increase in the amount of information. This approach tries to make a summary from many documents with the same topic [7]. Lin and Hovy [2002] proposed a system to select important content using sentence position, term frequency, topic signature, and term clustering from multiple documents on the same topic.

2.1.3 Web summarization

In contrast to the great number of work that has been undertaken in automatic text summarization, there has been much less work on web summarization. Several years ago, web summarization techniques were proposed for different purposes for example for blind people or handheld devices which have a limitation on the screen [6]. Later, search engines used them to generate the snippet. Different documents' structure and lack of textual information in world wide web, make web summarization a very difficult task. Web pages sometimes contain even more hypermedia information, such as picture, video, and audio than text. Thus, they should be summarized in a different way.

Generating snippets which, nowadays, is one of the most important features of a search engine relies on automatic text summarization. Query-dependent and query-independent which are the common approaches to generate snippets are described briefly in the previous chapter. Both query-dependant and query-independent summarization are categorized in extractive text summarization since they do not generate any new text. Some of the issues of these common approaches are addressed in the previous chapter as well.

For query-independent approach, search engines simply use common strategies of extractive summarization, such as location-based summarization (first or last paragraph) [1] or Luhn's approach [17]. One major issue of these approaches is dealing with web pages that have information from diverse sources. A blog, for instance, has usually some posts in each page which are not related to each other and might be completely different. One post could be about

the weather and the other about politics. There are also a few other studies conducted for this purpose. For instance, all the words appearing in the title and the headings of a web page are considered as important. Therefore, the weight of a sentence is computed based on the frequency of these words.

On the other hand, in query-dependent approaches, snippets are generated based on the location of the query terms in the documents. First, they search for the complete query in the document; if it is found in a phrase, that phrase will be shown as the snippet and if not, some fragments of the document which have some of the query terms will be selected [18]. Natural language processing techniques in query-dependent approach are very useful to find the best part of the document which is readable, short enough and very similar to the query.

As shown in the first chapter, external resources play an important role when a search engine cannot rely on the content and meta-description of a page. There we have also shown that external descriptions of a web page such as descriptions in web-directories and anchor fragments could be considered as acceptable external resources. The concept of using anchor text is not new in web information retrieval, especially in crawling and ranking. Search engines, for example, follow the links to find the new pages and to rank a page. The frequency of the pointed link to this page increases the score of the page.

To the best of our knowledge, current commercial search engines have not used the idea of anchor fragments so far. However, there have already been some works conducted using anchor fragments for generating a snippet. Amitay and Paris [2000] proposed the concept of using the semantic context of a link to generate snippets. They mentioned that if the author of a web page adds a link of another web page into the main content of the page, the anchor text of this link is most probably relevant in the overall context of his web page. They used InCommenSense system with the purpose of generating snippets for search engines. Their idea relies on the structure of hypertext and the way people describe information of the links. Using this system (InCommenSense), for each document, they were able to collect all the pages containing this link. To do so, InCommenSense used a query type ("link: URL") in a commercial search engine to fetch the pages having links to the document and then extract the surrounding paragraph of the link. They categorized the paragraphs into four different categories:

1. Paragraphs which have only one anchor text and begin with the same anchor text.
2. The paragraphs which have only one anchor and it is neither at the beginning nor at the end of the segment.
3. Paragraphs which have only one anchor text but it is at the end of the

Segment.

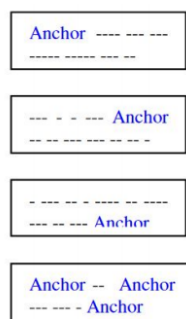


Figure 2.1: Different types of anchor paragraphs

Source: Amitay and Paris [2000]

According to their study of Amitay [2000], paragraphs in the first category are most probably about the context of the anchor text. To choose the most accurate paragraphs among those which have the first pattern, they proposed a filtering system with more than 60 features, such as length, punctuation, use of verbs, the position of verbs, etc. Afterward, they compared the results from more than 700 people with AltaVista Style and Google Style to evaluate the generated snippets. A snippet in AltaVista style is generated from the top X words of a document. While in Google style, phrases containing the query terms are used as a snippet. On average, people preferred the results generated by the InCommenSense system from anchor texts.

Later, Delort et al. [2003] pointed out some issues of Amitay and Paris's approach and improved it by using an algorithm which to select a sentence about the same topic covering as many aspects of the web page as possible. Partiality and topicality are some of the issues that they have mentioned about Amitay and Paris's work. Partiality is the links which point not to the document entirely, but to a part of the document. Topicality is the paragraphs which are about the content of a target, not the context. For instance, a link and its surrounded texts points out a specific piece of news from CNN, and not the context of the CNN. The purpose of their work was to generate context base snippet though. They generated the snippet for each document as follows:

- Split a document into fragments and then extract all the fragments that have the links to the document and they want to generate the snippet for.

- Filter the fragments.

Remove the sentences containing more than 7 links, or with a total number of words larger than 50 or less than 3.

Use part-of-speech (POS) tagger to remove sentences with either no verb or more than 4 unknown tags.

Keep only words tagged as adjectives, verbs or nouns.

- Measure the overlap between each sentence of the context and text of the web page.
- Rank them based on this measurement and keep those which have the highest value in summary.

As research proceeded, Wang et al. [2007] used sentences from anchor texts and the content of a page to show the feasibility of using machine learning approaches for sentence selection in generating a snippet. They categorized sentences to content and context. All the sentences containing a link to the page are extracted and categorized in context. The models were SVM and ranking SVM which use “Relevance” and “Fidelity” as the features. The occurrences of query terms in a sentence are defined as the “Relevance” feature and the properties of the sentence such as location and format are defined by the “Fidelity”. For the evaluation, they used a dataset of 175 documents of 10 random queries from TREC (Web Tracks TREC-2003)¹. The content of each document and their context (anchor text phrases) were extracted. Then they summarize these 175 documents manually by two human evaluators. Intrinsic evaluation which introduced by Robertson and Walker [1994] and tests the summarization system in itself was adopted to measure the precision, recall, and F1 score. After all, their results showed ranking SVM outperform SVM classifier.

2.2 Summary

In this chapter, we gave an overview of related work in the field of text automatic summarization and reviewed some works very similar to this study.

According to this research, since 2007 when Wang carried out a research on anchor fragments there has been no more research on using anchor fragments for the purpose of generating a snippet. Furthermore, a serious weakness of all of these three works is that they have focused on generating snippets query-independent. Their results might have been more interesting if they would have taken query into account in their approach as well.

¹<http://trec.nist.gov/overview.html>

Chapter 3

Proposed approach

This section gives you an end-to-end description of our proposed approach from mining anchor fragments to ranking them.

The approach is divided into four steps which are explained below. The first step demonstrates how to extract the anchor fragments from web pages. Then, we explain how to remove noisy data and will show some strategy to reduce the identical information. In the third step, it is described how to optimize the system by indexing the fragments and their attributes. Finally, our strategies for ranking the extracted fragments will be discussed. The most noticeable difference between our approach to the similar approaches is in our strategies to rank and retrieve the top fragments. In order to fully understand our desired system, primarily the architecture of the system is expressed, followed by a more detailed description of each part.

3.1 The architecture of proposed system

In this section, we show how our suggested system looks like. The goal in our approach is to provide a system in which its inputs are a link and a query related to that link, and its output is a ranked list of anchor fragments. These outputs should be readable and could be considered as a snippet.

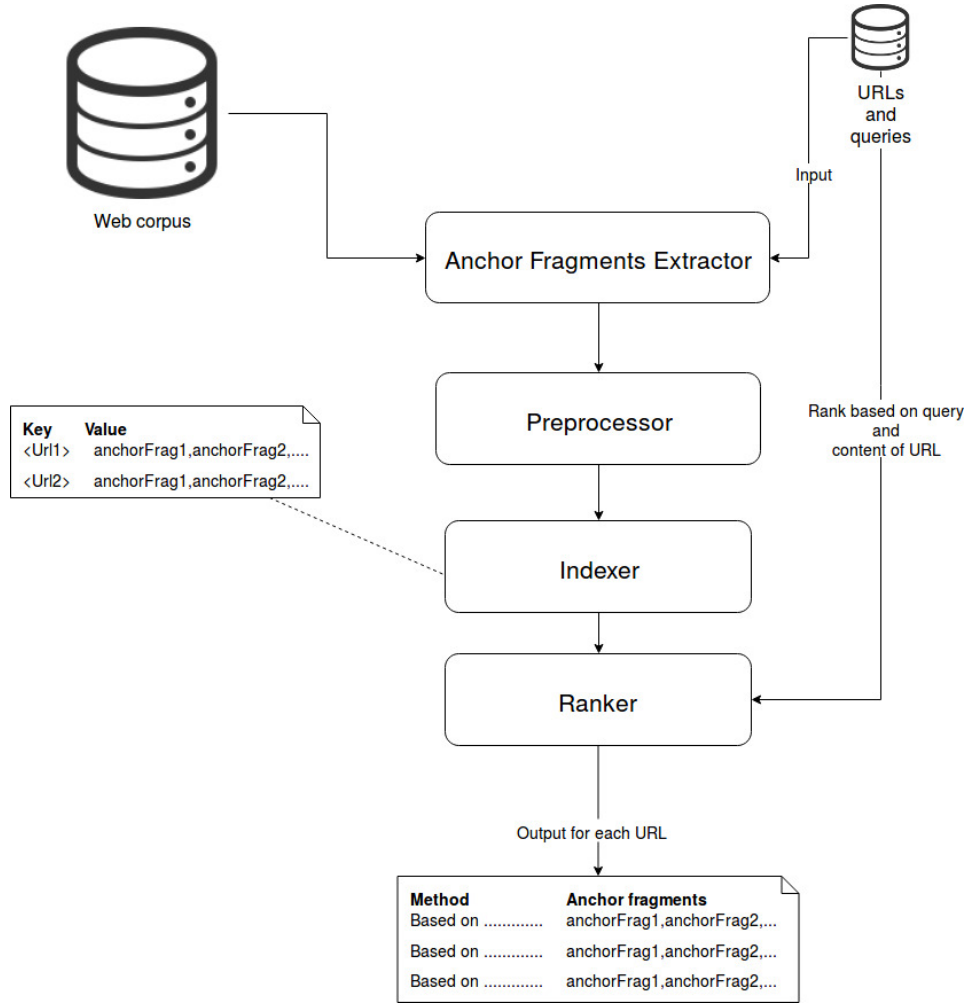


Figure 3.1: Architecture of the desired system

Figure 3.1 provides evidence that the system has four different parts, namely: *Extractor*, *Preprocessor*, *Indexer*, and *Ranker*. In order to make each part more clear, an example in a real scenario will be described in the following. Assuming that a search engine, in its first page showed 10 documents for a search query “Wind power”. The search engine wants to use our system in order to make a snippet based on anchor fragments for the first link in its results. The inputs will be the link of the first document with a searched query which is “Windpower”. In the initial stage of the process, the *Extractor* will take the link and go through the corpus of web documents. This corpus contains the raw source of web pages written in HTML. Next, the *Extractor* tries to find the documents having this link. Afterwards, it extracts the surrounded text (fragment) of the found link from each document and sends them to the *Preprocessor*. The *Preprocessor* has some policies to remove or merge some

parts of the fragment or the whole fragment. After anchor fragments have been cleaned by *Preprocessor*, Indexer will index the link as key and all the fragments as values. Fragments are now ready for the next step. Finally, the *Ranker* will give each fragment a score in seven different methods, sort them according to this score and will send them to the output.

3.2 Extracting anchor fragment

Anchor fragment mining is the first step involved in extracting a textual fragment from an HTML document. In this section, methods and challenges of extracting anchor fragments will be discussed.

The surrounded text of a link is the text which we guess is about the context of a link. While the first challenge is to find out how many of the sentences could be about that link, the second challenge is how to extract them among the unstructured text of a web document.

After having found an anchor text in a text, there are two methods to define the scope of an anchor fragment. The first way is to take some sentences before and after the anchor text which we call *Surrounded text*. The second one is the whole paragraph of anchor text, named *Paragraph*. Both of them, have some advantages and disadvantages, will be described in more details in the next section.

3.2.1 Surrounded text or paragraph

To deal with Surrounded text method, we first need to tokenize the text into sentences and then we should know the number of surrounded sentences. A sentence is a sequence of words which is finished with one of these special characters “? ! .”. However, there are some cases where these characters are not always playing the role of the sentence break. For example, decimal point, email address, or when there is a sentence in quotations inside the sentence.

To find out the number of surrounded sentences a maximum number of sentences should be defined. In that regard, sentences closer to the anchor text might be more relevant to it. Furthermore, the maximum number of sentences should sound rational. For instance, the possibility of the relevancy of the 10th sentence after the anchor text to the anchor text should not be high. As soon as we define this number, sentences, which are not related to the link, could be removed from the fragment. Since we do not incorporate a highly secure algorithm to detect the relevancy of the link and a sentence, some of the suitable sentences could be removed through this method.

On the other hand, for the *Paragraph* idea, we only need to tokenize the text into paragraphs and retrieve the paragraph having the anchor text. A

paragraph is a sequence of sentences which is finished by a line break. There are some hidden characters inside the digital texts which make this task unchallenging for us. Depending on the operating system, the line break is defined by “\n”, “\r”, or a combination of both.

Although the *Paragraph* method is very simple and could be more relevant to the anchor text, it has some drawbacks. This method restricts the scope to only the sentences inside that paragraph, while in the web, the users’ writing behavior is different. *TL;DR* (too long;did not read) is an Internet slang, indicating that the user will ignore the text in case it would be too long. For this very reason, web pages’ authors try to make the text as short as possible. For example, instead of a long paragraph, they break the text into a list and exhibit it with bullet points. Furthermore, in some documents, anchor text is in the title of an article and the following paragraph is related to that. In this case, only the title will be retrieved and the interesting paragraph will be skipped. In that regard, there was an experiment carried out, in order to figure out the probability of the mentioned issue.

Experiment

As expressed in the former, the following experiment was conducted to grasp a better understanding of the paragraph approach. Our interest was to find the percentage of relevancy of a paragraph to the context of its embedded link. Furthermore, we wanted to see how often it is possible to have an anchor fragment in the headings of a web page. We randomly opened fifty different pages using “www.uroulette.com”, a website to open a random link. Once the experiment was completed, the following report was delivered:

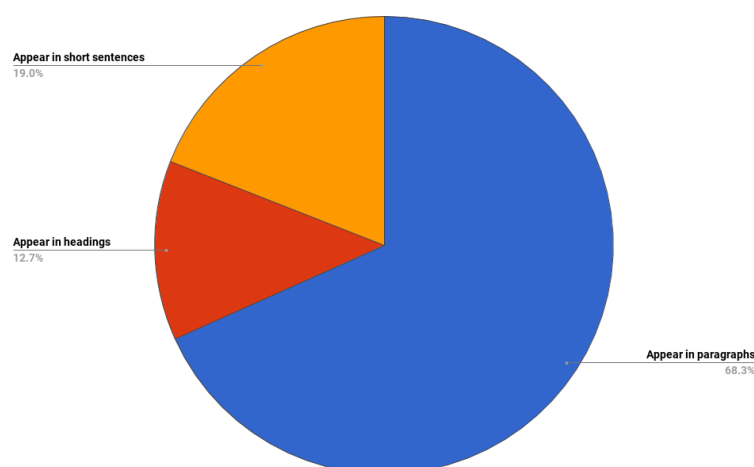


Figure 3.2: Percentage of occurrences of a link in paragraphs, short sentences, or headings

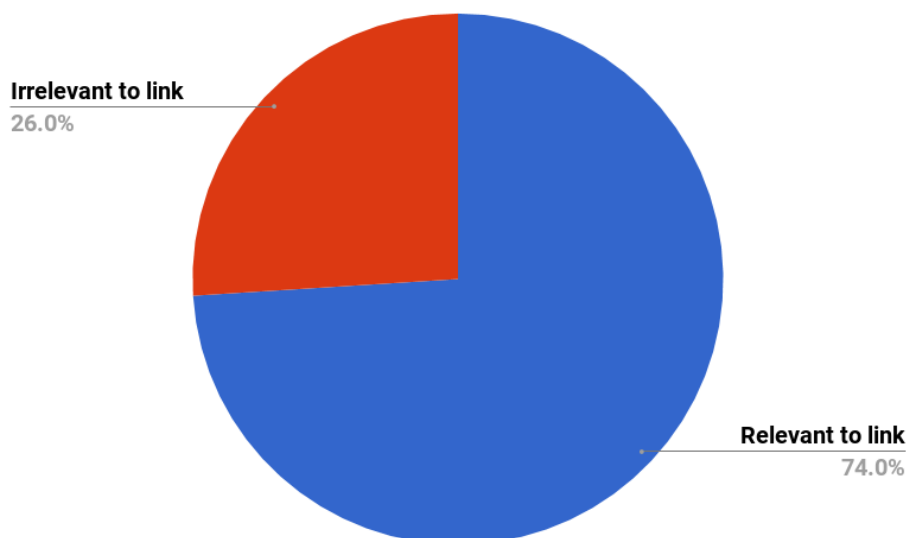


Figure 3.3: Percentage of the relevancy of paragraphs to their embedded links

As suggested in Figure 3.2 on average the possibility of the occurrences of links in paragraphs is high. They are most probably relevant to their embedded link (see Figure 3.3). This test revealed that *paragraph* method is reliable enough to be used in our system. Furthermore, this method is relatively challenge free. As a result, the *Paragraph* method was used in the *Extractor*.

3.2.2 Extractor

Distinguishing a paragraph and extracting it in a web page are not as uncomplicated as it sounds. Paragraphs can be tokenized easily with the line breaks when we are dealing with a document, made of pure text, while web pages are written in hypertext markup (HTML) format, making the task more challenging. The simplest solution is to change the HTML to a readable text, the same as what a user sees in a browser without any hypermedia. To do that, the content of each node should be extracted and added to a container. To make this easy, we should parse each HTML as a hierarchy tree structure using document object model (DOM)¹. After parsing it with DOM, each tag of HTML represents a node of the tree. Therefore, we can go through each node and its children to find the content and extract them. Furthermore, some elements need to change to line breaks and some of them to tab spaces. For instance tag `
` always means line break. Tag `<td>` and sometimes tag

¹http://en.wikipedia.org/wiki/Document_Object_Model

`<div>` could be tab space. Since our goal is to find the surrounded paragraph of a text, we change both cases to line breaks.

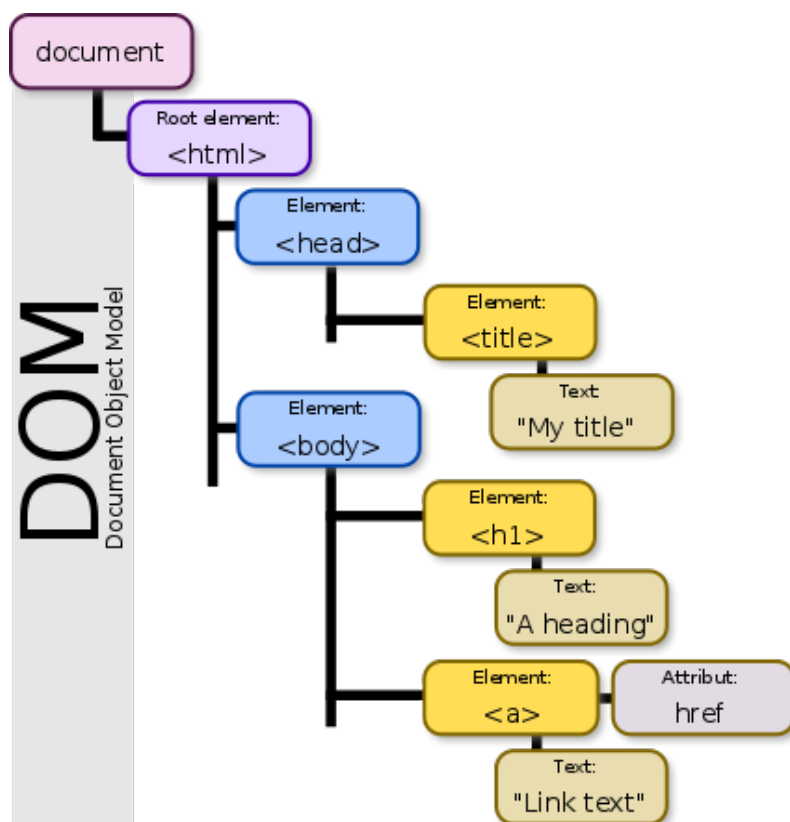


Figure 3.4: Example of DOM hierarchy in an HTML document

Source: From http://wikipedia.org/wiki/Document_Object_Mode

When *Extractor* is seeking for pages with an interesting URL, it should end and extract all the anchor fragments pointing to the host of that URL. For example, if the URL is “www.example.com/blog/1020”, it should extract all the fragments pointing to “www.example.com”. On the other hand, If we restrict the scope of *Extractor* to the exact URL, we would probably end up with very small anchor fragments or even nothing; because web authors usually drop some general points about the other web pages and rarely about a specific page on that website. Since the structure of a website is hierarchical, most of the times, all the web pages of a website have the same context. The home page of a Web site often promotes a general idea about the context of the site. Therefore, the *Extractor* should extract anchor fragments from all the web pages having the same host as the interesting URL has. However, the exacted URLs within fragments should be preserved for the Ranking algorithm.

Although dealing with HTML to extract paragraphs is challenging, it also benefits from some advantages. In the Ranking sections, we will explain that some of the attributes of the text are very useful and important. For instance, the format of the text such as being bold or italic, the location of text in the document and etc. A self-contained text cannot hold this information while in HTML they are added by some additional tags or attributes of a tag. This is precisely why, during the parsing process, we should preserve this kind of information with the same format (HTML) or changing them to another format, which means we will be able to parse them more easily later. For example, a tag `` or `` makes its embedded text bold, meaning that this part of the text is more important than the other parts. Or, embedded text within tags `<h1>`, `<h2>` and `<h3>` shows this text is a fraction of an article.

Different steps of *Extractor* are elaborated in the following:

- Extracting the host name of the interesting URL
- The HTML of Webpages having the host name should be parsed with DOM
- Starting from the root nodes of node BODY
- Going through each node to the lowest level
- Changing nodes "div", "p", "br" "li", "ol" and "td" to line breaks which in text file it is "\n"
- Keeping important tags
 - "b", "strong": bold
 - "h1", "h2", "h3": headings
 - "em": emphasized text
 - "mark": marked text
 - "i": Italic
 - "a": link
- Generate the output and return it
- Table 3.1 shows the suggestion of *Extractor*'s output

Table 3.1: Suggested output for the *Extractor*

Key(URL)	Target	Attributes
Free content and city guide in your site...	mytt.tv/sitemapcust.html	h2:city guide
Medications that may cause heartburn...	heartburn.about.com/cs/longt...	b:heartburn
bookmooch is a community for exchanging...	es.bookmooch.com/detail/999...	

3.3 Preprocessing anchor fragments

Preprocessing plays a very important role in our system. It removes the unreadable fragments and improves their quality. As we are dealing with unstructured data from web pages and from unknown authors, it is very probable that they are vastly noisy and unclear. Furthermore, during our experiments of parsing the HTML to extract the paragraphs, we ended up with a great number of meaningless fragments.

There are a great deal of reasons for making the web pages noisy. For instance, a poorly written text could be found in web pages, blogs, discussion forums, users' comments, and wikis. Furthermore, web pages usually contain some information that may not be interesting for the user, such as information in the footer and navigation menus, or even advertisements. Gibson et al. [2005] in their study has provided evidence that 40% to 50% of the information in the web is noisy and this volume is growing at a rate of between 6% to 8% per year. This noisy data makes our extracted anchor fragments noisy or in the other word unreadable. Figure 3.5 is an example of an anchor text in a noisy web page.

TRENDING: [MITT ROMNEY](#) | [RON PAUL](#) | [RICK SANTORUM](#) | [ELECTIONS 2012](#) | [NEWT GINGRICH](#)
February 10, 2012

Figure 3.5: The extracted text of this anchor fragment will be "TRENDING: MIT ROMNEY | RON PAUL | RICK" which is a meaningless phrase.

Discovering a pattern or using machine learning are some of the approaches to preprocess data. It is well known that an annotated dataset is needed to apply any algorithm of machine learning for preprocessing. Regarding our time

restriction and budget, first, we investigate to identify a pattern to find out if it could be useful or not.

To establish a pattern, we need to manually check some of the data and define some rules. Therefore, we manually categorized 4000 of the extracted anchor texts from different links into two categories: *meaningless* and *meaningful*. An anchor fragment is meaningless when it is unreadable or it is not fit enough to consider it as a snippet. We collected some additional labels for meaningless fragments to determine the reasons for making them meaningless. Therefore, each fragment could incorporate more than one label. Finally, we came up with 3057 meaningless fragments (around 75% of 4000) and seven different reasons that make them meaningless, which will be explained in the next section.

3.3.1 Features of meaningless fragments

As formerly demonstrated, we did an experiment to find some pattern which makes the anchor fragments meaningless. As a result, five different reasons were identified, which will be explained in more detail in this segment.

Figure 3.6 illustrates the number of meaningless fragments for each reason. Although the number of meaningless fragments with the label of *No Sentence* is the largest issue (2365) which is around 24% of all, considering *Too Long* and *Too Short* jointly as one label, which describes the length of fragments, will account for the largest one (together 2551 around 26%). *The number of links*, fragments having more than one link, has the next largest amount of meaningless fragments around 20%. Fragments including stop phrases, have also similar amount to the number of links label, around 18%. The chart shows two more labels which have the smallest numbers: *Different Language* around 3% and 8% for *Contain Date Time*.

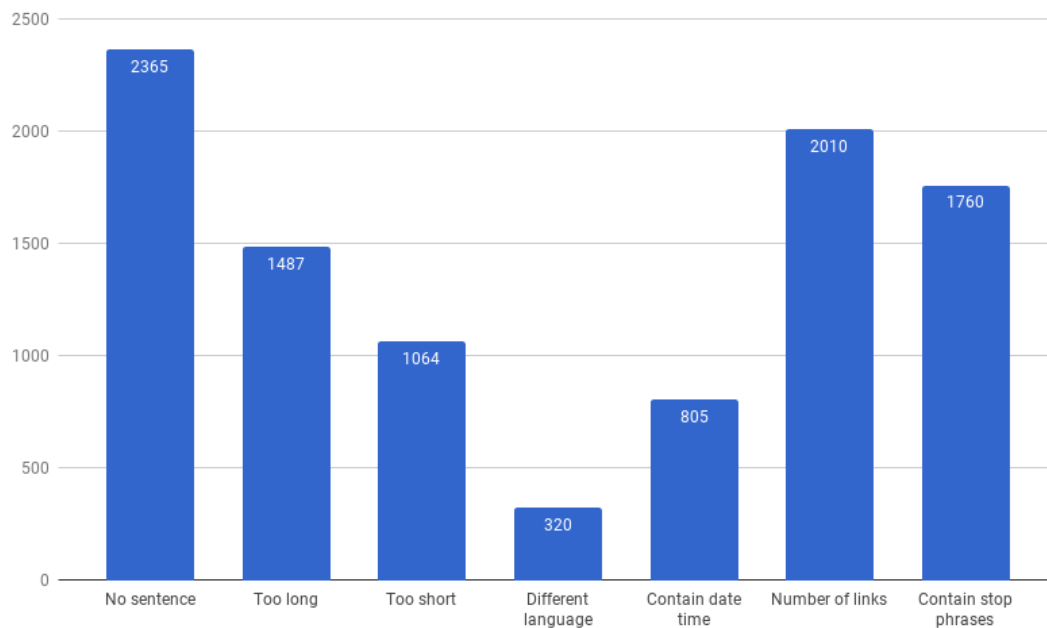


Figure 3.6: Reasons which make the anchor fragments meaningless

Although the number of meaningless fragments with label of *No sentence* has the largest number (2365) which is around 24% of all, *Too long* and *Too short* could be combine to one label describing the length of fragments and be the largest one (together 2551 around 26%). *Number of links*, fragments having more than one link, has the next largest amount of meaningless fragments around 20%. Fragments having *Stop Phrases* has also similar amount to *Number of links* label, around 18%. The chart shows two more labels which have the smallest numbers: *Different Language* around 3% and 8% for *Contain date time*. Each of these reasons is explained in the following:

Length of fragments

As evident in Figure 3.6, short and long sentences are the most likely causes of making fragments meaningless. Due to the unstructured data in the web, there are some possibilities to have some fragments containing text with no end of sentence punctuation. HTML errors, such as unclosed tags, are another reason letting the system extract a very long sentence. For these reasons, our system might extract very short sentences.

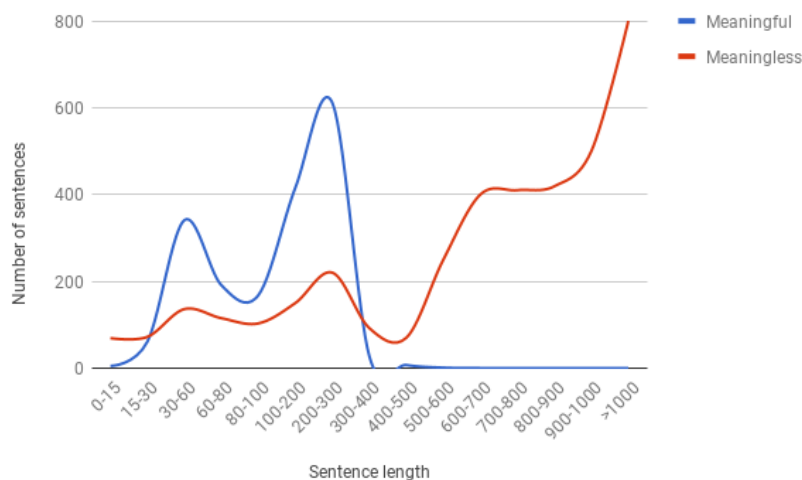


Figure 3.7: Reasons which make the anchor fragments meaningless

Figure 3.7 provides clear evidence that, sentences with the length of more than 400 and less than 30 characters are always meaningless. However, as there were not enough annotated fragments, we did not completely rely on this report.

Fragments without sentence

A big portion of meaningless fragments had a label of *No Sentence*. This label was defined for the fragments which do not have the structure of a sentence. Table 3.2 shows some anchor fragments without any sentences. A sentence is a set of words, containing at least a subject and a verb, that expresses a thought. A simple pattern is to tokenize sentences and then check if each one has the structure of a sentence or not. In our case, the occurrence of a verb in the sentence could be enough regardless of expressing the thought (main clause). POS tagger, which is a well-known procedure in NLP and a framework or piece of a software to assign parts of speech, such as *Noun*, *Verb*, *Adjective* and etc, can be used to identify verbs in a fragment.

Table 3.2: Some examples of extracted fragments which does not have sentence structure

- author michael pollan on this topic here
- deepak chopra audio book
- Brian Rudy Recent blogs 10 Feb 2012.

Fragments having links

Anchor fragments benefiting from more than one link are another reason causing the system to extract meaningless anchor fragments. Links in the navigation bar, footer or link farm web pages are the most common source of this problem. A link farm is an expression for any group of web pages all linking to one another. They do that for the purpose of increasing their score for search engines. Nowadays, search engines are improved enough to recognize such pages and consider them as Spam pages although there are still lots of them on the Internet.

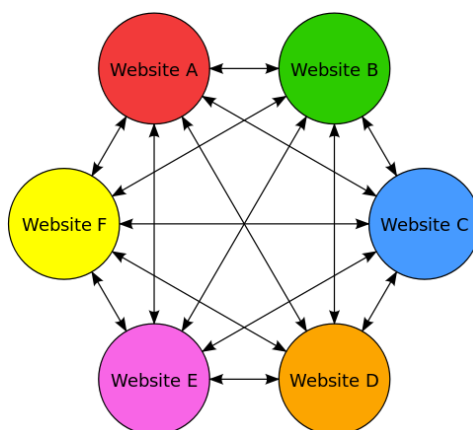


Figure 3.8: A diagram of a link farm. Each circle represents a website, and each arrow represents a pair of hyper-links between two websites.

Source: www.wikipedia.org/wiki/DLink_farm

Fragments having Date Time

The occurrence of a date time in a fragment does not necessarily make it noisy or unreadable, however, it does make it a bad fragment to consider as a snippet. Therefore, we should only add a tag to these fragments to reduce their score in the ranking process. “comment about robots.net on April 2, 2012”, “28 March 2011 by lonely planet” or “September 8, 2011, Szev report” are some examples to support this idea. Figure 3.9 provides evidence that, one of the big sources of these kinds of fragments is comments or forums.

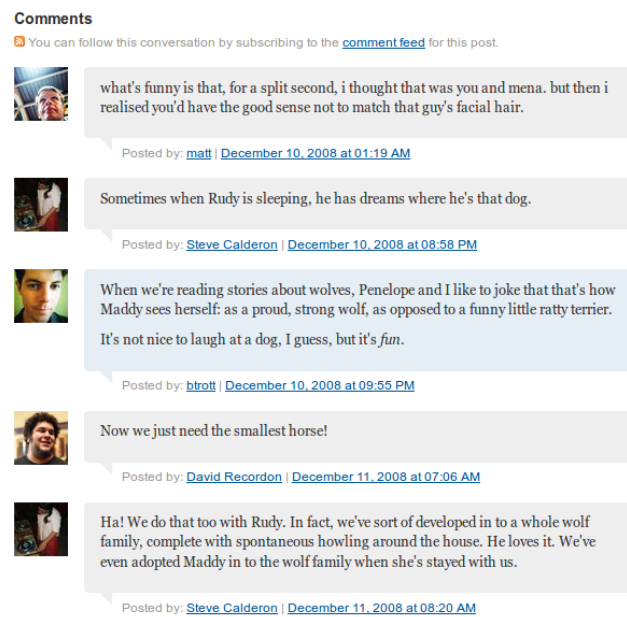


Figure 3.9: Example of links with date time
Source: www.ben.stupidfool.org

Fragments in different language

Anchor fragments which have a different language than the language of their target web page (the content of their embedded link) are considered as meaningless fragments. For instance, when a web page is in English, it could have some anchor fragments in any other language. Since the goal is to make a snippet about this web page and it is in English, there is no need for the anchor fragments which are not in English. Figure 3.10 provides an example on an anchor fragment extracted from a Persian language blog which points to a news in bloomberg.com. This piece of news is English, while its anchor fragment is in Persian.

[confessions of an Instagram Influencer](#) رو نوشته و اعتراف کرده که چطوری از یک آدم معمولی مثل من و شما تبدیل شده به یک آدم مشهور اینستاگرام: یکی از تاثیرگذارها! قیافه مکس عادی و غیرمدل مانند است و تیپش کاملاً معمولی ولی برای نوشتن این مقاله در مسیر مشهور شدن پیش رفته. تو این مقاله از سرویس هایی می گیم که برای شما فالوئر واقعی و الکی می یارن، از آژانس هایی که عکسها تون رو می گیرن و بهتون لباس می دن و .. توی این شماره به عمق آدم مشهورها می ریم. با ما باشین!

Figure 3.10: Example of a fragment in different language than its embedded link
Source: www.jadi.net

Fragments with stop phrases

Stop phrases, which are the most common phrases appearing in the fragments, also showed a great portion of meaningless fragments. Regarding the Consistency rule for web designers, they try to have very common patterns in their content which are very familiar to users. For example, instead of an informative text for a link, they use common anchor texts, such as, “Click here”, “next page”, “«”, “»”, and etc., which will lead to the appearance of stop phrases in the fragments. According to our experiment, occurrences of such stop phrases in fragments was a reason to make a fragment meaningless. Therefore, we need to identify them and try to remove them from the fragments.

There is a simple solution, which is partially automatic, to identify stop phrases. By tokenizing the texts to N-gram Word and counting the frequency of each token, we can make a list of most common phrases. Because the number of very long stop phrases will not be very high, we can restrict the scope of tokenizer to 8-gram. Now, if we run this algorithm on our extracted anchor fragments corpus, there will be a great number of good common phrases among them, which can not be considered as stop phrases, such as, “I am”, “we are”, “he should”, “if you want to”, “on the other hand”, “it will be a”, and etc. For this very reason, this part could not be applied automatically and has to be checked manually afterward. As soon as we sort the list from high to low, we should check them one by one and take the stop phrases to a new list which is called stop phrases list. During our experiment, which will be explained in the next chapter, we made a list of stop phrases containing 200 of them, some of which are illustrated in Table 3.3.

Similar fragments, Redundant information

Removal of fragments redundancy is one of the crucial steps to optimize the system and make it ideal. During our experiment, which will be explained in the next chapter, we came up with a great number of fragments which were either identical or very similar to each other. Furthermore, some of these fragments could be generated with the common approaches of snippet generation (query dependent and independent); while, we need fragments which are different regarding the goals of the system. In the following, some ideas have been proposed regarding each.

There are lots of reasons causing the system to generate similar fragments. For instance, an advertisement for a website has usually a link and system would extract it as an anchor fragment. Since this advertisement has the same content in different web pages, the system will extract all of them with an identical content. In addition, Sidebar, Footer, and Header of the website are usually identical for all of its web pages. As a result, links which are in these

Table 3.3: Some examples of stop phrases extracted in our experiment

you are here home	download read buy online	for more information
for more information on	for more information about	read the full article
below are links to	listed below are links	listed below are links to
links to weblogs that	to learn more about	for more information visit
can be found at	all text is available	the rest of the
click here to	read the full	reviews on
to share with friends	click here for	find out more about
share with friends	more information about the	learn more about the
click here to read	to find out more	click here for more
read the rest of	read the full	you are here home

parts are the other resources causing the system to generate similar fragments.

To avoid having fragments similar to the snippets which might be generated with the common approaches (query dependent and independent), there is a simple solution. As the common approaches rely on the content of web pages, we should remove those of the extracted anchor fragments which are very similar to the content of their target or their meta-data description. To do so, we should extract all sentences of the target of each link altogether with its meta-data description. Then, the fragments which are very similar to these sentences (query dependent and independent) should be removed and the similarity between each fragment and these sentences should be measured.

To remove the rest of redundant fragments, very similar fragments should be removed and fragments which differ in some parts should be combined to one fragment. You can see a good example in the Figure 3.11.

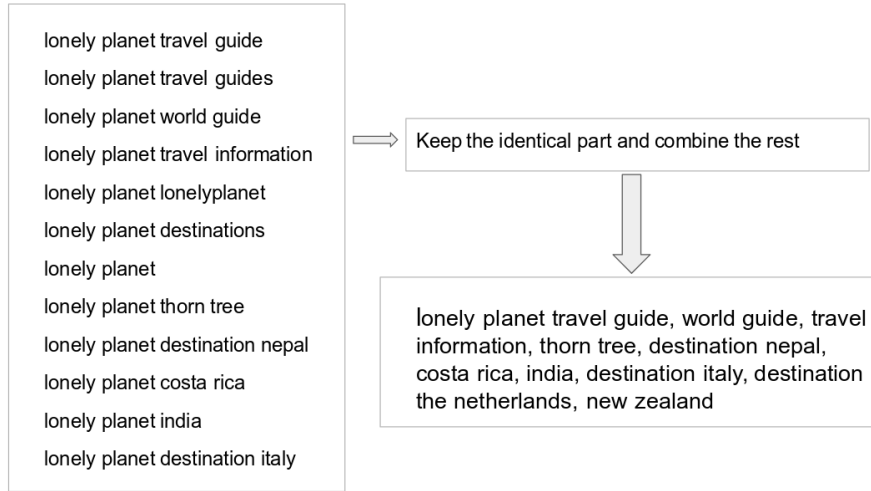


Figure 3.11: A good example of combining very similar fragments

Although similar fragments are not valuable in our system, we may need them when we are ranking the fragments. One of our ranking fragments methods is based on the frequency of similar fragments, which we will explain it in the next chapter. Since this method only needs the number of similar fragments, we may only keep one of these similar fragments with the number of its frequency. This should add to one of the fragment attributes and later in ranking part, we can use this number. As already explained, similar fragments with one host name are not valuable and therefore, only fragments with a different host name should be counted.

To find and remove very similar fragments, they should simply be compared with each other. We define “very similar fragments” as fragments which are almost identical to each other regardless of case sensitivity of the characters, stop words, non-alphabet characters, such as, punctuation marks and numbers. For instance, this fragment “The company, which Hardwick co-founded in 2012, did not respond” should be identical to this phrase “Company, which hardwick co founded didn’t respond”. Thus, as soon as we change all the characters to lowercase, we should remove all of their stop words and then remove non-alphabet characters. Moreover, the Stemming algorithm can be applied to the fragments to remove more redundant fragments. The Stemming algorithm identifies the root or stem of a word. For example, the words “argue”, “argued”, “argues”, “arguing”, and “argus”, all can be stemmed to “argu”. In Table 3.4 a good example is demonstrated.

Table 3.4: All of these fragments will be identical after applying Stemm algorithm and removing stop words

fragment	After stemm and stop words removal
Here you can find chinese new year	Find chines year lesson plan activ
Find chinese new year, lesson, plans , activities and more here	
Find the chinese new year, lessons, plans and activities	
You can find more about the chinese new year, lessons, plans and activities here	

Combining the fragments which have an identical part is a very complex process. To find the identical part of each fragment, each fragment should be tokenized to N-gram Word. Then each token of each fragment should be compared with all the other tokens. To make this a little bit easier, we can limit this algorithm to find the overlapping of each token with fragments when it occurs only at the beginning of fragments. As a result, each token should be compared to the fragments and not to the tokens of fragments, resulting in decreasing the computations to a high extent. Furthermore, since there will be too many identical phrases up to three words, each token should at least have four words. Table 3.5 pictures some of the combined fragments.

Table 3.5: Some example of combined similar fragments

Result of combination procedure

- Book reviews and compare prices for carruth hayden, chatwin bruce, costikyan greg, sterling bruce, burnejones edward, crane hart, cavelos jeanne, cadigan pat, carr emily, chalker jack

- Website for favorite baby names, boys name asher, popular girls names , appellation mountain, nameberry nine, unusual names for boys, baby name indiana, unique names for boys, jewel names, top unisex names, names that end with x

- More info about anaphase i, anaphase ii, gene

3.3.2 Preprocessor

As already described, we are seeking for a pattern to distinguish the fragments which are not suitable to be considered as a snippet. This section explains what exactly this pattern is. In the previous chapter, we extracted some features which make the anchor fragments Meaningless. We need to investigate more on these features because the number of fragments in our experiment was not high enough to rely on this report.

Some features only make some parts of the fragments noisy. For instance, if a fragment contains stop phrases, it does not mean that it is a bad fragment and by omitting that part we can save this fragment. Therefore, fragments having data time, stop phrase or different language should not be removed completely, we should only remove that part containing these features. When we remove a part of a fragment, we should check it with the length threshold which has been defined in the Extractor in order to make sure if it can still pass this condition, otherwise it needs to be removed.

Very short fragment, less than 30 characters, was another reason of making fragments meaningless. The goal of our system is not to make the snippet but to suggest some fragments which are the good candidates to be considered as fragments. A short fragment can be combined to another one and generate a good fragment later. Consequently, our threshold should only restrict the maximum length of fragments. Although fragments without sentence structure could be one of the issue making fragments meaningless, there could be some fragments without any verb but a very good candidate to consider as a fragment. For instance, “An Unpublished Essay on the Trinity Jonathan Edwards (1703-53)” or “A Brief Declaration and Vindication of the Doctrine of the Trinity” are two long fragments which we found later on the Internet. This kind of phrases are usually written for a title of an article and when they contain a link, our system will extract them as anchor fragment which does not have the structure of sentence but it is readable and very good to consider as a snippet. Accordingly, we did not apply this feature to the pattern for recognizing the meaningless fragments. Our suggested features to identify meaningless fragments are elaborated in the following:

- Fragments which their length are more than 400 characters should be removed
- Fragments which have more than two links are inside them should be removed
- Stop phrases should be identified and removed from the fragments
- Date time patterns should be identified and removed from the fragments

- Language of the target of each fragment should be identified and each parts of fragment which is in different language should be removed

3.4 Indexing anchor fragments information

In order to optimize speed and performance in our system, we will show how indexing can be used and what information we need to index in this part. Indexing is a technique in the information retrieval to access the data quickly without scanning all the available data. As the ranking algorithm in our system happens online, when a user enters a query, the system should retrieve a ranked list of the fragments as fast as possible.

Although the amount of anchor fragments for each link is not huge, the ranking algorithm could be very complex, consequently causing the system to work extremely slowly. For each request, the ranker needs to scan all fragments, extract those belonging to the request. Then it should extract some information from each fragment and finally rank them based on its algorithms. However, some parts of ranking algorithm rely on the user's query and therefore, ought to be done online. Hence, all the steps before the ranking shall be processed and all information we need in ranking could be prepared and indexed. Consequently, the ranking algorithm will process the information faster.

The required information in indexing can be divided into two parts. The first part is for all the information belonging to anchor fragments, such as text attributes (being bold or in the heading), embedded links and the frequency of the similar anchor fragment. The second category keeps the information from the target of anchor fragments together with their home pages. Main content, First paragraph, Important sentences, Meta-data description, Title, and headings are some of the information which can be extracted from the target or home page of each link and be indexed. In the next chapter, we will explain how to extract this information, and the reason that we need them one by one.

3.5 Ranking anchor fragments

In this section, the different strategies will be discussed to rank the fragments, namely the most important part of the system. When the last steps have been completed, we have some fragments for each URL which are meaningful and readable but would not be necessarily considered as a snippet. As it is not easy to know which fragment is the best candidate for the snippet, the methods which can help us choose the best candidates, should be investigated.

We describe how to sort suggested snippets based on their score of similarity to six different resources. Furthermore, another approach will be explained which is based on the frequency of the similar fragments from different resources.

Since six of our suggested methods are based on the similarity between two texts, our suggested algorithms to measure the similarity will be explained at first.

3.5.1 Computing the similarity

Measuring similarity in text retrieval information is usually between a short text and a long text (document). In our system, we need to compare two short texts, fragments up to 400 characters and the extracted text up to one paragraph. Therefore, we should use different methods to measure the similarity. Term Frequency Score and ROUGE metric are our two measurements, which we suggest to use. We have proposed to use Term Frequency, in the case one of the texts is too short, up to five words and Rouge metric when they are longer. These two methods are described in the following along with their algorithms:

Term frequency score

Term frequency is how frequently a term occurs in a particular document or in our case fragments. Suppose we have a set of fragments and plan to find the most relevant one to the query, “symptoms of the heart attack”. A simple way to begin, is by removing fragments that do not contain all four terms “symptoms”, “of”, “heart”, and “attack”. Nevertheless, we would still be left with many fragments containing some of them. Then, we should count the number of occurrence of each term in each fragment. This is called term frequency. Regarding [web.stanford.edu/class/linguist289/luhn57.pdf], we can measure it by this formula:

$$TF(t) = \frac{(\text{Occurrences of term in document})}{(\text{Total number of terms in the document})}$$

We can improve this algorithm to involve not only the fragments containing the exact terms but also fragments which have synonym or stem of each term. Furthermore, we should skip the stop words in both query and fragments to make the process simpler. As soon as stop words are removed from the fragments, Stemming, which has been explained in the last section, should be applied to them. Then all the synonyms of each term have to be extracted and involved in the comparisons. There are some databases which possess all the synonyms of words. For example, *WordNet* could be used if dealing with

the English language. *WordNet* is a database of English words linked together by their semantic relationships. The pseudo-code of our suggested algorithm is in the following:

Algorithm 1: Coumpute Term Frequency score

```

1 function CoumputeTermFrequency (query, fragment);
   Input : String query, String fragment
   Output: Decimal F1_Score
   /* Remove stop words */
2 query = REMOVESTOPWORDS(query);
   /* Stem words */
3 stemmedfragment = APPLYSTEMMING(fragment);
4 stemmedfragmentWords = TOKENIZEWORDS(stemmedfragment);
   /* Initialize an ampty list for synonyms and tf for
   computing term frequency */
5 listOfSynonyms = null;
6 tf = 0;
   /* Iterate over each word of query */
7 foreach word in TOKENIZEWORDS(query) do
   /* Get synonym of each word */
8   synonymWords = GETSYNONYMLIST(word);
9   foreach synonym in synonymWords do
10    stemmedOfSynonym = APPLYSTEMMING(synonym);
11    wordNumber = COUNTWORDINTEXT(stemmedOfSynonym, stemmedfragment)
12    tf += wordNumber / GETSIZE(stemmedfragmentWords);
13  end
14 end
15 return tf;

```

Rouge

In recent years there has been considerable number of metrics to automatically evaluate summaries. ROUGE is the most widely used metric for automatic evaluation. Lin [2004] introduced a set of metrics called ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, to evaluate the quality of a machine-generated summary (System) by comparing it to human-generated summaries (Reference). There are several ROUGE measures which we explain the most broadly used ones briefly here:

ROUGE-N: How many overlaps with the words (and/or n-grams) between the human reference summaries appeared in the machine-generated summaries. This works based on recall-based measure and comparison of n-grams. For instance, ROUGE-1 refers to the overlap of unigrams between the system summary and reference summary. ROUGE-2 refers to the overlap of bigrams between the system and reference summaries. The score is computed as:

$$ROUGE - N = \frac{\text{Number of overlapping n-grams}}{\text{Total number of n-grams extracted from reference summary}}$$

ROUGE-L: This measures longest common subsequence (LCS) of texts. Two summaries are more similar when the LCS between them are longer in this metric. The advantage of this metric is that there is no need to define the length of n-gram. Although this metric is more flexible than the previous one, it has a drawback that all n-grams must be consecutive.

The pseudo-code of our proposed ROUGE-N algorithm is discussed in the following:

Algorithm 2: Compute RougeN metric

```
1 function ComputeRougeN (query, fragment);  
   Input : Integer nGRAM, String referenceSummary, String  
           systemSummary  
   Output: Decimal F1_Score  
   /* Remove stop words in both reference and system summary */  
2 referenceSummary = REMOVESTOPWORDS(referenceSummary);  
   systemSummary = REMOVESTOPWORDS(systemSummary);  
   /* Stem words in both reference and system summary */  
3 referenceSummary = APPLYSTEMMING(referenceSummary);  
   systemSummary = APPLYSTEMMING(systemSummary);  
   /* Generate two list of 1 to N gram for both system and  
   reference summary */  
4 referenceSummary_NGrams =  
   GETNGRAM(ngram, referenceSummary);  
   systemSummary_NGrams =  
   GETNGRAM(ngram, systemSummary); /* Initialize an empty  
   list for synonyms and overlap for computing overlaps */  
5 listOfSynonyms=null; overlap = 0 /* Iterate over all generated  
   NGrams of systemSummary */  
6 foreach systemSummary_NGram in systemSummary_NGrams do  
   /* Split each NGram to a list of words */  
7   unigramList = TOKENIZEWORDS(systemSummary_NGram);  
   foreach unigram in unigramList do  
8     listOfSynonyms.add(GetSynonymList(unigram)) /* for each  
       synonym, check if the reference summary contains  
       it. if at least one matches, then this is a hit */  
9     foreach synonym in listOfSynonyms if  
       (referenceSummary_NGrams contains synonym) /* remove  
       it from referenceSummary NGrams */  
10      referenceSummary_NGrams.remove(synonym); /* Increase  
       the overlap */  
11      overlap += 1;  
12   end  
13 end  
   /* Compute Precision and Recall */  
14 recall = overlap / referenceSummary_NGrams.size() precision = (overlap) /  
       referenceSummary_NGrams.size() /* Compute F1 score */  
   /* **beta for F1-score */  
15 beta=1.0; f1 = ((1 + Math.pow(beta, 2)) * recall * precision) /  
       ((Math.pow(beta, 2) * precision) + recall) return f1;
```

3.5.2 Ranking methods

This part of the thesis discusses the methods for ranking anchor fragments. For each method, we will describe the reasons that we believe a method could be useful. Furthermore, how to prepare data and how to rank them will be shown.

For most of the methods, we need to extract some information not only from the target of fragments but also from their homepage. Because in the Extractor, we extract all anchor fragments which have the root of our URL, it is possible to generate a context-based snippet as well. We define context-based snippet as a snippet which is about the whole website, not a specific web page. To do so, we need to extract information from the main pages of each website as well. The initial page of a website is called main page or home page. Therefore, we have an alternative ranked list which, in case the score of ranked fragments are not high enough, the system can rely on. The methods will be explained in more detail in the following.

Frequency of similar anchor fragments

One of our hypothesis is that if an anchor fragment with very similar words has been repeated in different web pages, it could be important. For example, when five anchor fragments of a web page mentions something similar about cosmetic and only one mentions something else, the possibility of the relevancy of fragments which mentioned cosmetic should be higher than the others. Because the links which are in Sidebar, Footer, and Header of a website are usually identical, their anchor fragments will be identical as well. As a result, in this method, such anchor fragments will get a high rank. To avoid such problem, our solution is to count only the frequency of identical fragments which are from different hosts and simply remove the identical one with the same host name.

In the last chapter, the method to count the similar fragments has been explained in detail. Furthermore, we mentioned that we index this number as an attribute of each anchor fragment. In addition, another method was demonstrated to combine the fragments which start with an identical phrase. The combined fragments will get more score per merge and therefore, they are usually on the top of results of this method. In this method, we only need to retrieve a sorted list of the anchor fragments based on their frequency number which has been computed already.

Title

The site title appearing on the top web browsers, usually contains very important information on the web page and could be very useful to rank our fragments. A title tag is paramount for the web authors as it helps search engines understand what a web page is about and search engines show this title to users in their result page. Therefore, anchor fragments containing more title words could possibly be more vital than the ones not possessing any title terms. Because search engines do not display all the titles, usually up to 60 characters, and omit the rest by inserting "...", this title is usually very short.

The title of a Web page can be extracted easily from the title field of the HTML document. A web page title exist in the top of an HTML document inside the `<head>` tag:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Bauhaus-Universität Weimar: Programme structure</title>
</head>
```

As a result of the limitation of search engines for showing the web page title, this title is usually very short. Therefore, for measuring the similarity we should use our Term Frequency algorithm. A very common way to write the title is to have the company or site name in the first part of the title and then depending on the content of each page generate a unique text. It could also be the other way around which would be even fitter for SEO (search engine optimization).

Heading and title of articles

Similar to the titles of web pages, headings or titles of the content of web pages also hold very important information. Headings describe what a paragraph or piece of text is about. They are signposts guiding readers through a web page. Furthermore, they have an effective impact on search engines' rank algorithm. Therefore, we could extract this information from the target anchor web pages and rank our fragments based on the similarity to them. Anchor fragments containing more title words could possibly be more influential.

Headings can be simply extracted from a HTML document by identifying *Heading tags*. Heading tags have a top-down hierarchy from `<h1>` to `<h6>`:

```
<H1>This is heading 1</H1>
<p>This is some text related to heading 1.</p>
```

```
<H2>This is heading 2</H2>
```

```
<p>This is some other text related to heading 2.</p>
```

```
<H3>This is heading 2</H3>
```

```
<p>This is some other text related to heading 2.</p>
```

The most important heading tag is `<H1>` which is usually used for the main title of contents. Other headings, usually, are being used when there is a hierarchical structure in the content. As a result, `<H2>` comes before `<H3>` within a topic. `<H4>`, `<H5>` and `<H6>` rarely exist in web pages. Usually, when the content is very long, authors might use them to add an extra layer to the page structure. Furthermore, they can be used for sidebar or footer headings.

Ranking the fragments based on extracted heading is similar to the title approach but we can improve it by involving the importance of heading tag (number of heading tag shows the importance). Therefore, fragments which are similar to `<h1>` heading should have more ranking score than the others similar to an `<h2>` heading.

Using heading on the Internet is not always for the purpose of important information. Sometimes, web authors, use headings as text format to emphasize something which is not related to the content of web page; for example, “Follow us on Tweeter” which is written in an `<h2>` tag could be found in most of the websites. accordingly, we should remove the irrelevant heading. There are some methods to find out the context of a web page. When we know the context, we can remove the heading not relevant to the context. Since our goal is to generate fragments for a query dependant snippet, we can only rely on the users’ query. As a result, headings which do not have any query terms should be removed.

Meta-data description

Meta-data description, which is very well known in this work, could be also a good resource to rank anchor fragments based on their similarity to it. In the first chapter, we explained that one of the most common ways for generating snippet is using the exact meta-data description. Therefore, we can extract this information from both web page and its homepage. As the length of this information is variable, we should measure the similarity either with TF algorithm or ROUGE algorithm depending on the length of this information.

Similar to the other methods, meta-data description can be extracted easily from a HTML document:

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta tag="description" content="page's description" >
```

Significant sentences (Luhn's approach)

Important sentences of a web page are very reliable resources to rank the fragments based on them. Using Luhn's approach, these sentences can be extracted. As mentioned in the second chapter, the first approach in automatic text summarization and the snippet was Luhn's approach. To apply his algorithm, we need to find the most frequent words first. Stop words should, however, first be removed from the document. As soon as we find the top words, we can find the top sentences. This can be done by counting the number of top words in each sentence. Therefore, the top sentence contains most of the top words. This method works well if there is enough textual information in a web page. As this text will be a long text, the ROUG algorithm should be used.

Applying *Luhn's approach* to a web page is a bit challenging. Web pages are written in HTML and therefore should be parsed to the text. Style sheets, scripts, comments, and meta-data tags are some of the components of a web page to make it user-friendly, interactive, responsive for different size of screens and optimize it for the search engines. If we extract the significant sentences from this data, we might come up with some Javascript or CSS code. The simple way is to parse this document to a readable text using DOM, similar to what we do in extracting the anchor fragments. Furthermore, web pages usually have some information, not related to the topic of that page. Figure 3.12 shows that most of the web pages usually have a header, navigation menu, footer and some advertisements on their sidebar and the main content in the middle. Therefore, we should extract the significant sentences from the content of web pages. Kohlschütter et al. [2010] proposed an approach to extract a cleaned HTML for scraping content out of boilerplate like navigation and advertisements. An implementation of their system which is described in their study is available under an Apache 2.0 license².

²<http://github.com/kohlschutter/boilerpipe>

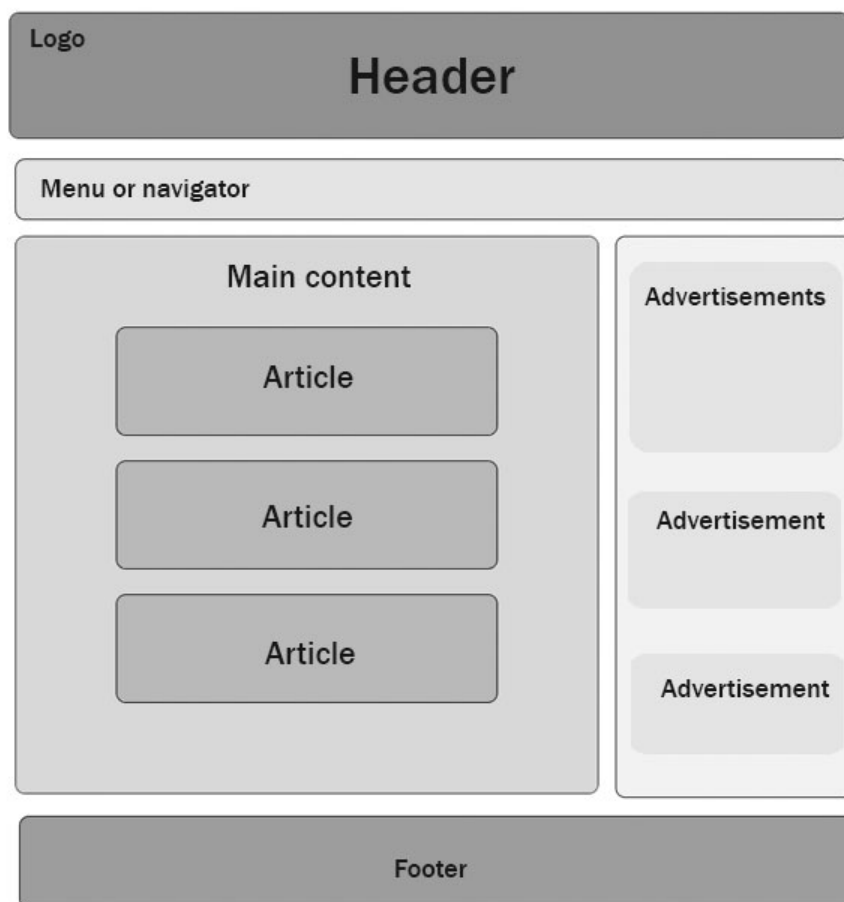


Figure 3.12: Common anatomy of web pages

First paragraph

Similar to the last method (significant sentences), the first paragraph of a web page might hold an overview of a web page. To extract the first paragraph we need to parse the document and extract the main content of it. To prevent any errors such as extracting a heading as the first paragraph, we can count the sentences of the founded paragraph and if it has more than one sentence then it is the first paragraph. This method also works properly, if there is enough textual information in a web page. Since this text paragraph should at least have two sentences, the length of this text would be long; therefore, the ROUG algorithm should be used.

Similarity to query

The query is another available data, based on similarity to which, the anchor fragments can be sorted. Search engines retrieve a page in their result when it is very relevant to the user's query. Therefore, this query should have very relevant terms for us to rank our fragments.

3.6 Summary

In this chapter, we presented our approach towards building a system for extracting and ranking meaningful anchor fragments from a web corpus. At the beginning of the chapter, a formal overview of the architect of the system has been given. Then our suggested strategies to extract fragments from HTML documents, and remove the meaningless fragments was presented. Later in the chapter, all of our suggested approaches to rank the fragments were given. In the following chapter, we proceed to implement the system and analyze the output.

Chapter 4

Evaluation

The experiments in this chapter are performed in order to ascertain aspects of the effectiveness for the proposed ranking methods. Here, we will first introduce the common methods of snippet evaluation and propose our subjective evaluation method. Then, we describe both the implementation of the system to develop the output (anchor fragments), and the steps of conducting the user study. Finally, the result of our evaluation will be shown and analyzed.

4.1 Common approaches

Inasmuch as there is no clear-cut definition for a high-quality snippet, the evaluation and assessment of an effective snippet has been a controversial issue in text retrieval information. Before describing our subjective evaluation to assess the quality of our ranking methods, we will explore the other possible snippets' methods of evaluation.

Since a snippet could be considered as a summary of a webpage, summary evaluations methods could be taken into account to evaluate them. Although automatic text summarization dates back to Luhn's work in the 1950s, evaluation of a summary has still remained a complicated task. The biggest problem is that there are no criteria for the definition of an ideal summary. Broadly, the most common evaluation methods can be categorized into the fully-automated evaluation, semi-automated evaluation, and manual evaluation. There is a number of literature referring to these methods, such as Jones and Galliers [1996] and Sanderson [2000].

According to Saggion and Poibeau [2013] study, the readability of a summary and inclusivity of the important parts of a document are of a major difficulty in the fully-automatic and semi-automatic summary evaluation. Therefore, the simplest and safest way to evaluate a summary is to do so manually, which is done by human evaluators. Human experts take many factors into

account when scoring a summary, namely: *grammar, non-redundancy, integration of most important pieces of information, structure, and coherence* [2].

Although a snippet could be considered as the summary of a web page, our snippet candidates, anchor fragments, are not necessarily the summaries of web pages. The anchor fragments could be written by machines or humans, and might only hold some information about their targets. As a result, we do not follow the summary evaluation approaches.

However, since the goal of our evaluation is to assess the quality of the ranking methods, and this evaluation is subjective, human judgment has been utilized for evaluation.

The theory of using human judgments for evaluating snippets is pretty simple; however, there exist some drawbacks. First, it is very costly due to the scale of the task. Second, the time restriction brings about the need for more evaluators in number to assess the anchor fragments. Nevertheless, thanks to the availability of crowdsourcing platforms like Crowdfunder¹ and Amazon Mechanical Turk², we were able to assess all the available data in a short period of about 48 hours via MTurk. A moderate budget was also provided by the University, which made it possible for us to run the task for a small number of fragments. Another impediment of manual evaluation, making it a complex task, is that the human evaluator may not be able to perceive exactly what we are looking for. In order to tackle this, we made up with a very informative instruction page to make the demand explicit. In the next section, these steps are specified in more detail.

4.2 Experiment setup

In the previous chapter, it was explained, that we used human judgments via a crowdsourcing platform to assess the quality of anchor fragments. In order to provide the data to evaluate, we initially implemented the system by following the suggested approaches for the *Extractor*, *Preprocessor*, and *Ranker* to generate anchor fragments. Since the *Indexer* is helpful for a real scenario such as search engine, we did not implement it in our experiment. Then, we assess the outputs of each ranking method by running an online survey via Amazon Mechanical Turk (MTurk), which is an online platform that allows researchers annotate data with the help of human evaluators time-effectively.

¹www.figure-eight.com

²www.mturk.com

4.2.1 Dataset

Regarding the architecture of the system, which is elaborated in the third chapter, our system needs a list of links and related queries to each link as the inputs and also a corpus of web pages as the web corpus. Furthermore, the *Ranker* needs the content of the target of links together with the content of the home page of links.

For the web corpus we used a full version of ClueWeb12³ corpus. ClueWeb12 contains around 733 million pages crawled from a large general English-language web corpus, collected between February 10, 2012, and May 10, 2012.

Generating some random queries was a pretty straightforward task, but since the relevant links to these queries from the same corpus (ClueWeb12) of the *Extractor* was required, using a search engine which works with this corpus was essential. Fortunately, we have found some queries and their relevant web pages in some of the last Text REtrieval Conference (TREC) tracks. We used the queries and relevance judgments from Web Tracks in TREC-2013⁴, TREC-2014⁵ and Session Track⁶ in TREC-2014. A total of 50 queries were available for each task. Therefore, 150 queries with all of their relevant judgments were chosen, about 19975 relevant documents.

Since only Trec-Id of the relevant web pages is available in TREC judgments' results, the links for these Trec-Ids needs to be detected. Furthermore, the content of this document along with the content of their homepage is also required. Therefore, we first implemented a MapReduce job to go through all of the available documents and extract the relevant document together with their URL from all of the selected Trec-Ids. In addition, we extracted the content of home pages of extracted URL. In total, 29802 documents were extracted. As some of the URLs can have the same hostname, 19975 documents of the URLs and 9827 documents their home pages.

4.2.2 Implementation of approaches

As the size of ClueWeb is enormous, 5.54 TB compressed and 27.3 TB uncompressed, we integrated MapReduce techniques to our system to process the documents efficiently. The Extractor and the Preprocessor procedures were distributed into several machines using Hadoop. Due to the complexity of the Preprocessor, only some parts of the Preprocessor were used in each node, namely: links and length threshold. Therefore, counting and removing similar fragments and combining them have been applied after this step.

³www.lemurproject.org/clueweb12.php

⁴www.trec.nist.gov/data/web2013.html

⁵www.trec.nist.gov/data/web2014.html

⁶www.trec.nist.gov/data/session2014.html

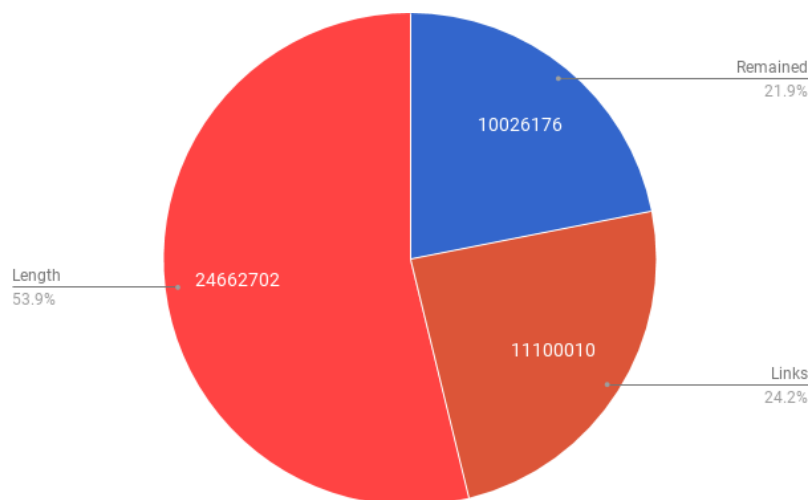


Figure 4.1: This pie chart shows the statistics of preprocessing which has happened during the extraction of anchor fragments

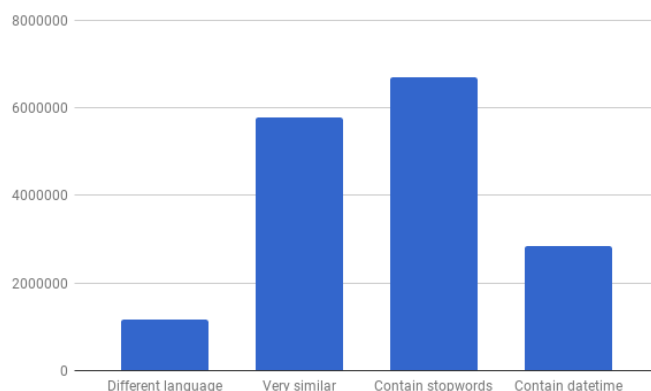


Figure 4.2: The bar chart shows the statistics of preprocessing which has happened separately after extracting then anchor fragments

As mentioned in the index, it was necessary to extract some more information from the target and the homepage of each link. Based on the proposed approaches, meta-data description, title, headings, important sentences, and the first paragraph of these web pages (target of links and their home pages) were extracted. Since the main content of web pages, excluding header, footer, sidebars, and advertisements, was required for some of this information, we had to extract these contents as well. We used Boilerpipe [13] to extract the main content of web pages.

To develop a list of stop phrases we followed our proposed approach in

chapter 3, and consequently, a list of 200 stop phrases were built after having checked them manually.

Finally, we generated the outputs (anchor fragments) for all the available links. Having all the seven ranking algorithms implemented, for each URL we saved 12 ranked lists of fragments separately. As we had five more information for the information extracted from the homepage of the targets, for each URL, we got 12 ranked lists. We used common open source Java and Python libraries for all NLP processes which are elaborated in the Table 4.1.

Table 4.1: Libraries or frameworks which are used in the implementation

Processor	Library
Stemmer	Porter stemmer[20]
Pos Tagger	Maximum entropy tagger[24]
Language Detection	TIKA - Language Detection ⁷
Stop words list	NLTK's list of English stop words
Date time finder	datefinder ⁸ which is a python module

4.2.3 Mturk setup

In this section, we walk through all the steps carried out to conduct the survey on Mechanical Turk. First, the preparation of required data will be described. Then, we explain different parts of the user interface. Afterwards, our strategies to improve the quality of the survey and prevent cheating are elaborated. finally, the general result of the task will be discussed and in the future sections we will analyze the results.

There are some specific definitions in MTurk, which should be made clear. Each rating task, done by multiple workers is called a Human Intelligence Task (HIT). Each HIT, done by a worker is called Assignment. Workers are also called Turkers in MTurk. The reward is the amount of money, defined for the assignment.

Scenario

In the following, the desired scenario of this study will be explained. In order to have the opinion of the workers about the anchor fragments, the workers would be shown a screenshot of a webpage, a query and bunch of fragments to rate.

The workers should see a page with instructions and the task to fulfill when they start working. As a consequence, the instruction should be very short and easy to understand. In case of the complexity of the task, an extra tutorial would be required to make the task more clear. When a worker understands the task and starts it, she should see a screenshot of a web page along with its related query and a bunch of fragments ready to rank. He or she can rank each fragment to one of these labels: Bad, Mediocre, Good, and Awesome. Finally, we are able to collect the rates and analyze them. Table 4.2 describes the rating scheme we developed for evaluating the quality of fragments on MTurk.

Table 4.2: This scheme shows the definition of each rate

Rating	Details
Bad	The fragment has major problems.
Mediocre	The fragment has a minor problem.
Good	The fragment is very good but has some minor problems.
Awesome	The fragment does not have problems and it is awesome.

Data

To set up the task, each HIT needs three main data. First of all, workers need to see the content of target links to have an idea about the expected fragments. Therefore, we decided to capture a screenshot of each webpage. Secondly, workers should see the related query to this link. Third, the most important data is a list of anchor fragments categorized by seven ranking methods.

In order to prepare these data and in regards to our budget, we should choose some of the available anchor fragments which comply with some conditions. However, regarding the preparing of screenshots, there occurred an issue. ClueWeb has only the content of web pages and not the other resources such as images and script files. Therefore, the screenshots captured from this corpus are useless. To solve this problem, the easiest way was to check the available content of the URL on the Internet to see whether it is still the same as it was in 2012 or not. In this case, we simply capture web page entirely with an add-ons called FireShot⁹.

On the other hand, the number of fragments for each HIT should pass a threshold to be fair for the workers and quality of the task. We computed the cost of each HIT by estimating the time that takes a worker to complete a HIT. Therefore, we should have a fixed number of fragments for each HIT.

⁹addons.mozilla.org/en-US/firefox/addon/fireshot

Furthermore, to assess the ranking methods we should have the same amount of rated fragments for each method. The list of ranked fragments of some ranking methods could be zero or a big number. For example, if a page does not have a meta-data description, the output of related method will be empty.

To sum up, the required data was prepared, taking these two conditions and our budget into consideration. Firstly, we run a pilot study, 3 HITs rated by 3 workers, in order to estimate an average time for completing a HIT, and also to discover the user interface's bugs. After dividing the average time in the budgets, we found out that we can run the task for 600 HITs. As we wanted to have at least three different opinions of workers for each anchor fragment, we prepared data for 200 HITs. In order to benefit from various information, we chose four links from 50 queries. We chose links which have at least three fragments for each ranking method and their content is the same as the available content on the Internet.

User Interface

In order to adapt the task (rating fragments) to the MTurk format, we designed two web-based graphical user interfaces using HTML and Javascript. One template page which allows the workers to rate the fragments called HIT page as well as a tutorial page to make the task clear for the workers, which is called tutorial page.

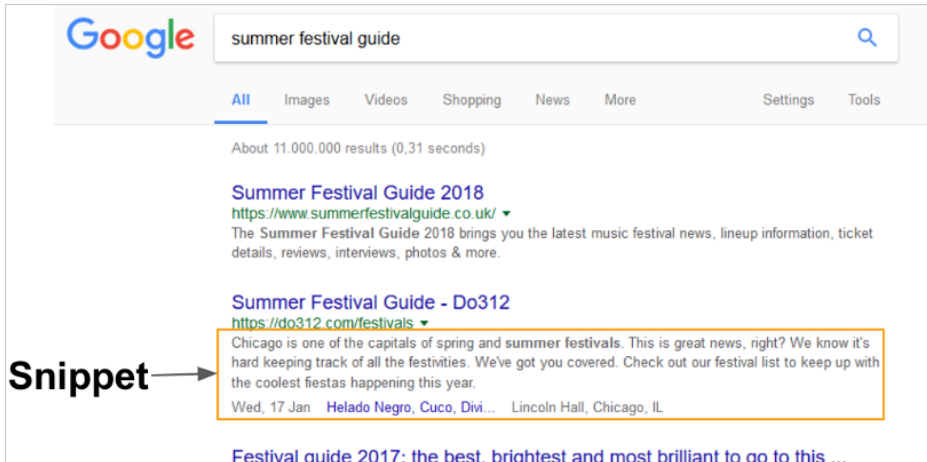
We divided the HIT page into two sections: *Instruction*, and *Task*. Figure 4.3 reveals the instructions and descriptions of the task, and workers are able to open the tutorial page in it. Figure 4.4 exhibits the other part which is the main section. Workers could see a clear screenshot in each HIT and a query related to this web page with a fragment which they should rate. In order to keep the query and web page always on top, only one fragment was shown to the workers each time. After they rate each, the next one was shown to them automatically. To let the workers correct their mistakes, we made a navigation bar below the fragment which allows them to go forward or backward. Furthermore, by validating the rates we forced them to rate all of the fragments and then let them submit the HIT.

Instructions

Instructions

You will see the content of a website and some ‘snippets’ relevant to this website and you should rate them. ‘**Snippet**’ is a short summary of the content of a website that appears in the search results and helps users to judge about the content of the target website.

You read snippet always when you search something in Google:



The screenshot shows a Google search interface. The search bar contains 'summer festival guide'. Below the search bar, there are tabs for 'All', 'Images', 'Videos', 'Shopping', 'News', 'More', 'Settings', and 'Tools'. The search results show 'About 11,000,000 results (0,31 seconds)'. The first result is 'Summer Festival Guide 2018' with the URL 'https://www.summerfestivalguide.co.uk/'. The second result is 'Summer Festival Guide - Do312' with the URL 'https://do312.com/festivals'. A yellow box highlights a snippet from the second result: 'Chicago is one of the capitals of spring and summer festivals. This is great news, right? We know it's hard keeping track of all the festivities. We've got you covered. Check out our festival list to keep up with the coolest fiestas happening this year. Wed, 17 Jan Helado Negro, Cuco, Divi... Lincoln Hall, Chicago, IL'. An arrow points from the word 'Snippet' to this highlighted text.

How to rate:

- Imagine you have searched something and you have the result of the search engine
- You read the snippet and then you will decide to open it or not.
- We will concentrate on one of the result which we call it "Founded webpage"
- First, We show you the content of the founded webpage
- Then we show you bunch of fragments which are somehow related to this webpage
- You should rate each of them based on how good they could be considered as an snippet.
- Before starting, we recomend you to work on this [example](#) to understand the instruction better.
- Your rate is very important for our system, please rate each of them carefully otherwise we can not approve your "Hit".

Figure 4.3: A screenshot from the instruction of the task

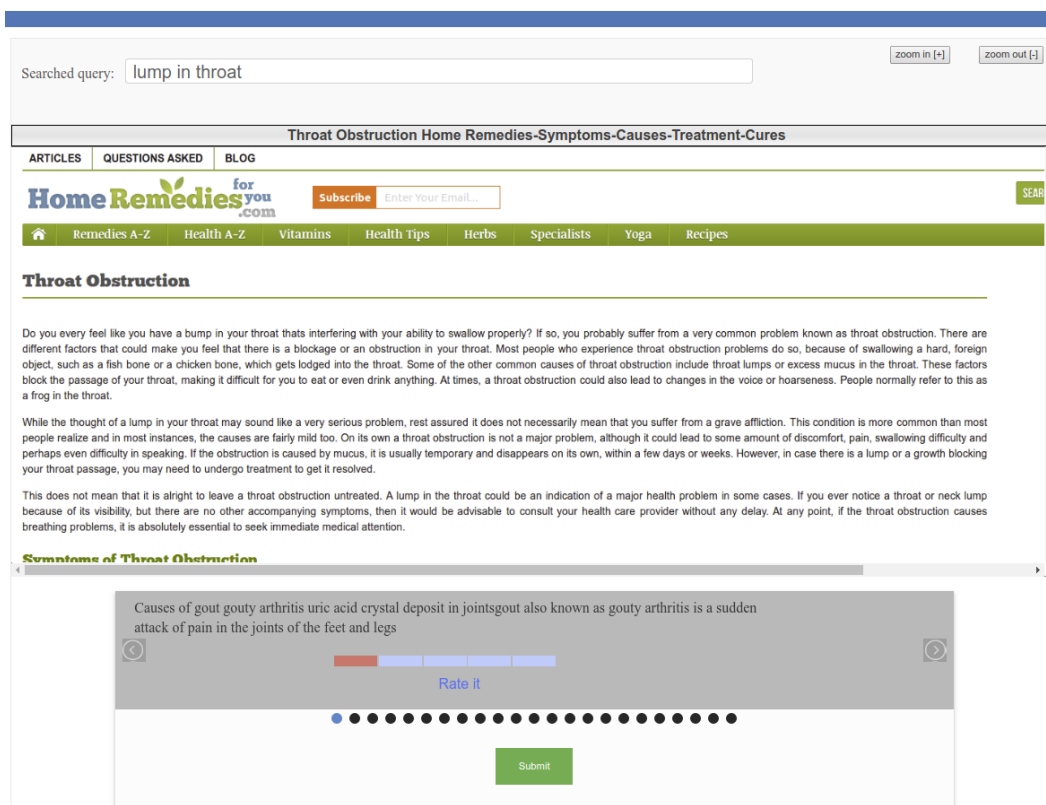


Figure 4.4: A screenshot from the main task, workers can rate each fragment and navigate among them

The tutorial page was created in an interactive way. We generated a step-by-step introduction for a sample HIT. In the first steps, we described all different elements of the page and then in the rest steps helped them to figure out how to rate and the reason that a fragment could be good or bad.

More screenshots and information about the user interface are available in Appendix A.

The quality of annotation

The quality of the data produced by the workers has particularly been important for us since we planned to rely on them to judge the ranking methods. Furthermore, we needed to make sure, that workers understand the task fully, perform it properly, and also prevent cheating in the task. That is precisely why, during the experiment, we used MTurk methods to restrict the cheating and misunderstanding of the task as well as improving the quality. Then we measured the inter-rater agreement of the data to check their extent of reliability.

MTurk offers several qualification features to assure the quality of the results automatically, such as using experienced workers and assignments rejection method. To direct our HITs to the most experienced workers, MTurk offers two procedures; it either recommends to use the master workers who are qualified by MTurk itself or to specify additional qualification filters which should be achieved by workers in order for them to be authorized to work. In addition, we can manually approve or reject the assignments in MTurk. Therefore, using some strategies, we can find out the results of assignments which we are sure they come from cheatings or misunderstandings and rejecting them.

Since Mturk charges an additional fee for Master workers, we define two common and useful filters for the workers. We set the “*Number of HITs Approved*” at a level greater than 100 approved assignments. This means that only workers who have completed 100 assignments with approved status will have access to our HIT. In addition, we set the “*HIT Approval Rate*” at greater than 80%. This filter chooses the workers who have the approval rate from other requesters (the owners of the task) in the MTurk greater than 80%.

Our strategy to prevent cheating and do not have results which came from a worker who did not understand the task is to use *Known Answers*. In this strategy, workers are given some questions, to which the requesters already know the answers, therefore requesters can see how workers are performing. In order to generate *Known Answers* in HITs, We injected three random irrelevant fragments or even meaningless fragments into each HIT. Because they were irrelevant to the web pages, we expected workers to choose rate *Bad* for such fragments. In case, they vote anything than *Bad* we consider that fragment as a spam. Our policy was to reject the HITs which had at least two spams and one error could be ignored due to the difficulty of the task. Finally, we revealed this strategy to the workers who complained about the rejections.

Statistics

According to the summary of the task from Amazon we found out some interesting statistics. The main phase of experiment, consisting of 197 HITs, was completed in less than 6 hours and it cost \$283, including Amazon fee. In total about 300 workers worked on the annotations and spent in average 4 minutes per assignment. 135 annotation sets were rejected which corresponds to 22% of the all assignments.

Reliability and Validity

Before evaluating the results, we did an inter-rater agreement test to see how reliable our rated fragments are. Inter-rater agreement measures the degree of

agreement among workers. There are several approaches to measure inter-rater agreement, each of which is appropriate for a different situation.

Depending on type of the data, there are appropriate measures to compute inter-rater agreement, such as *Percentage Agreement*, *Cohen's Kappa*, *Fleiss's Kappa*, and *Krippendorff*. Some of these measurements get results just in the case that there are only two raters and categories while some of the others work for multi raters and categories. More information is available in Artstein and Poesio [2008] study, giving an overview of all of these measures. Possible values for these statistics range from 0 to 1, with 1 indicating perfect agreement, 0 indicating completely random agreement. Landis and G. Koch [1977] provided Table 4.4 for interpreting k values.

Table 4.3: Equivalent values of the labels

Rate	value
Bad	1
Mediocre	2
Good	3
Awesome	4

Table 4.4: Interpretation of Fleiss kappa(from Landis and G. Koch [1977])

k value	interpretation
<0	poor
0 - 0.2	slight
0.21 - 0.4	fair
0.41 - 0.6	moderate
0.61 - 0.8	substantial
0.81 - 1.0	almost perfect

According to Artstein and Poesio [2008], we used Fleiss' kappa to measure the inter-rater agreement. Fleiss kappa is suitable for situations in which there are multiple raters with different examples. Therefore, as we have multiple raters and categories we chose this method. The first step to measure the data was to change the rates label to numbers as Table 4.3 explicitly pictures. Then we used DKPro Agreement, a novel Java-based software library [19] for

computing multiple inter-rater agreement measures using a shared interface and data model.

After computing the Fleiss' kappa measures for each ranking method, we generated Table 4.5 and computed the level of agreement according to Interpretation of Fleiss' kappa from Landis and G. Koch [1977] as following:

Table 4.5: Inter-rattor agreement results for each ranking method

Direction	Fleiss' Kappa k value	Agreement level
Title	0.551	Moderate
Query fragmetns	0.589	Moderate
Paragraph	0.483	Moderate
Meta-data desc	0.495	Moderate
Significant sentences	0.503	Moderate
Headings	0.569	Moderate
Frequency	0.600	Moderate
Total	0,542	Moderate

As seen in Table 4.5, the level of agreement for all seven methods is between 0.48 to 0.60 which is equivalent to "Moderate" referring to interpretation table from Landis and G. Koch [1977]. Since "Moderate" agreement is good enough in inter-rater agreements, we proceed the evaluation.

4.3 Evaluation Results and Discussion

In this section, using the data we gathered from the user study, we investigate the effectiveness of ranking methods, which has been the primary goal of this study. First, we compared the average evaluation scores from both top three and top one fragments of all the seven ranking methods. Afterward, we applied significant statistics into our results to see whether the methods show any significant differences in comparison with one another.

In order to analyze the effectiveness of each ranking method, we computed the numbers of votes of all the anchor fragments per ranking method. The

total number of HITs was 200, each of which has 21 fragments rated by three workers. In total, we had 12600 rated fragments, as each ranking method has 1800 fragments. Since three of the Hits were from the pilot study, they were not counted in statistics. We categorized anchor fragments by their ranking method and counted their rates to generate Table 4.6 and Figure 4.5.

Table 4.6: Number of votes for all rated fragments per ranking method categorized by rates

<i>Method</i>	<i>Rates</i>			
	Bad	Mediocre	Good	Awesome
Frequency	30% (519)	30% (516)	26% (457)	14% (241)
Heading	35% (608)	33% (579)	21% (368)	10% (178)
Mata-tag description	50% (892)	24% (430)	19% (329)	7% (122)
Significant sentences	52% (921)	29% (509)	14% (254)	5% (89)
Query	30% (532)	29% (511)	31% (551)	10% (179)
Paragraph	53% (940)	23% (413)	18% (314)	6% (106)
Title	48% (852)	24% (421)	20% (346)	9% (154)

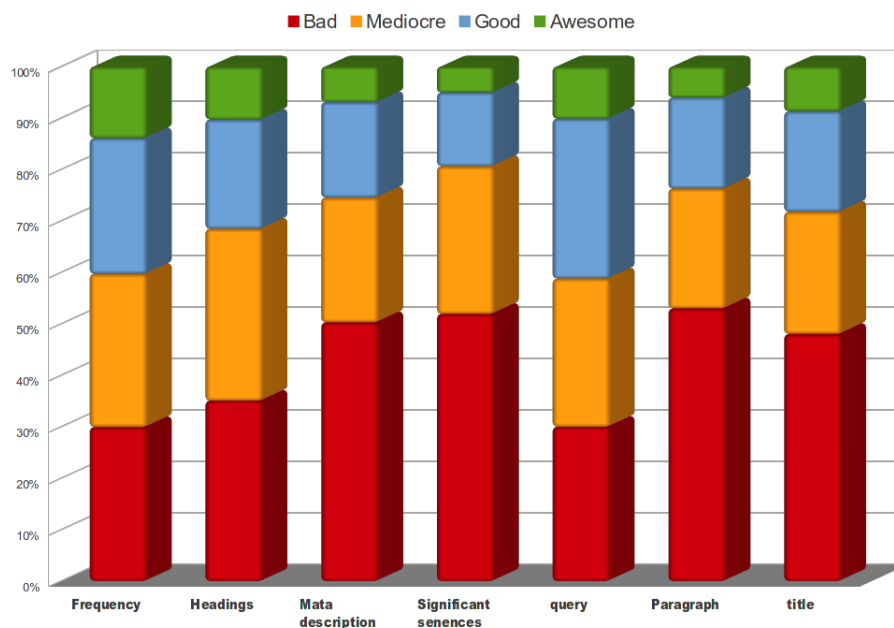


Figure 4.5: Percentage of votes for all rated fragments per ranking method, each color represent a rate as they are described in the legend of the chart

As obvious in Table 4.6 and Figure 4.5, it seems that workers were not

satisfied with the result. A big portion of votes belongs to label Bad (42% of all the votes) which means fragments with this label were either meaningless or not relevant to the web page. This result might be explained by the fact that the task was not fully understood by the workers as well. They rated better for the fragments which are ranked based on the frequency of them, 40% of the fragments in this method are Good or Awesome. Frequency method has also the least votes for Bad. In the second place, workers liked fragments which are ranked by Heading and Query which include more than 30% Good or Awesome. Although about 30% of fragments ranked by the Title method are also good, a big portion of them is Bad. On the other hand, workers voted Bad for more than half of the fragments ranked by Paragraph, Meta-data description, and Significant sentences while less than 30% of them are ranked as Good or Awesome.

Since the first fragment retrieved from the Ranker is more important than the second and third one, we count the rates only for the first fragment of each method and generated Table 4.7 and Figure 4.6. As evident in the Figures, results are somehow similar to the last results. Frequency method enjoys the best votes (Good, Awesome) and least Bad votes. More than half of the fragments ranked by Paragraph, Meta-data description, and Significant sentences again have the worst votes. We can see the Bad votes for Title method decreased to 23% and they added to Mediocre label which it has increased to 37%.

Table 4.7: Number of votes for the first fragments of each ranking method categorized by rates

<i>Method</i>	<i>Rates</i>			
	Bad	Mediocre	Good	Awesome
Frequency	8% (49)	38% (222)	39% (232)	14% (88)
Heading	20% (117)	34% (198)	33% (196)	10% (80)
Mata-tag description	49% (291)	24% (142)	20% (118)	7% (40)
Significant sentences	55% (324)	22% (130)	17% (98)	5% (39)
Query	13% (76)	37% (221)	35% (209)	10% (35)
Paragraph	53% (314)	23% (134)	18% (105)	6% (38)
Title	23% (135)	37% (217)	27% (161)	9% (78)

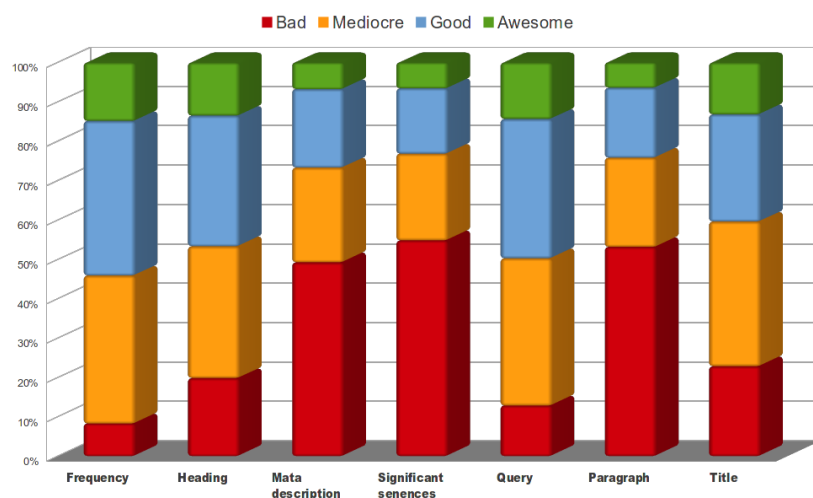


Figure 4.6: Percentage of votes for first rated fragment of each ranking method, each color represent a rate as they are described in the legend of the chart

In order to find out if there exists a significant difference between the ranking methods, the t-test was applied on our data, since t-test determines if two sets of data are significantly different from each other. We compared each ranking method with all the others. Specifically, we mapped the rating from bad to awesome onto the range from 1 to 4 and gained the average votes of the fragments. Table 4.8, which is a confusion matrix, depicts the computed p-value of each pair. As it is clear, there are only five significant values in this result. Frequency method has more significant values than the others. According to Table 4.8 Frequency method is significantly different than Meta-data description method and Paragraph method. On the other hand, there is not a significant difference between Frequency, Heading, and Query methods. Moreover, the related table demonstrates that the results from Paragraph method to Query method are significantly different and the same applies for Significant sentences method to Title Method.

Table 4.8: This confusion matrix depicts the computed p-value between ranking methods

<i>Methods</i>	Headings	Significant.S	Meta.D	Paragraph	Query	Title
Frequency	0.643	0.153	0.003	0.027	0.237	0.136
Headings		0.133	0.0128	0.443	0.459	0.210
Significant sentences			0.292	0.484	0.136	0.028
Meta description				0.089	0.242	0.162
Paragraph					0.017	0.770
Query						0.746

Bold values are the significant results. The result is significant at $p < .05$.

4.3.1 Discussion

After having investigated more in the data, some interesting results were detected. Firstly, we realized workers liked the combined fragments more than the others. Secondly, we understood that the algorithm behind the ranking methods changes the effectiveness of the method a lot.

There is a noticeable difference in the number of good rates for fragments which had an identical part and were combined into one fragment. As it is clear in Figure 4.7, workers rated these kinds of fragments mostly Good (44% Good, 19% Awesome). According to this result, we can prove that the strategy to combine the very similar fragments will improve the quality of fragments.

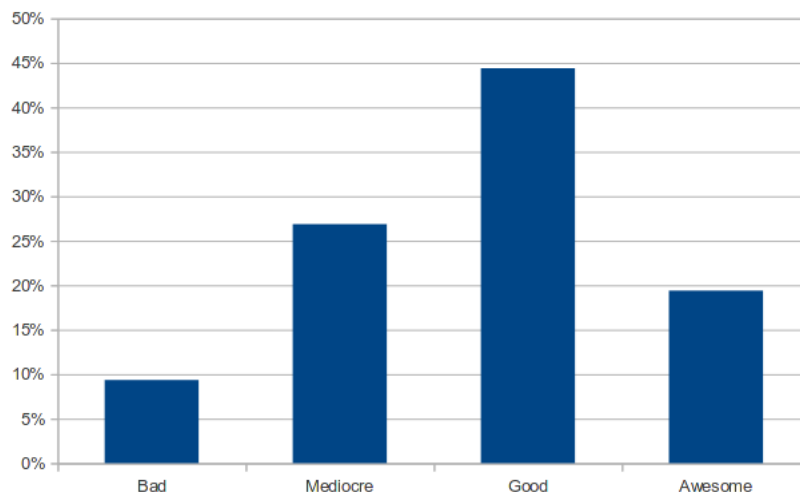


Figure 4.7: Percentage of each rate for fragments which were combined

As we explained the two algorithms behind each ranking method in the last chapter, it might have been noticed, ranking methods based on Term Frequency are significantly different than methods based on ROUGE algorithm. Paragraph, Meta-data description, and Significant sentences methods rank the fragments based on ROUGE algorithms while Title, Headings, and Query rank fragments based on Term Frequency algorithm. To prove this, the p-value of the votes categorized by these two algorithms was computed. This time we applied Chi-Square statistic, which is commonly used for testing relationships between categorical variables [11], to compute the p-value. Table 4.9 describes the results of this computation and shows that p-value is less than $<.05$ which shows the significant difference between them.

Table 4.9: This contingency table provides the Chi-Square statistics between two ranking algorithms

<i>Method</i>	Bad	Mediocre	Good	Awesome	Total
ROUGE	2753 (2381) [57]	1352 (1436) [5.02]	897 (1085) [32.60]	317(415) [23]	5319
Term Frequency	1992 (2363.55) [58]	1511 (1426) [5.05]	1265 (1076) [32]	511(412) [23.55]	5279
Total	4745	2863	2162	828	10598

The chi-square statistic is 238.8238. The p-value is < 0.00001 . The result is significant at $p < .05$.

4.4 Summary

In this chapter, we first presented the common approaches for evaluating the snippets in the field of information retrieval. We explained that human judges should be used and the common methods for that were also described. Furthermore, we describe all the steps that we have done in order to do our evaluation, using human judges. We showed how we implemented the system and generated some anchor fragments in order to use them during the user study. Subsequently, different steps to design and run the user study in a crowdsourcing platform has been described. Finally, the results of the user study were illustrated and discussed. In this section, we found out fragments which ranked based on their frequency had a significant difference in comparison with the other methods. Some other reasons that let the workers like the fragments have been discovered and discussed as well.

Chapter 5

Conclusion and Future work

In this work, we introduced a novel idea to generate some fragments which might be good candidates for a snippet of a webpage. We proposed a pipeline which can mine the surrounded text of an anchor text and rank them with seven different methods. In order to assess the effectiveness of each ranking method, we conducted a user study and evaluate the results out of them. In this chapter, we highlight our main contributions and findings. We also present the possible areas of improvement on the current work and applications.

5.1 Contributions

We started our research by asking the following research questions; (1) What are the issues in common snippet generation approaches and why do we need alternative approaches? (2) How does this alternative approach work? (3) How effective is this approach?

To answer these questions, in chapter 1 and chapter 2, we reviewed common approaches to generate snippet and described the drawbacks. Then, we described our hypothesis and some similar studies to this hypothesis along with their issues. Our hypothesis, proposed in chapter 3, was a pipeline with three main parts, namely: *Extractor*, *Preprocessor*, *Ranker*. The *Extractor* proposed for mining surrounded text of an interesting anchor text from HTML files. Then, the *Preprocessor* was introduced to show the possible ways of filtering out the meaningless fragments and improving the quality of meaningful fragments. Finally, the *Ranker* was proposed to show some different methods of ranking the extracted fragments.

To answer our last research questions, in chapter 4, we showed all the steps for implementing the proposed pipeline in order to generate some sample anchor fragments to assess the effectiveness of each ranking method. As soon as we generated the anchor fragments, we conducted an online survey via a

crowdsourcing platform. After clarifying the task for participants, we asked them to rate our generated fragments. Finally, we analyzed the rated fragments to see how participants like them and consequently, we evaluated this data.

Future Work

Generating snippet based on anchor fragments could have more scenarios and it is beyond the scope of this work. Therefore, our findings open the door for further and more focused research on the topic of web page summarization.

We believe this contribution could be used by the search engines in order to improve the quality of the snippets. Therefore, future work should concentrate on enhancing the quality of ranking methods to retrieve only the best candidate for anchor fragments. More broadly, further research is also needed to determine how a search engine can integrate an anchor fragment into its snippet pipeline. Figure A.2 illustrates one of the possible ways of improving the quality of snippets by using anchor fragments. As evident, a search engine in both methods, query dependent and independent, should follow its methods to generate a snippet, however, they compare the generated snippet with a threshold to see how good it is; in case it is not good, they use anchor fragments.

According to the workers' votes, we found out the ROUGE algorithm behind some of the ranking methods decreased the quality of ranking and may not be suitable for our pipeline. As a result, it needs to be investigated more specifically in this field to find new algorithms or improve the existing one.

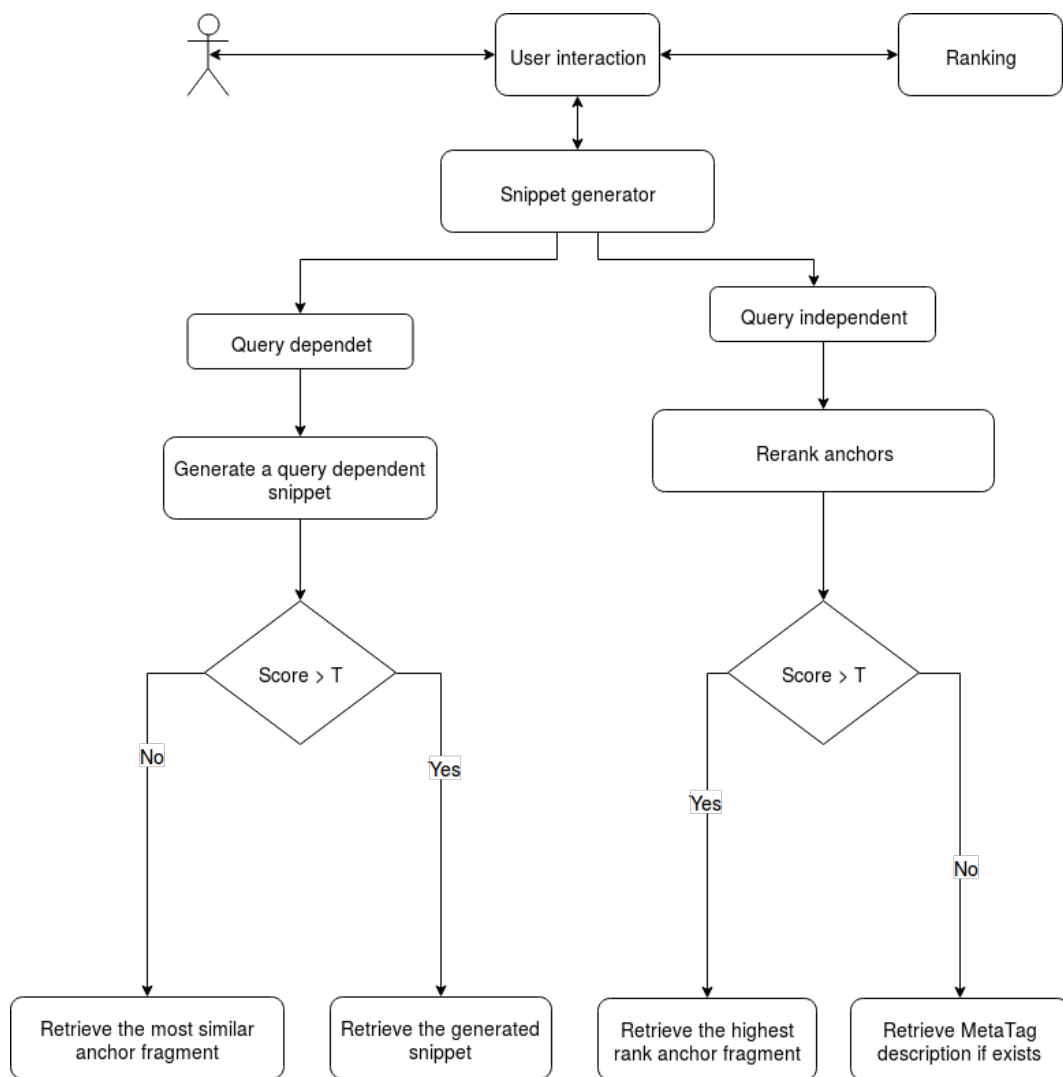


Figure 5.1: This diagram shows the integration of anchor fragments into a search engine

Appendix A

HIT Interface

This appendix describes the instruction of the HIT in the experiments. In addition there are some screen shots from tutorial page.

INSTRUCTIONS

You will see the content of a website and some ‘snippets’ relevant to this website and you should rate them.

‘Snippet’ is a short summary of the content of a website that appears in the search results and helps users to judge about the content of the target website.

How to rate:

- Imagine you have searched something and you have the result of the search engine
- You read the snippet and then you will decide to open it or not.
- We will concentrate on one of the result which we call it "Founded webpage"
- First, We show you the content of the founded webpage.
- Then we show you bunch of fragments which are somehow related to this webpage
- You should rate each of them based on how good they could be considered as an snippet.
- Before starting, we recomend you to work on this example to underestand the instruction better.
- Your rate is very important for our system, please rate each of them carefully otherwise we can not approve your "Hit".

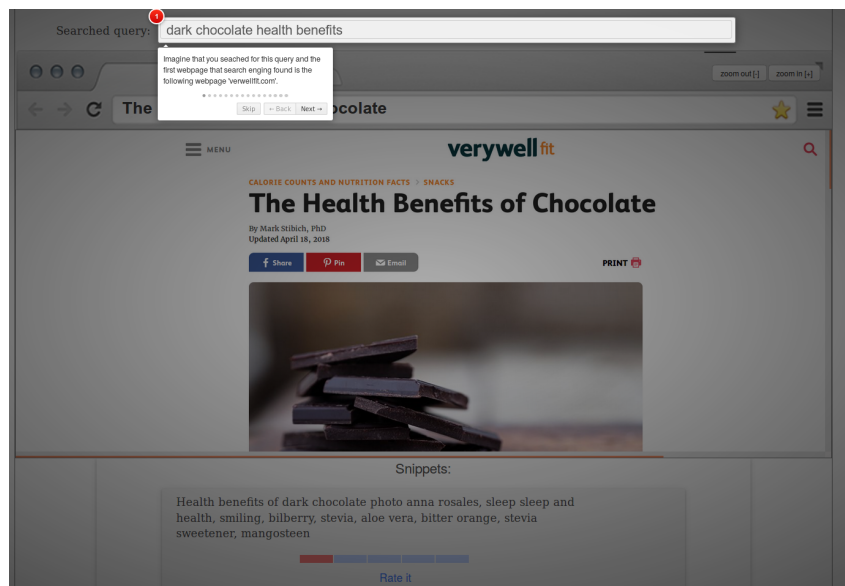


Figure A.1: Example of bad snippet

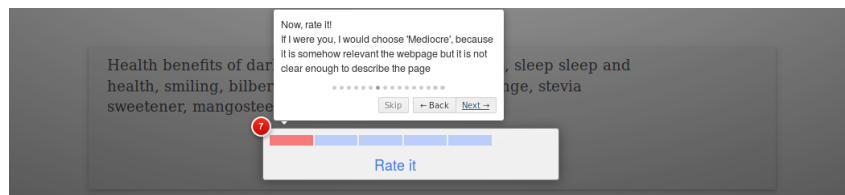


Figure A.2: Example of bad snippet

Bibliography

- [1] Abrae Os, J. and Lopes, G. (2002). Statistical methods for retrieving most significant paragraphs in newspaper articles. 2.1.1, 2.1.3
- [2] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). Text summarization techniques: A brief survey. cite arxiv:1707.02268Comment: Some of references format have updated. 2.1, 4.1
- [3] Amitay, E. (2000). Trends, fashions, patterns, norms, conventions . . . and hyper-text too. *Journal of the American Society for Information Science and Technology*, 52(1):36–43. 2.1.3
- [4] Amitay, E. and Paris, C. (2000). Automatically summarising web sites: Is there a way around it? In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 173–179, New York, NY, USA. ACM. 2.1.3, 2.1
- [5] Artstein, R. and Poesio, M. (2008). Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4):555–596. 4.2.3, 4.2.3
- [6] Buyukkokten, O., Kaljuvee, O., Garcia-Molina, H., Paepcke, A., and Winograd, T. (2002). Efficient web browsing on handheld devices using page and form summarization. 20:82–115. 2.1.3
- [7] Dalal, V. and Malik, L. (2013). A survey of extractive and abstractive text summarization techniques. In *2013 6th International Conference on Emerging Trends in Engineering and Technology*, pages 109–110. 2.1.2
- [8] Delort, J., Bouchon-Meunier, B., and Rifqi, M. (2003). Enhanced web document summarization using hyperlinks. 14. 2.1.3
- [9] Edmundson, H. P. (1969). New methods in automatic extracting. *J. ACM*, 16:264–285. 2.1.1
- [10] Gibson, D., Punera, K., and Tomkins, A. (2005). The volume and evolution of web page templates. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web*, WWW '05, pages 830–839, New York, NY, USA. ACM. 3.3

- [11] Greenland, S. (2016). Statistical tests, p values, confidence intervals, and power: a guide to misinterpretations. *European Journal of Epidemiology*, 31(4):337–350. 4.3.1
- [12] Jones, K. S. and Galliers, J. R. (1996). *Evaluating Natural Language Processing Systems: An Analysis and Review*. Springer-Verlag, Berlin, Heidelberg. 4.1
- [13] Kohlschütter, C., Fankhauser, P., and Nejdl, W. (2010). Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM '10*, pages 441–450, New York, NY, USA. ACM. 3.5.2, 4.2.2
- [14] Landis, J. and G. Koch, G. (1977). The measurement of observer agreement for categorical data. 33:159–74. 4.2.3, 4.4, 4.2.3, 4.2.3
- [15] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. 14
- [16] Lin, C.-Y. and Hovy, E. (2002). From single to multi-document summarization. 2.1.2
- [17] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165. 2.1.1, 2.1.3
- [18] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA. 1, 2.1.3
- [19] Meyer, C. M., Mieskes, M., Stab, C., and Gurevych, I. (2014). DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement. In Tounsi, L. and Rak, R., editors, *Proceedings of the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 105–109. 4.2.3
- [20] Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137. 4.1
- [21] Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '94*, pages 232–241, New York, NY, USA. Springer-Verlag New York, Inc. 2.1.3
- [22] Saggion, H. and Poibeau, T. (2013). Automatic text summarization: Past, present and future. 2.1.2, 4.1
- [23] Sanderson, M. (2000). Advances in automatic text summarization inderjeet mani and mark t. maybury (editors) (mitre corporation). 26:280–281. 4.1

- [24] Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics. 4.1
- [25] Wang, C., Jing, F., Zhang, L., and Zhang, H.-J. (2007). Learning query-biased web page summarization. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 555–562, New York, NY, USA. ACM. 2.1.3