Oxford Brookes University
School of Technology

Scoring pertinence of web page titles using Natural Language Processing

Master's degree Dissertation

Author:
Vivien Barousse
10091900@brookes.ac.uk

Supervisor:
Prof. David Duce
daduce@brookes.ac.uk

2nd September 2011

Abstract

Titles are the most important part of web pages. They are the first thing the reader see, they give the reader an idea of the content they are about to read, and are heavily used by search engines to respond to search queries. However, a lot of web pages are incorrectly entitled, and incorrect titles are misleading for readers and search engines.

This project aims at determining what constitutes a good title for a Web page, and developing a software able to analyse the title of a given web page using Natural Language Processing techniques in order to rate its pertinence.

Through an analysis of the most widely used algorithms and techniques in the area of title analysis and Information Retrieval, the best approaches are selected for each web page depending on corpora and languages. As a practical application, a software is then developed to apply these results and analyse titles of unknown web pages.

The experiments consider both statistical and semantic Information Retrieval techniques to have the most coherent Information Retrieval systems possible, and determines which combination of algorithms works the best in the context of analysing existing web pages titles.

Experimental results showed that a statistical analysis of text can extract its main ideas with an up to 30% accuracy, and that a semantical analysis can help improve the coherence of these results. A prototype is then developed to apply the findings of this experimentation in a real world environment, on diverse web pages and different corpora.

Evaluation of the experimental results and of the prototype suggests that despite the complex nature of Natural Language Processing, acceptable and usable results can be obtained.

Acknowledgments

I would like to thank a number of person which made this Master's project possible.

First of all, my supervisor, Professor David Duce, for the advices, guidance, and academic support all along.

Mathieu Faure (Open-S), for giving me the opportunity and the idea of working on Natural Language Processing as part of my dissertation.

Jérome Kowalczyk (Open-S), for the insightful conversations about the project and ideas about Natural Language Processing and web pages titling.

Martin Filliau, for the lenghty conversations about Information Retrieval, about Software Engineering, and also about nothing and everything.

And finally, thanks to all the people I forgot but helped me, somehow, during the 8 months of this project.

Contents

1 Introduction

Titles are one of the most important piece of information in a web page. It is used by search engines to show search results to users, it is the first thing a reader will see, and it gives people a general idea of the contents of the page.

However, studies have shown that too many web pages are somehow incorrectly entitled [1]. Incorrect titles are a problem for both website editors and readers. Website editors can loose audience due to poorly entitled pages and readers can have trouble finding quickly information they are looking for.

This project aims at analysing what makes a good title for a web page, analyse techniques to detect the relevant information that need to be located in the page title and to develop a tool that is able to rate the title of a given web page.

1.1 Relation to field of studies

This project focuses on Natural Language Processing, which is a field of Computer Science. More precisely, this project is related to the area of Information Retrieval.

Also, this project includes concepts related to other area in Computer Science, including Machine Learning and Information Extraction.

1.2 Aims and objectives

The aim of the project is to develop a software able to rate the pertinence of a web page title, based on Natural Language Processing techniques.

[1] Xue et al. found in 2007 that more than 30% of web pages on the Internet are "somewhat bogus" [XHX+07]

The objectives to achieve this are:

Determining what is a pertinent title A relevant title can take multiple forms, depending on the web page content, type, corpora and language. One of the first objectives is to define what makes a good title for a given web page.

Analysing algorithms that can be used to rate title relevancy A wide range of algorithms exist in the context of automatic titling and Information Retrieval. Some of these algorithms can be adapted to fit in the context of this project, some can not. An analysis of the existing approaches and techniques is required to determine which techniques will be useful to this project.

Develop a tool to rate the pertinence of a web page title This is a practical application of the previous theoretical work. The final objective of the project is to develop a robust tool which is able to rate a web page depending on its type, corpora and language.

1.3 Practical application

In a study conducted in 2007 by Xue et al., it has been found that 33.7% of the web pages on Internet are incorrectly entitled and the title of the web page somehow doesn't relate to its content. As said previously, incorrectly entitled web pages are an important problem for both website editors and readers. Thus, giving a relevant title to a web page is an important task for editors.

An automatic analysis of a web page title in order to detect incorrectly entitled pages can be used by website editors to improve the quality of their websites.

2 Literature review

2.1 Good title composition

A lot of research has been done to try to determine what makes a good title for a web page.

One of the most active area in this domain is Search Engine Optimisation, which tries to determine what techniques are relevant to improve the ranking of web pages in search engine results pages. Search Engine Optimisation suggests that the title is a really important part of a web page and plays a major role in the way the web page is indexed. An important number of good practices were deduced from these observations and are widely used over the Web.

More generally, a lot of papers and books also focuses on how to write good titles for general documents, not only web pages.

2.1.1 Good titles for general documents

Web pages are a specific kind of documents, therefore, most rules that apply to the redaction of good titles for general documents also apply to web pages.

In a book written in 1979 (Sixth edition published in 2006) by Day and Gastel, [DG06] a definition for a good title for a scientific paper is given. They answer the exact question of "What is a good title?" by the following: "We define it as the fewest possible words that adequately describe the contents of the paper".

2.1.2 Search Engine Optimisation

A study conducted in 1998 by Pringle et. al. [PAD98] showed that it is important to "include important keywords in the title, [...] but do not use excessive repetition". Their study showed that including important keywords relevant to the content of the page improved ranking within search engines, but that repetition was often considered as a trying to cheat the system and could result in the page being black listed.

Another important tactic described by Pringle et. al. [PAD98] is to give the page an "informative title".

In another paper written in 2010 by Klein, Shipman and Nelson [KSN10], the efficiency of a web page title is determined by submitting multiple parts of the title as a query in a search engine and measuring the position of the original page in the results list. This approach makes the assumption that a good title for a web page is a title that makes it efficient in a search engine, and makes the web page appear early in search results.

2.2 Information retrieval

The task of rating a title requires to retrieve information from the source document. Extraction of such information has already been treated in three distinct areas:

Title generation focuses on the extraction of important terms that should be present in a document title

Automatic summarisation can be considered, as a title can be seen as a very short summary of a document

Title word selection has been considered more recently, focusing on choosing single words that should be present in a title

Some other areas are worth investigating due to their usage in relative areas such as document searching and mining. These areas include anchor text analysis.

2.2.1 Title generation

The area of title generation is interesting for this project because it relates to the extraction of interesting information to generate relevant titles for documents.

Such approaches and algorithms can be adapted to rate existing titles instead of generating a new one, after the extraction of interesting terms has been performed.

Titles generation has already been covered in some papers and multiple techniques exist.

In a 2001 paper by Jin and Hauptmann [JH01a], different algorithms were experimented and compared. Tested algorithms are all based on a statistical approach to information extraction, and most of them are Machine Learning based. In this survey, TF-IDF[1] was highlighted as the most accurate approach. This is a statistical (but not machine learning based) algorithm.

TF-IDF is the most common statistical model used in the context of important terms extraction. It works by counting the frequency of a term in a document, and comparing this frequency to the number of document this term appears in. This techniques highlight frequent words in a document and words that are specific to a small amount of documents.

The same authors proposed in 2002 a new probabilistic model [JH02]. After a comparison to older statistical models, the paper shows that this new approach is more accurate than TF-IDF. However, this algorithm was designed with an important focus on title readability rather than important words extraction.

Kennedy and Hauptmann (2000) [KH00] introduced a new probabilistic title generation method inspired by statistical translation algorithms. In this approach, the title of a document is considered as a destination language and the text of the document as a source language. By applying the statistical translation algorithms to the document text, a title can be generated for the document. For the task of title generation, Kennedy and Hauptmann precise that this technique "outperformed another approach based on Bayesian probability estimates".

[1] Term Frequency - Inverse Document Frequency

2.2.2 Automatic summarisation

Kennedy and Hauptmann (2000) [KH00] state that "Extractive summarisation is the most common approach to generate titles".

That is, automatic summarisation can be considered as a suitable approach to extract information required for the analysis of existing titles.

Techniques used for extractive summarisation are mostly statistical approaches.

Kennedy and Hauptmann [KH00] use a variant of TF-IDF (which was also used in the context of title generation) to detect important words and interesting sentences.

Witbrock and Mittal [WM99] introduce a probabilistic model based on a Naive Bayesian approach. This algorithm is a machine learning algorithm, and tries to determine the probability of a term being present in the summary of a text by determining the probabilistic correlation between terms present in a document and terms present in the title of the same document.

Another interesting approach is used by Aone, Okurowski and Gorlinsky in 1998 [AOG98], and combines multiple existing techniques to improve information extraction results. This solution uses a statistical model, improved by a machine learning approach, in order to identify important words in a text. Finally, synonyms recognition is used to improve the coherence of extracted words.

2.2.3 Title word selection problem

In 2001, Jin and Hauptmann [JH01b] focused on the problem of extracting good title words for documents using a learning approach.

According to their experimentation, this approach out-performed other learning approaches such as TF-IDF and Naïve Bayes classifiers.

The Naïve Bayes approach out-performed in this approach is a reproduction of the work produced by Witbrock and Mittal in 1999 [WM99].

2.2.4 Anchor text analysis

Search engines are specialized in information retrieval from web pages. Information retrieved is then used to provide results to user queries. Even is the usage made of the extracted data is different from the task of analysing pages titles, multiple techniques have been applied in search engines and can be reused in the context of this project.

Such a feature is the anchor text analysis. During the development of Google, Sergey Brin and Lawrence Page [BP98] described a feature consisting in associating the text of an hyperlink to the page the link points to. This concept was already described in 1994 by McBryan [McB94], and implemented in its WWWW tool.

Their first motivation behind this feature was that links "often provide more accurate descriptions of web pages than the pages themselves" [BP98].

Such a technique could be reused in the context of this project to obtain terms about web pages using the text of the links pointing to that page.

A similarity between anchor text and web pages titles was assumed by Eiron and McCurley in a paper written in 2003 [EM03] attempting to prove the pertinence and relevancy of anchor text pointing to a web page.

2.3 Comparing titles

Considering the previous section on Information Retrieval, it can be interesting to compare an automatically generated title with the existing title for a web page.

Techniques to compare titles have been created to test the accuracy of title generation algorithms. Those techniques can be reused in the context of this project.

Rijesbergen introduced in 1979 [Rij79] a technique to evaluate the efficiency of information extraction techniques called the F_1 measurement.

This measure has been reused since by multiple authors, including Jin and Hauptmann in 2002 [JH02]; and Witbrock and Mittal in 1999 [WM99].

The idea behind this technique is to compare the number of words in common between automatically generated and pre-existing titles, related to the total number of words in

the titles.

While this metric is very useful to rate the efficiency of a generated title, it doesn't take in account the title readability or any general syntax. Also, this technique focuses only on the exact syntax of the word and doesn't give any semantic meaning to each word. Synonyms aren't taken into account, and this can reduce the score of a pertinent title.

Another metric exists to compare titles, and also comes from Information Retrieval techniques. This is the Average Precision metric. It has been widely used in the context of search-oriented Information Retrieval [TS06].

Even if this metric hasn't been used in the context of title evaluation, it can be adapted to suit this use case.

This metric takes into account the rank of the retrieved documents, which can be an important advantage compared to F_1.

2.4 Semantic analysis

When comparing a title to a list of relevant words for a document, words can be different from the initial ones while carrying the same meaning. This is the case with *synonyms* and *inflected words*.

Some algorithms discussed earlier introduce the notion of comparing words with some kind of synonyms differentiation to improve coherence in the algorithms results.

2.4.1 Synonyms

Synonyms differentiation can be a complex task. The most important problem to solve is the origin of synonyms. Since the differentiation can only be based on a list of synonyms for all the words of the language considered, the quality of such a list is a major issue for synonyms detection.

In a study conducted by Navarro, et al. in 2009 [NSB+09], Wiktionary [2] was used to

[2] A project of the Wikimedia foundation, `http://www.wiktionary.org/`

obtain a collaborative list of synonyms for a given word. Wiktionary is an open dictionary which contains information about words, including synonyms for a given word. Data are extracted from the website to generate a synonymy network. This paper also introduces methods that can be used to improve the obtained synonymy network; the first one is based on semantic proximity between words, the second uses translations of words to detect synonyms in other languages.

Wiktionary was also used to retrieve synonyms for a given word in another paper by Weale, Brew and Fosler-Lussier in 2009 [WBFL09].

2.4.2 Inflected words

Inflected words are derived from the same root, and thus, carry the same idea. However, they have different spelling. To increase coherence, it can be interesting to look at the root of a word, instead of its final form.

The process of transforming a word into its fundamental root is called stemmisation.

Stemmisation has already been largely covered by scientific papers. In 2001, Pirkola, Hedlund, Keskustalo and Jrvelin stated that "a commonly used method to deal with inflected search keys as well as derivationally related keys is to remove affixes from word forms" [PHKJ01]. This kind of stemmisation is called morphological stemmisation.

A well known morphological stemmisation algorithm for the English language is the Porter stemming algorithm. It was first described by Martin F. Porter in 1980 [Por80]. This algorithm shows good results, but is unfortunately limited to the English language.

However, according to a book written in 2000 by Strzalkowski, Lin, Wang and Perez-Carballo, this stemmisation can decrease the precision of the information retrieval [SLWPC00]. A workaround to this loss of precision is to use a dictionary and "replace a traditional morphological stemmer with a conservative dictionary-based suffix stemmer" [SLWPC00].

2.4.3 General words databases

Semantic analysis of a text may require an exhaustive database of words to detect synonyms, reduces word inflections and detect named entities.

While such databases can be constructed using websites such as Wiktionary (as discussed earlier), databases of words and concepts already exist and can be reused in the context of this project.

WordNet [3] is a database manually built, containing English words [Mil95]. This database contains the vast majority of English verbs, nouns, adverbs and adjectives. They are grouped by synonymy and concepts.

Benoît Sagot and Darja Fišer at INRIA started the WOLF project, *WordNet Libre du Français* ("Free French WordNet") [SF08]. The aim of this project is to translate the WordNet database in French so a synonymy database in French can be used.

The usage of such databases in the context of semantic analysis helps to improve the efficiency and precision of the Information Retrieval process.

[3] WordNet: A lexical database for English – `http://wordnet.princeton.edu/wordnet/`

3 Methodology

3.1 Approach to achieve objectives

The project is separated in two major parts:

- An analysis of the existing approaches to the problem.
- The development of a software able to chose the best approach for a given Web page.

The first part consists in looking for existing algorithms and approaches to information extraction that can be used in the context of this project. The algorithms will be run on pages that are known to be correctly entitled and results will be analysed in order to compare algorithms. The aim of this phase of the project is to determine which algorithms are useful, in which context they work at their best.

Once coherent results have been found and that best working conditions for each algorithms are known, the second part of the project will be the development of a software able to select algorithms based on criteria identified in the first part. The analysis of algorithms will give concrete results, and the software will take advantage of this results to chose the best algorithm to apply to a given Web page. This part is a practical application of the theoretical work made in the first part of the project.

The software developed will then be run on known Web pages to check its ability to analyse correctly Web pages in a known context. Once this check has been successfully passed, the software will also be run on an unknown website to check its ability to work on real world data.

3.2 Evaluation methods

In order to evaluate algorithms, a consequent amount of tests need to be run, on a wide variety of documents from different corpora and different languages. The algorithms will be tested on known documents extracted from quality websites. This allows the inclusion of real world data, while conserving a good quality standard on the evaluation material.

To test the algorithms in different contexts, a wide number of documents has to be extracted on different corpora and languages. In a first time, three corpora for two languages are considered.

The first considered corpus is news websites. This corpus has already been used in information extraction studies [1], and is considered as pertinent for Web page title analysis.

The second corpus studied is encyclopedia. Wikipedia is an example of such a corpus, and a good amount of good quality material can easily be found for this corpus.

Languages considered are English and French, which are the two languages the most known by the author.

3.3 Tests for final software

In order to check that the final software developed as part of this project behaves as expected, multiple tests campaigns will be run.

The first campaign will consist in running the software on all the extracted material that will serve as evaluation material for the algorithms. This material is known in advance and this campaign will assess the capacity of the software to rate known web page titles in known conditions. The tests results from this campaign will be compared to expected results computed manually from the evaluation material.

The second campaign that will take place is a real world application of the work, and will consist in running the software on unknown websites. The aim of this test

[1] CNN was used by Jin and Hauptmann [JH01a] and Reuters by Yuen-Hsien Tseng, Chi-Jen Lin, Hsiu-Han Chen, and Yu-I Lin [TLCL06]

	English	French
News website	CNN	Le Monde
Encyclopedia	Wikipedia EN	Wikipedia FR

Table 3.1: List of sources retained for each corpora and language combination

Website	URL	Number of pages
CNN	`http://edition.cnn.com/`	9870
Le Monde	`http://www.lemonde.fr`	13984
Wikipedia EN	`http://en.wikipedia.org`	6070
Wikipedia FR	`http://fr.wikipedia.org`	11998

Table 3.2: Internet addresses and number of extracted web pages for the retained sources

campaign is to assess the stability of the software in real world conditions, on unknown websites that can have some unexpected features. The tests results will then manually be analysed to check their coherence.

3.4 Resources

The most important resource in the scope of this project is the evaluation material that will be used to evaluate algorithms and test the final software.

Good quality website have to be identified for each corpus and language combination. The quality of the website will influence greatly evaluation results, and the quality of the website extracted has to be considered as a major criteria when choosing sources.

The sources retained for each corpora and language are displayed in table 3.1. Internet addresses for each websites are detailed in table 3.2, along with the number of web pages extracted for each one.

For each website extracted, the procedure was to look for official dumps of the web site. Dumps could be found for the Wikipedia [2]. When no official dump could be found,

[2]Static Wikipedia dumps are available at `http://static.wikipedia.org/`

web pages were extracted using httrack [3]. The data sets where then cleaned up to retain only reliable web pages, and reduced to a size processable on a single computer.

3.5 Planning of activities

A Gantt chart of the activities is provided in figure 3.1.

[3] http://www.httrack.com/

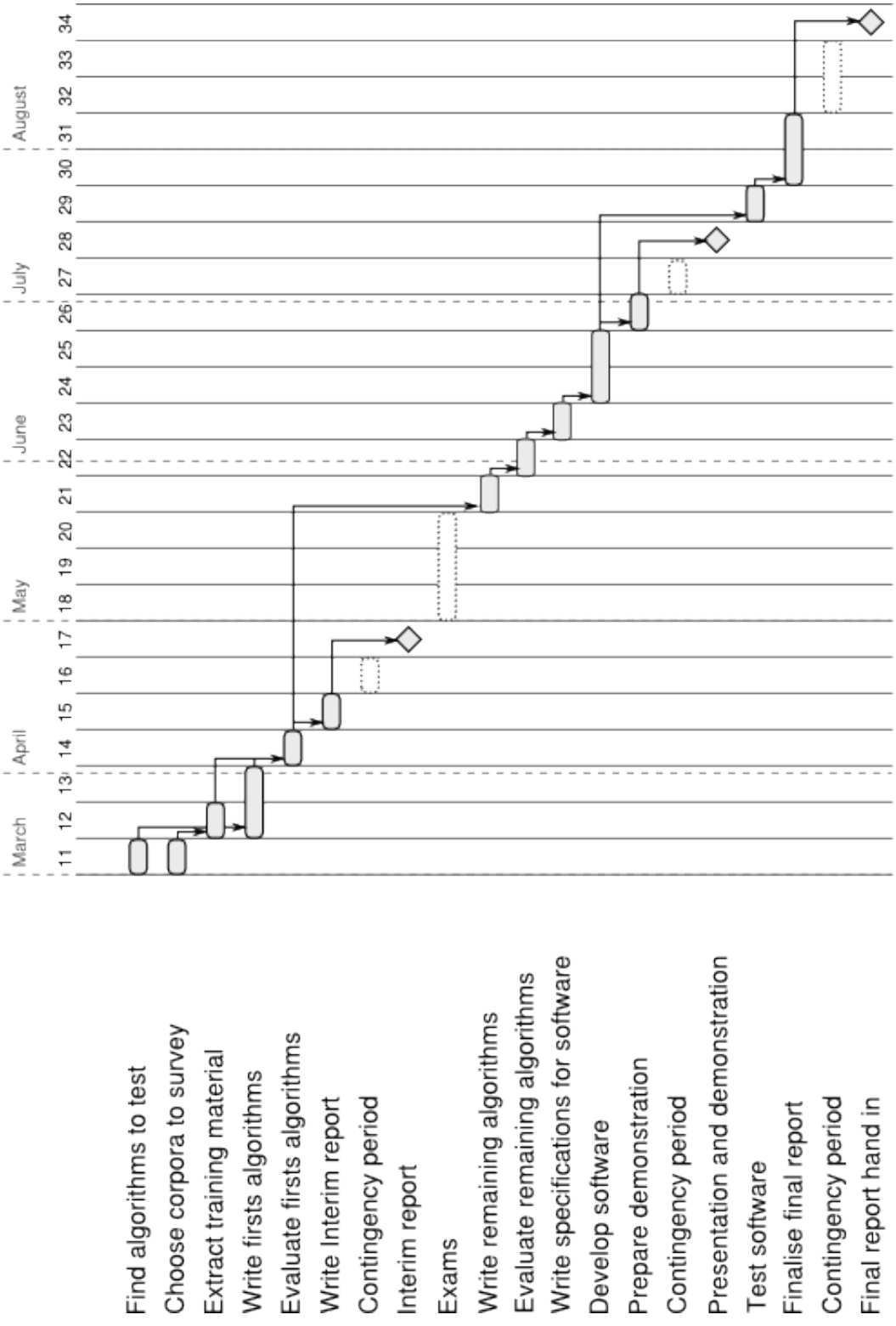

Figure 3.1: GANTT chart of the activities

4 Defining good titles

In order to rate the pertinence of Web page title, it is important to focus on the question of "What makes a *good* title?"

The notion of *good* title can be ambiguous and dependent on the context of the web page. The corpus of the web site, its language, its geographical area, its targeted audience can cause different titles for the same page to be *good* or not.

This chapter tries to summary all the techniques that relate to the entitling of web pages, and tries to determine good practices and bad practices in this area, in order to define what makes a *good* title for a web page.

4.1 Instinctive titling

Most of the time, the task of titling is instinctive. This means that no particular technique has to be applied to find a *good* title and that human judgment is the only required skill to create and analyse a web page title.

That is, a title is often considered as the summary of a document. By reading the title, the reader should be able to know what are the topic of the document, the main ideas and in a general sense, what the document is talking about.

This introduces the notion of pertinence: a web page title is pertinent if the title and the content of the web page are talking about the same things, share the same main ideas and same general sense.

Most researches in the area of title generation [KH00] [JH02] [JH01a] are using instinctive titling as a standard for evaluating computer generated titles. Instinctive titling is used through human-generated titles that are enclosed with each document.

The interest in human-generated titles as a standard in titles evaluation shows the importance of intuitive titling in the process of generating titles.

4.2 Good practices

Even if the task of titling is mostly instinctive, practices have been designed to help writers to give *good* titles to their documents.

Some of these practices are generic and can be applied to any kind of document, others are specialised in the creation of titles for web pages.

The most known practice related to the good titling of web pages is SEO, Search Engine Optimisation. SEO is not directly related to web page titling, and focuses on improving the ranking of web pages in search engines results pages. However, SEO introduces a lot of good practices related to the title of web pages.

4.2.1 General document titling

A definition for a *good* title is given by Day and Gastel in a book written in 1979, [DG06]: "We define it [a good title], as the fewest possible words that adequately describe the contents of the paper".

This definition implies a number of important and interesting concepts:

- A good title is a *description* of the contents
- A good title is *adequate* to the contents (relevant)
- A good title is rather *short* (the fewest words as possible)

4.2.2 Search-Engine Optimisation

Some Search Engine Optimisation (SEO) experts suggest that placing the title of the organisation is a good practice to ensure that searches using the name of your web site will lead to your actual site and not another one. A good quantity of important and well-known web sites are following this practice. As an example, all four web sites

considered as good material in the context of this project are including the title of their organisation in titles of their web pages:

- CNN (`http://edition.cnn.com/`) includes "CNN.com"
- Le Monde (`http://www.lemonde.fr/`) includes "LeMonde.fr"
- Wikipedia EN (`http://en.wikipedia.org/`) includes "Wikipedia, the free encyclopedia"
- Wikipedia FR (`http://fr.wikipedia.org/`) includes "Wikipédia"

In an analysis of Search Engine Optimisation practices conducted in 2005, Burdon [Bur05] states that including the name of the corporation is subject to controversy. It can be either good or bad depending on the context of the web pages, and is also subject to personal convictions.

Analysing the previous examples, Wikipedia (in its English edition), includes "Wikipedia, the free encyclopedia" in all the web pages titles. This add four extra words to all the titles, that are most of the time not directly related to the content of the page but rather on the web site on itself. This goes against the recommendations of Day and Gastel [DG06] stating that titles should be as short as possible. However, including the words "free encyclopedia" in all titles improves the referencing on search engines for the queries including these words. Including the corporation name in web pages titles is a compromise: you reduce the accuracy and pertinence of your title, but on the other hand improve the ranking of your web site in search engines.

4.3 Bad practices

In addition to a list of good practices written by experts to guide non-experts in the task of writing *good* titles, a list of bad practices emerges from the observations made on web pages considered as badly entitled.

4.3.1 Unrelated title

The biggest and most critical bad practice is to give a web page a title that is somewhat unrelated to the content of the web page. This misleads the reader, search engines, and is harmful to readers, search engines and web sites editors.

This practice comes directly in opposition with the concept of instinctive titling. An unrelated and unrelevant title can be detected really easily by a human being.

4.3.2 Uncorrectly sized titles

While Day and Gastel [DG06] recommend to write short titles that are descriptive, this recommandation can easily be overlooked in two different ways:

- The title can be too long
- The title can be so short it is undescriptive

Long titles can present an important drawback: important terms are diluted in the mass of words presents in the title. SEO techniques recommend to avoid long titles as they may be hidden in search engines. Google for example shows only the first 70 characters of each title.

Short titles have to be avoided as well as they are most of the time undescriptive and do not contain useful information related to the content of the web page. A common example of this case is the huge amount of web pages entitled “Home”. This is a description of the web page in its context, the “home” page of the web site, but does not give any information about its content.

4.3.3 Excessive repetition

Other SEO studies [ZLD09] show that using excessive repetition in titles is a common malpractice.

Excessive repetition was very common a few years ago, as a technique to improve the ranking of web pages in search engines. Most search engines are now immune to this practice and tend to decrease the ranking of web sites repeating keywords in the title.

Moreover, analysing the pertinence of repetitive titles, it appears clearly that such titles are often less relevant than others due to the fact that the same information is duplicated. Repeating words augments the total number of words without augmenting the information given about the page. This also goes against the recommandation of Day and Gastel [DG06] that titles should be as concise as possible.

5 Information retrieval

The first task of each algorithm is to retrieve information in the web page content in order to analyse its title. This process is known as Information Retrieval.

The process of Information Retrieval is a complex task and there is no universal approach. Multiple algorithms exist, each one giving different results. Also, the stage of Information Retrieval can be divided in two parts: the statistical analysis, which aims at detecting high importance words by applying statistics to the web page content; and the semantic analysis, whose objective is to give meaning to words depending on the considered language to improve the coherence of the retrieval process.

5.1 Statistical analysis

Most of the recent work done in the context of analysing web pages titles is based on a statistical approach. Statistical algorithms are an important part of Natural Language Processing, and give good results in the area of Information Retrieval.

Lots of techniques and algorithms exist, and some are more useful than others in the context of this project. This section aims at analysing in details each algorithm and tries to determine where each algorithm works the best.

5.1.1 TF-IDF

TF-IDF is one of the most used technique to identify important words in a document. It is often used by search engines to weight terms in a search query against a known set of documents.

The concept behind this approach is very simple: count the number of occurrences of

one word in one document and relate this number to the number of document the word appears in.

The TF-IDF weighting approach is described by Salton and McGill in 1986 [SM86]: the TF-IDF for a word is defined as the Term Frequency in the considered document multiplied by the Inverse Document Frequency of the term:

$$(TF - IDF_{i,j}) = TF_{i,j} \times IDF_i \tag{5.1}$$

In this equation, the Term Frequency is defined as the frequency of the term in the considered document, that is, the number of occurrences of the term divided by the number of occurrences of all terms in the document. For a term i in a document j, the $TF_{i,j}$ is defined as follow:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{5.2}$$

With $n_{i,j}$ the number of occurrences of i in j, and $\sum_k n_{k,j}$ the sum of the occurrences of all terms in the document.

The Inverse Document Frequency for a term is defined as the total number of documents divided by the number of documents containing the term. This measure is logarithmically scaled. For a term i, the IDF_i is defined as follow:

$$IDF_i = log(\frac{|D|}{|j : t_i \in d_j|}) \tag{5.3}$$

Where D is the set of all documents, d_j the document j and t_i the term i.

This technique is very interesting because it focuses only on rare words that are found in few documents, and reduce the importance of very common words.

As an example, the inverse document frequency was computed for some CNN press releases issued between 2008 and 2011. Some results are presented in Table 5.1.

It is interesting to note that very common words such as “the”, “in”, “to” and “of” have a very low IDF, whereas some more uncommon words such as “obese”, “photograph” and “championships” have a higher IDF. This means that very common words are under-weighted compared to uncommon words when computing the TF-IDF score for a term.

It is also worth noticing that the highest IDF are allocated to the rarest words found in

Term i	$\lvert j : t_i \in d_j \rvert$	IDF_i
of	9870	0.00051
in	9868	0.00071
to	9866	0.00091
the	9793	0.00834
and	9751	0.01264
tuesdays	594	2.81088
growth	572	2.84862
positive	563	2.86448
style	557	2.87520
sea	555	2.87879
obese	80	4.81573
premature	77	4.85396
championships	73	4.90730
photograph	72	4.92110
suburb	65	5.02337
everywehre [*sic*]	1	9.19776
exposé	1	9.19776
extinguish	1	9.19776
fanta	1	9.19776
glycerin	1	9.19776

Table 5.1: IDF score for some terms extracted from CNN press releases

Term i	$n_{i,j}$	$\lvert j : t_i \in d_j \rvert$	$TF_{i,j}$	IDF_i	$TF - IDF_{i,j}$
messages	21	339	0.016419077	1.464337406	0.024043069
bosworth	9	16	0.007036747	2.790417121	0.019635460
seligstein	5	1	0.003909304	3.994537104	0.015615860
messaging	9	76	0.007036747	2.113723512	0.014873738
zuckerberg	6	58	0.004691165	2.231109110	0.010466500
undertakings	3	3	0.002345582	3.517415849	0.008250389
beluga	3	6	0.002345582	3.216385853	0.007544298
communication	6	283	0.004691165	1.542750668	0.007237297
simplify	3	18	0.002345582	2.739264599	0.006425171

Table 5.2: Highest TF-IDF score for a CNN press release

the corpus, including spelling errors ("everywehre" [*sic*]), uncommon spelling for foreign words ("exposé") and brands ("fanta", "glycerin").

As an experiment, TF-IDF for terms present in a specific press release were computed. The results for the terms with the highest TF-IDF are shown in table 5.2

The original title for the press release was "Messages feature among Facebook's 'biggest undertakings' – CNN.com" [1]

Out of the nine words in the original title [2], three are found in the terms with the highest TF-IDF for the article. Also, although very present in the text of this article (34 occurrences), the term 'Facebook' have a very low TF-IDF. This can be explained by the very low IDF of the term, which appears in 9811 documents out of the 9870 extracted (more than 99% of the documents). This is due to the "Share on Facebook" button that can be found next to every CNN press article. The same explanation can be applied to the terms "CNN.com", that are present on all the pages of the web site.

[1] The complete press release can be found at `http://edition.cnn.com/2011/TECH/social.media/04/07/facebook.messages/`

[2] messages, feature, among, facebook, s, biggest, undertakings, cnn and com

5.1.2 Naïve Bayesian

Naïve Bayes classifiers have been used widely in statistics. With the introduction of statistical natural language processing, Naïve Bayes classifiers have also been used to process natural language.

The basis of this approach is to determine the probability for each word of being included in the summary of the corresponding text. Once this probability is known for all the words of the considered text, the words having the highest probability are selected for inclusion in the generated summary.

In order to use this technique to extract important words regarding the title of a web page, the probability of a word being present in the title given that it is present in the text of a page has to be computed.

The formula to compute this probability is the following:

$$BC_i = \frac{|j : t_i \in T_j|}{|k : t_i \in d_k|} \tag{5.4}$$

Where t_i is the term i, T_j the title of the document j and d_k the document k itself.

This algorithm doesn't focus on the actual importance of a word, but more on its usual appearance in titles. Instead of answering the question "Which words are important?", this algorithm focuses on the question of "Which words are usually included in the title of a web page?".

An experiment conducted on the CNN document set (more than 9000 CNN press releases issues between 2008 and 2011) showed better results than a TF-IDF based approach.

As an example, the same document as the previous experiment is considered. On the press release titled "Messages feature among Facebook's 'biggest undertakings' – CNN.com", a Naïve Bayes classifier extracted four out of the nine words present in the title. Those words where (in the order returned by the algorithm) "com", "cnn", "undertakings" and "s".

It is interesting to note that some of the most important words for this article, according to the algorithm, were "to", "the" and "in". Unlike TF-IDF, this algorithm has a tendency to return common stop words as interesting, where TF-IDF excludes those

Term i	$\lvert j : t_i \in T_j \rvert$	$\lvert k : t_i \in d_k \rvert$	BC_i
com	5746	10629	1.8498085625
cnn	11439	10629	0.9291896145
undertakings	3	1	0.3333333333
zuckerberg	68	11	0.1617647059
to	11740	1834	0.1562180579
s	11649	1771	0.1520302172
the	11794	1569	0.133033746
in	11757	1507	0.1281789572
microsoft	237	30	0.1265822785

Table 5.3: Top words returned by a Naïve Bayes classifier for the CNN press release "Messages feature among Facebook's 'biggest undertakings' – CNN.com"

words from results.

The head of the list of words returned by the algorithm for the considered article is shown in table 5.3.

The probability for a term to be present in the title is presented in table 5.4 for some representative terms.

It is also worth noticing that out of the 84336 extracted terms, 72870 (86%) have a probability of being present in the title strictly equals to zero, meaning they don't appear in the title of any of the training documents.

5.1.3 Witbrock and Mittal Naïve Bayes approach

Witbrock and Mittal used a different approach to the extraction of important words based on a Naïve Bayes classifier computed differently than the one described in the previous section.

In their 1999 paper [WM99], instead of trying to determine the probability for a word of being present in the title given the occurrences of this word in the title of documents, they try to determine the correlation between terms present in the document and terms

Term i	$\lvert j : t_i \in T_j \rvert$	$\lvert k : t_i \in d_k \rvert$	BC_i
cnn	11439	10629	0.9291896145
jonestown	6	5	0.8333333333
klitschko	5	4	0.8
neymar	4	3	0.75
outsells	4	3	0.75
resigns	19	9	0.4736842105
djokovic	55	25	0.4545454545
attiyah	18	8	0.4444444444
malmo	18	8	0.4444444444
dortmund	9	4	0.4444444444
palin	110	27	0.2454545455
wikileaks	53	13	0.2452830189
giffords	54	13	0.2407407407
chilean	46	11	0.2391304348
ireporters	172	41	0.238372093
reserved	10630	0	0
advertise	10628	0	0
podcasts	10628	0	0
mixx	10565	0	0
stumbleuponi	10565	0	0

Table 5.4: Probability for words to be present in the title of a web page according to Naïve Bayes classifier

present in the title of a text.

The major difference between this algorithm and the previous one is the set of words considered. In the previous algorithm, the occurrence of a specific word in a document would only influence the probability of this word appearing in the title. In the Witbrock and Mittal approach, the presence of a word in a text can also influence the probability of other related words to be included in the title.

The statistical probability $P_{i,j}$ for a term i to be included in a document j is defined as:

$$PD_{i,j} = \sum_{k} n_{k,j} \times P_{i,k} \tag{5.5}$$

Where $n_{k,j}$ is the number of occurrences of the term k in the document j and $P_{i,k}$ the probability for the term i to appear in the title of the page given that k is present in the page.

The conditional probability $P_{i,j}$ between two terms i and j is defined as:

$$PT_{i,j} = \frac{|k : t_i \in Tk \wedge t_j \in d_k|}{|l : t_j \in d_l|} \tag{5.6}$$

Where $|k : t_i \in Tk \wedge t_j \in d_k|$ is the number of documents k having the term i in the title and the term j in the content of the page, and $|l : t_j \in d_l|$ is the number of documents containing the term j.

Due to the compexity of this algorithm, a huge amount of data has to be collected during the learning approach. This makes this algorithm very complex to implement, and it requires a lot of computer resources.

Also, while adding new interesting concepts in the way of scoring terms in the contents of a page, this approach is similar to a Naïve Bayes classifier technique and is not radically different.

For this reasons, this algorithm was not tested as part of this project, but it could be considered in further developments of this project.

5.1.4 Comparison of statistical algorithms

Statistical algorithms were all ran on the same data sets in order to test their accuracy on the extracted training documents.

Algorithm	CNN	Le Monde	Wikipedia EN	Wikipedia FR	Average
TF-IDF	0.19346	0.14568	0.16382	0.18669	0.17129
Naïve Bayes classifier	0.30741	0.30593	0.54493	0.29619	0.33809

Table 5.5: Average F_1 for statistical algorithms

The data sets included all extracted websites described in the "Methodology" chapter. This includes four websites, representing two corpora and two different languages.

Once title were extracted for all the pages, the accuracy of the retrieval task was measured using the F_1 measure. For each algorithm, the average F_1 result was computed.

The full average results are shown in table 5.5.

5.2 Semantic analysis

The statistical analysis tries to analyse only the form of words that is present in the text and is not aware of the meaning of each word.

While this can give surprisingly good results, words are used in the human language to share ideas and they have a meaning. The semantic analysis is a process aimed at improving the coherence of the Information Retrieval process by trying to extract the meaning behind words.

This process of semantic analysis can contains multiple steps. Multiple techniques are evaluated in this chapter, in order to understand which ones are useful, and which are not:

- Reduction of word inflections
- Detection of synonyms
- Removing words with no meaning

5.2.1 Reducing inflected words

Inflected words are words sharing the same root, but on a different form imposed by the language grammar. Since they have the same root, they carry the same meaning, but using a different spelling.

Common examples of inflected words include:

- Plurals. The two words "message" and "messages" represent the same thing, and the "s" in the second one has to be removed.
- Conjugated verbs. The words "message", "messaged" and "messaging" are three different spelling for the verb "message". They can all be reduced to the same form.
- Gender variation. In some non-English languages, words can change depending on the gender. This is very common for animal names. In French for example, "chien" refers to a male dog and "chienne" to a female dog.

The process of reducing the word inflections to retrieve the basic form of each term is called "lemmatisation". The base form of a word is called a "lemma".

The lemmatisation process can be done using two different techniques.

- The first technique involves using a dictionary to look-up each word and retrieve the corresponding lemma
- The second uses a deep language knowledge to remove common suffixes from words and retrieve algorithmically the basic form of each term

It is also worth noticing that some languages doesn't use inflections at all. Mandarin Chinese is an example of this specific case. Such languages don't require to reduce word inflections when the process of Information Retrieval is applied to them.

Dictionary-based lemmatisation

Dictionaries containing a list of words and their corresponding basic form can be found and reused.

WordNet is such a dictionary, including most of the words of the English language.

A dictionary based approach provides efficient and reliable stemmisation, but might fail if the dictionary is incomplete.

Also, dictionaries can be found and/or created for all existing languages.

Dictionary-based lemmatisation requires to look-up in a dictionary containing a list of words each term found in the original document. This step is very costly and a dictionary-based lemmatisation can slow down the process of information retrieval.

One of the most important problem when using a dictionary to reduce word inflections is the treatment of words that aren't present in the dictionary. This can happen in two specific cases:

- The word is too common (for example, stop swords such as "the", "or", "of" or "and"), and doesn't carry a real meaning
- The dictionary is incomplete and doesn't know the word (this case is common when dealing with proper nouns, for example "Facebook", "Obama" or "Strauss-Kahn")

A different behaviour has to be applied depending on the case we encounter:

- If the word is a stop-word and doesn't carry any meaning, it can safely be removed
- If the word is just unknown in the dictionary because it is incomplete, the word has to be kept

There is no simple solution to this problem. Keeping all the unknown words would keep words with no meaning. Removing all the words would remove words carrying actual meaning. Both approaches would reduce the accuracy of the Information Retrieval process.

A good approach would be to conserve a list of stop words, and remove these stop words before running the lemmatisation. If all stop words are already removed from the list of words to process, all unknown words can safely be kept. The removal of stop words is discussed later in this chapter.

Original word	Reduced word
message	messag
messages	messag
engine	engin
engineer	engin
simplify	simplifi
simplification	simplif
facebook	facebook

Table 5.6: Sample of words reduced using the Porter stemmer

Algorithmic stemming

Algorithmic stemming is the process of removing common suffixes to words to extract their basic root.

Stemming algorithms are usually written for a given language and require a deep knowledge of the specific grammar of this language. Since all grammars are different, an algorithmic stemmer usually works only for a known and restricted set of languages.

While these algorithms work usually well for known language and provide acceptable results, the accuracy of such a stemmer is usually lower than a dictionary based one. The main advantage of algorithmic stemming over dictionary-based is the ability to reduce all words, including words not present in traditional dictionaries.

Another problem is that a stemming algorithm is language-dependent and writing a stemmer for a new language is a complex task that requires a deep linguistic knowledge of the target.

A well known algorithmic stemmer for the English stemmer is the Porter stemmer, named after Martin Porter, who wrote it in 1980 [Por80]. This algorithm is able to reduce all common suffixes for the English language. Table 5.6 shows a sample of words reduced using the Porter stemming algorithm.

Analysing these results shows interesting facts about algorithmic stemming:

- “message” and “messages” are correctly reduced to a simpler common form. The

same can be said about “engine” and “engineer”.

- “simplify” and “simplification” are both reduced, but to different forms, although they are both from the same root.
- “facebook” is left untouched
- Lastly, the stems extracted aren’t correctly spelled words. “messag” [*sic*] and “engin” [*sic*] are correct roots, but incorrectly spelled.

Hybrid approach

A hybrid approach exist, halfway between the algorithmic and dictionary-based ones.

This hybrid approach tries to algorithmically remove the suffixes of a word, and matches in a dictionary to see if the corresponding words exists.

This approach has multiple advantages:

- It produces only existing words, where pure algorithmic stemmers can produce unfinished words (such as “messag” [*sic*] instead of “message”).
- It matches much more original words than a pure dictionary based approach. For example, this approach can match plurals of words where the word exists in the dictionary without its plural form

It also comes with some drawbacks:

- Such an approach is still unable to reduce words that are totally absent from dictionaries
- A dictionary has to be compiled for the language to process, where a pure algorithmic approach doesn’t need one
- An algorithmic stemmer has to be written, where a dictionnary-based approach doesn’t require one

This approach is implemented in Java in the JWI library [3]

[3] JWI: Java Wordnet Interface ; `http://projects.csail.mit.edu/jwi/`

Comparing inflection reduction techniques

Comparing all the techniques to reduce word inflections, only one seems appropriate in the context of this project.

A pure algorithmic stemmer generates incomplete words ("messag [*sic*]" for example, which is unacceptable in the context of this project. Incomplete words would break the following steps in the process, especially the synonyms detection.

A database of inflections seems envisageable, but comes with the huge drawback that unknown words can't be stemmed. A big database has to be found, but none are exhaustive and removing unknown words is unacceptable as well. Removing unknown words could result in loosing important imformation in the contents of the page.

The best approach is to use an hybrid algorithm, combining an algorithmic stemmer and with a database of words. The algorithmic stemmer takes care of reducing words, and the database allows the algorithm to match words to find existing forms of the terms. The implementation of such an hybrid stemmer present in JWI can be used in this project.

Another important feature in the stemmer is the conservation of unknown words. If a word can't be stemmed and can't be found in the dictionnary, it has to be kept in its current form and can carry important meaning for the rest of the Information Retrieval process. If the word is a stop word, it will be removed later in the process.

5.2.2 Extracting concepts from terms

Once the most important terms have been cleaned and inflections have been removed, an important part to improve the coherence of the Information Retrieval process is to detect synonyms and words sharing the same meaning [AOG98].

This can only be done using databases of synonyms and concepts.

Such a database that was already used in the context of this project is Wordnet [Mil95]. Wordnet contains not only a list of words, but also groups words into synsets (sets of synonyms) that can be used to detect all the synonyms and reduce them to a common form. For example, "undertaking" is a synonym of the terms "project", "task" and

"labor", and all four of them can be used to convey the same idea.

Also, some words can convey ideas about a concept without being directly synonyms of the concept in itself. "message" for example is related to the concept of "communication", but the terms are not synonyms. If the terms "message" is used by itself in the contents of a page, the concept of messaging is probably the intended meaning. However, if the contents contains multiple occurences of the terms "message", "publication" and "document", the contents is more likely to be about the broader concept of "communication".

The aim of the concept extraction phase of the Information Retrieval process is to detect all these words and relations between concepts and try to determine which concepts are expressed in the contents of the page.

Wordnet contains a list of synonyms and concepts, and synonyms can easily be reduced to their common form. Also, Wordnet can retrieve the broader concept of any term. This database can safely be used in this context.

The biggest problem is the extraction of concepts from general terms. While "message" can be retrieved and eventually transformed to the general concept of "communication", it is unclear at what point this transformation should occur. It depends on the occurence of different terms in the text that could be related to the same concept of "communication". The broader concept of a given term is called his hypernym.

Two approaches are valid to solve this issue:

- Try to simplify the words based on the statistical occurences of other synonyms. This require to have a knowledge of all the other words in the contents of the page
- Another approach consists to extract all the possible meanings for the same word and consider them as potential candidates. Statistically, the probability of having synonyms for the correct meaning is higher to the probability of having synonyms of unrelated meanings, and the correct synset will emerge on top of the statistical analysis.

The first solution is very error-prone and there is no chance of recovery if the wrong synonym group is selected at a certain point. The second solution offers the ability of recovering when wrong synsets are deduced. Statistically, the results balance themselves

and the correct meaning of each term is correctly extracted.

The same can be deduced about concepts extraction. For a given term, both the term and the hypernym can be considered as credible candidates. Statistically, the most important of the two will be statistically extracted where the least important will be underweighted.

While this technique can be applied to the content of the text, where statistics can have an effect due to the important size of the web page, it is impossible to apply the same process to the title of the web page. The title of a web page being short, the statistical balance doesn't apply and erroneous synonyms considered won't be underweighted by any statistical algorithm. The title is hence kept intact and synonyms aren't simplified. On the other end, if a synonym is used in the title, it will also occur in the text as a synonym of another word. Simplification of synonyms in the title is not required.

The technique applied in this project works the following way: for each pair of words found in the contents of the web page, if the two terms are synonyms, both are reduced to the basic form of the synset they belong to. This way, only one form of each synonym is conserved and the number of occurences of important terms is augmented.

Experimental results ran with synonyms detection showed that TF-IDF is greatly improved by this process. A Naïve Bayes classifier is also improved slightly, but not significantly.

5.2.3 Removing stop words

Stop words are words that are relevant to the grammar of a language but do not carry a relevant meaning. Examples of such words are "the", "for", "at" or "else". They are, in a sence, semantically neutral.

Such words can be safely removed during the Information Retrieval process, as they augment the number of words without carrying meaningful information.

While the TF-IDF extraction algorithm automatically discards words with a high occurence frequency due to its inner working, experimental results showed that a Naïve Bayes classifier algorithm doesn't discard such terms. This results in the inclusion of stop words in the top terms returned by the Information Retrieval process, and reduces

the accuracy of the algorithm.

Considering this, removing stop words become an important part of the process.

Multiple techniques exist to remove stop words. Some techniques automatically exclude all terms that occurs too frequently in the content of the considered corpora, considering those terms as too frequent to be meaningful [WS92]. Another technique consist in removing words according to a database of stop words, also known as a stop list. Stop lists for most languages have been compiled and can be reused [Fox89].

While the first technique does not require a database to be embedded in the software, its accuracy can be biaised by the considered corpora. For example, on the "CNN.com" website, the words "CNN" and "com" are often present in the title and are part of the web site identity. They should not be considered as stop words as they carry an important meaning for the web site.

The second technique offers better results but requires a database to be compiled for each language considered. A lot of databases already exist, and can be reused. Stop lists exist for both English [Fox89] and French (as part of the Snowball project) [4].

5.2.4 Influence of semantic analysis on algorithms efficiency

While the pure statistical algorithms showed surprisingly good results, the semantic analysis improves the accuracy and the coherence of these algorithms.

The reduction of inflected words is an important step to improve the coherence in the information retrieval. It reduces common word derivations in order to group together words sharing the same meaning.

Then, the detection of synonyms and semantic concepts helps improve this coherence further by grouping together words of the same concepts and extracts ideas behind words instead of considering only strict words meanings.

On average, the whole semantic analysis process improved the accuracy of the Information Retrieval by up to 16% on the considered corpora. The semantic analysis is hence a fundamental part of the Information Retrieval process.

[4] Snowball project, `http://snowball.tartarus.org/`

5.3 Conclusion on Information Retrieval

In conclusion, the experiments on Information Retrieval algorithms showed that on average, a Naïve Bayes classifier approach is more accurate than a TF-IDF one.

While both algorithms have slightly different objectives (TF-IDF tries to extract meaningful words, while Naïve Bayes tries to determine which words are usually present in titles), the approach based on determining which words are likely to be present in the titles seems to be more accurate when analysing titles on a single website.

This is likely to be due to the presence of recurring words in the titles of all the web pages accross web sites. For example, the CNN website includes "CNN.com" in all its titles. A Naïve Bayes approach is more likely to detect this technique than a TF-IDF one.

Considering the semantic analysis, the reduction of inflected words is a fundamental part of the Information Retrieval process in order to improve the results of the whole extraction. In the same way, synonyms detection and removal of stop words help improve the results of the Information Retrieval process.

It is also worth noticing that while the semantic analysis helps improving the results of the TF-IDF statistical algorithm, it has little effect (and sometimes, even negative effects) on the Naïve Bayes classifier. This can be explained by the basic results of a Naïve Bayes classifier, where most of the retrieved words are words carrying little or no meaning (stop words, words appearing often on a particular web site).

In conclusion, none of the algorithms are perfect, and both can be used depending on the use case. TF-IDF tends to extract words specific to a web page, where Naïve Bayes tries to extract general titling behaviours and terms often included in titles. Both algorithms can be interchanged and can give different results on different pages.

6 Scoring titles

Once the information extraction step has been performed, the next step is to give a score to a title depending on the results of the previous step.

6.1 Metrics to score titles

Multiple algorithms to score existing titles for web pages exist.

While F_1 has been used for a long time in the context of extracting and generating titles for web pages, other metrics have advantages and are worth considering:

- Average Precision
- Spearman's measure
- Discounted cumulative gain

This section focuses on comparing the different metrics in order to choose one that would be reliable in the context of this project.

6.1.1 F_1 measure

The F_1 measure is a technique to compare titles introduced by Rijesbergen in 1979 [Rij79], and has been reused a lot since. It is considered by many as the standard measure to evaluate the Information Retrieval process in a system [JH02] [WM99] [JH01b] [JH01a].

The F_1 measure uses the notion of "precision" and "recall", also common in Information Retrieval evaluation.

The "precision" is the proportion of relevant document found during the Informa-

tion Retrieval process. The "recall" is the proportion of relevant documents that were retrieved. They are defined mathematically as:

$$presicion = \frac{|retrieved \cap relevant|}{|retrieved|} \tag{6.1}$$

$$recall = \frac{|retrieved \cap relevant|}{|relevant|} \tag{6.2}$$

The F_1 measure in itself is defined by the following equation:

$$F_1 = \frac{2 \times (precision \times recall)}{precision + recall} \tag{6.3}$$

Variants of the F_1 measure

The F_1 measure is a specific case of the F-measure derived by Rijesbergen in 1979 [Rij79].

In the F_1 measure, the same importance is given to the precision and to the recall of the considered sets.

It is possible to give different weights to the precision and recall of the evaluation using the following equation:

$$F_\beta = \frac{(1 + \beta^2) \times (precision \times recall)}{(\beta^2) \times (precision \times recall)} \tag{6.4}$$

Considering this equation, the F_1 measure is a specific case of the F measure with β equals to 1.

Two widely used variants are the F_2 and the $F_{0.5}$ measures, which give respectively more importance to the precision and recall, with a factor 2 difference. The F_2 measure was used in the context of full-text indexing in a paper by Gay et al. in 2005 [GKA05]. While theorically usable, not previous usage of the $F_{0.5}$ measure in Information Retrieval systems was found in the recent litterature.

6.1.2 Average precision

Another interesting metric is the Average Precision. Compared to F_1, this metric has the advantage of taking in account the rank of the retrieved document.

This metric has not been widely used used in the context of scoring and evaluating titles, but it can be adapted to be used in the context of this project. Also, this metric is very popular in the area of search-oriented Information Retrieval [TS06].

It is defined by the following equation:

$$AP = \frac{\sum_{r=1}^{N}(P(r) \times rel(r))}{|RT|} \tag{6.5}$$

Where N is the number of retrieved elements, RT is the set of all the relevant terms, $rel(r)$ a binary function on the relevancy of the term r and $P(r)$ is the precision of the term of rank r.

rel and P are defined as follows:

$$rel(r) = \begin{cases} 1 & \text{if } r \text{ is relevant} \\ 0 & \text{if } r \text{ is not relevant} \end{cases} \tag{6.6}$$

$$P(r) = \frac{|RRT_r|}{r} \tag{6.7}$$

With RRT_r being the set of retrieved terms of rank r or less.

The most important feature of this metric is the rank it gives to all the relevant elements. This allows to differenciate titles containing the most relevant words against titles containing less relevant words.

Also, it only considers the n most importants terms retrieved from the Information Retrieval process, n being the number of terms in the title. This implies that adding terms to the title maintains the score for the given title if and only if the added term is the next most relevant term of the content of the page. If the added term is of a lesser importance, the score will be decreased for this title. This follows Day and Gastel [DG06] statement that a good title is "the fewest possible words that adequately describe the contents".

6.1.3 Other measures

Other measures to rate the efficiency of Information Retrieval process exist. Those include:

- Spearman's measure

- Discounted cumulative gain

While these measures were used in some research to rate the efficiency of Information Retrieval systems, none was used in the context of analysing titles.

The F_1 measure was the only one used in all the research done on the subject of title generation and analysis. Also, the Average Precision added interesting features in the context of this project. This seems to be the only two reliable measures usable in this conditions.

6.2 Adapting the measure for this project

The Average Precision measure is adaptable to all Information Retrieval systems, but was initially designed to focus on documents retrieved from a database given some keywords. In this application, the set of retrieved elements was all the documents returned by the system, and the set of relevant documents was the documents in the database relevant to the query.

This measure needs some adaptation to be used in the context of rating existing titles.

In the case of this project, where the title is the subject of the analysis, the terms in the existing title for a web page are considered to be the retrieved elements. The words returned by the Information Retrieval process on the content of the web page is considered to be the relevant elements.

Given this, we can define the set *retrieved* as the set of words in the title of the web page, and *relevant* the set of words returned by the web page analyser.

The Average Precision measure can then be applied using the formulas given earlier. This gives the following formula to compute the Average Precision for a document i:

$$AP_i = \frac{\sum_{k \in T_i} (P(k) \times rel(k,i))}{|T_i|} \tag{6.8}$$

$$rel(k,i) = \begin{cases} 1 & \text{if } k \in d_i \\ 0 & \text{if } k \notin d_i \end{cases} \tag{6.9}$$

$$P(k) = \frac{RRT_k}{rank(k)} \tag{6.10}$$

Where T_i is the title of the document i, d_i the contents of the document i, $rank(k)$ the rank of the term k in the list of words returned by the Information Retrieval process and RRT_k is the numbers of terms in the title having a rank lesser than $rank(k)$.

6.3 Detecting synonyms in title analysis

One important aspect of title scoring is the detection of synonyms.

While words included in the title are compared to words extracted from the Information Retrieval process, it is important to keep in mind that the title stays relevant if a synonym is used instead of another word.

This implies that the scoring algorithm has to detect synonyms as well. The process is the following: instead of comparing the words in the title and the terms extracted for exact matches, we can consider two terms to be equals if they are in the same synset.

Applying this technique, synonyms are correctly detected during the scoring, and using a synonym instead of another word in the title won't decrease the score of the title.

As an example, if the Information Retrieval process extracts the word "undertaking", using either "undertaking", "project" or "labour" in the title will be considered as relevant.

6.4 Conclusion on titles scoring

Multiple metrics exist to compare titles to words extracted during the Information Retrieval process.

A detailed analysis of each algorithm shows that while F_1 has been widely used, an Average Precision metric can also give interesting results due to its ability to score terms relative to their position in the stream of words returned by the Information Retrieval process.

7 Development of a prototype

Once all the experimental results were extracted, a prototype was developed to apply the conclusions of the theoretical analysis to real world web sites.

7.1 Specification of the software

The prototype will be a small console-based application, allowing a user to train the software on known web pages and then evaluate an unknown web page using the results of the training phase.

The software has to allow both the operations to be performed. The results of the traning phase have to be stored in a specific folder in order to be reused during the evaluation.

7.1.1 Technical specification

Command line arguments

The program has to perform two different tasks: training and analysis.

When starting the prototype, the user has to specify which task has to be performed. This is done using a command line argument.

More over, each task might require one or more arguments to specify the behaviour of this task. In the case of the training phase, the folder containing all the training documents has to be given. When analysing a document, the path to the document is added.

Listings 7.1 and 7.2 show sample command line that could start a traning or analysis

Listing 7.1: Sample command line for traning phase

```
prototype train path/to/training/docs/folder
```

Listing 7.2: Sample command line for analysis phase

```
prototype analyse path/to/analysed/doc.html
```

process, respectively.

Storage folder

All the results of the training phase have to be stored in a folder so they can be reused later on, during the analysis of a file.

A hidden folder in the user home directory (as commonly created by Unix programs) is more than sufficient to store the training data.

7.1.2 Algorithms to use

Statistical algorithm

The experiments showed that a Naïve Bayes classifier approach has better results than a TF-IDF approach in the context of this project.

This algorithm should be used as a default for statistical analysis, but could be replaced by a TF-IDF algorithm if required.

Semantic analysis

The experiments also showed that semantical analysis improves greatly the results of the Information Retrieval process.

In the working prototype, a full semantic analysis has to be enabled by default. This includes performing a reduction of word inflections, as well as a detection of synonyms.

The word inflections and synonyms detection could be, however, disabled at some point if the user requires it.

Ranking algorithm

The analysis of the different ranking algorithms suggested that the Average precision measure was the most suited metric to score titles. This implies that titles will be ranked using this metric by default.

The F_1 ranking metric will also be implemented, and can be used instead of an Average precision, if required by the user.

7.2 Technical choices

7.2.1 Programming language

The software was developed using Java as a programming language.

Java was selected due to the large amount of libraries available related to the context of this project. Java is a mature programming language for which lots of tools have been developed and can be reused to make the development of tools easier.

Also, Java is the programming language the best known to the author.

The choice of Java as a programming language made a complete focus on the Natural Language Processing techniques possible, without any problem related to the programming language knowledge or lack of libraries.

7.2.2 Required dependencies

The project requires multiple dependencies to make the development of the prototype easier. These dependencies include:

- A HTML parsing library, to parse HTML files during analysis
- Libraries to access synonyms databases, in particular Wordnet.

Using Java as a programming language, this offers choices for the different libraries to use.

HTML parser

A few HTML parsers exist in Java:

- JSoup [1]
- Mozilla Java Html Parser [2]
- HTML Parser [3]

Java Mozilla Html Parser wasn't considered because, despite its name, it is not a Mozilla-supported project. It is a Java API wrapper around the Mozilla web browser, making its deployment harder than any other library.

Jsoup and HTML Parser are both pure Java libraries. Jsoup was retained in the end because its API is much simpler to use, and it provides good performance.

Wordnet parser

The official Wordnet project [Mil95] was written in Prolog. While Prolog can't be easily used in a Java application, the Wordnet database files can be downloaded separately [4], and numerous Java implementation to parse Wordnet files can be found.

The most known implementations are:

- JWI (Java Wordnet Inteface), a MIT project [5]
- JWAS (Java API for Wordnet Searching) [6]
- JWNL (Java WordNet Library) [7]

[1] Jsoup: Java HTML Parser, `http://jsoup.org/`
[2] Java Mozilla Html Parser, `http://mozillaparser.sourceforge.net/`
[3] HTML Parser, `http://htmlparser.sourceforge.net/`
[4] Download – Wordnet, `http://wordnet.princeton.edu/wordnet/download/`
[5] JWI (the MIT Java Wordnet Interface), `http://projects.csail.mit.edu/jwi/`
[6] Java API for WordNet Searching (JAWS), `http://lyle.smu.edu/~tspell/jaws/index.html`
[7] JWNL (Java WordNet Library), `http://sourceforge.net/projects/jwordnet/`

No implementation covered completely all the need of this project. JAWS provides good search capacities for synonyms detection, whereas JWI provides a good stemming algorithm based on Wordnet.

As a consequence, a combination of two libraries was used. JWI was used during the stemmisation process to reduce word inflections, and JAWS was used during the synonyms detection phase.

7.2.3 Data storage

Data extracted during the training phase needs to be stored between executions. A data storage engine has to be used to store this data and make sure it doesn't get lost.

This prototype being a fairly simple software, and the data required for storage being really simple and unstructured, a simple key-value dictionary is required.

Setting up a complete key-value based data storage engine would be too complex in the context of this project, and a simple, in-memory, hash table is more than enough to store the required data.

Once the data are complete, a hash table can be stored to a file using Java's serialisation mechanism. This provides an easy to implement data storage system, with very little overhead.

7.3 Implementation of the specification

The software is implemented as a Maven project, which generates an executable JAR file, that can be run on any platform using the Java Virtual Machine.

7.3.1 Statistical algorithms

Statistical algorithms (TF-IDF and Naïve Bayes) were implemented according to the formulas given in the previous chapters.

In order to apply these formulas, the algorithms need to acquire data on training documents. In the case of TF-IDF, this is the Inverse Document Frequency for each

term. For the Naïve Bayes classifier, the data required is the probability for each term of being included in the title. Data is acquired by each algorithm during a training phase.

During the training phase, algorithms acquire this precise information:

- TF-IDF recolts information about the Inverse Document Frequency:
  - Number of document in which each term appears
  - Total count of documents
- Naïve Bayes classifier gets information to compute the probability for a term if being included in the title:
  - Number of documents in which each term appears in the contents
  - Number of documents in which each term appears in the title

In order to represent in the code the two stages of each statistical algorithm, training and analysis, an interface was created. Each algorithm implement this interface and conforms to the training and analysis process. The common interface for statistical algorithms is showed in listing 7.3. The interface shown also includes two methods allowing algorithms to save their state between executions.

The complete implementation of statistical algorithms can be found in the appendices. Listing 9.1 shows the complete implementation of the TF-IDF algorithm, listing 9.2 shows the Naïve Bayes classifier.

7.3.2 Parsing words

Statistical algorithms have to parse streams of characters to extract words from them. This task is executed by a parser that takes a Java String object, and returns multiple objects for each word that was found in the text.

Multiple parsers can be considered, and a generic interface defines the behaviour of word parsers. The interface is shown in listing 7.4

A default implementation was written for this parser. The default implementation consider a word is a sequence of letters separated by one or more non-letters characters.

Listing 7.3: Interface implemented by statistical algorithms

```
public interface WordExtractor {

    /**
     * Trains the word extractor on the given document.
     */
    public void train(Document doc);

    /**
     * Processes a document to extract its most important words.
     */
    public List<String> extract(Document doc);

    /**
     * Saves the state of this current extractor to
     * the specified output stream.
     */
    public void save(OutputStream out) throws IOException;

    /**
     * Loads the state in the current extractor from
     * the specified input stream.
     */
    public void load(InputStream in) throws IOException;

}
```

Listing 7.4: Interface defining the behaviour of parsers

```
public interface WordParser {

    /**
     * Returns the next word in the text stream, or null if
     * the end is reached
     */
    public String next();

}
```

7.3.3 Word inflection reducer

The word inflection reducer was implemented as a wrapper around a parser.

The reducer implements the same interface as parsers, and delegates the word extraction process to another parser. For each word extracted by the original parser, the reducer removes any inflection that can be found on the word, and returned the basic form of the word.

This way, word inflection reduction can easily be injected in any statistical algorithm without having to change the algorithm structure.

Word inflection reducers were implemented two ways:

- The first is an implementation using the Porter stemmer
- The second uses Wordnet and is included in JWI

The implementation for the Porter stemmer can be found on Martin Porter's website [8]. The JWI implementation used in the second algorithm is an hybrid approach, as described in the chapter about *Information Retrieval*.

[8]The Porter Stemming Algorithm, `http://tartarus.org/~martin/PorterStemmer/`

7.3.4 Detecting synonyms

In the same way as the inflection reducer, the synonyms detector was implemented using a wrapper around a word parser.

The default implementation looks up for synonyms in Wordnet, and when synonyms are found, the basic form for the term is returned.

Synonyms are looked up in Wordnet database using the JAWS (Java API for Wordnet Search) library.

7.3.5 Scoring algorithms

Two implementations of scoring algorithms are present in the code base: F_1 and Average precision.

Both algorithms were implemented according to the formulas described in the chapter *Scoring titles*.

7.4 Testing the prototype

The prototype was tested against all the training documents used in the experimentation phase of this project. The aim was to check the behaviour of the prototype in well known conditions.

Results obtained using a F_1 measure were similar to the results obtained during the experimentations. However, results obtained using the Average Precision metric are much lower. This is due to the fact that the Average Precision metric is harsher and more strict than the F_1 measure.

7.4.1 Tests on incorrect titles

The prototype was also tested on unknown documents, known to be incorrectly entitled, in order to check the ability of the prototype to detect incorrect titles.

Poor quality titles were forged with this aim in mind, containing titles completely

unrelated to the contents of pages. On hand-crafted documents, the prototype performed well and gave bad scores to all the tested pages.

8 Evaluation and future work

8.1 Critique of the solution

The final solution to this project is a prototype giving scores to titles depending on the pertinence of their titles towards the contents of the page.

While scores given are influenced by the quality of the page, the resulting solution is not 100% reliable and they are many ways of influencing the results. This implies that correctly entitled pages may have low scores, and incorrectly entitled pages may have high scores. There are many ways of influencing and tricking the prototype.

This comes mostly from the complexity of the human language and all its subtleties. Moreover, most of the time the processing of the human language rely on knowledge that is required to understand the meaning of a text or title.

On average, the scores given by the prototype are very low. The metric to score titles is very harsh and it becomes very difficult to have high scores. The average score for a correctly entitled website (as an example, CNN.com) is 28%.

However, variations in the score appear depending on the quality of the title for a given web page.

Even if results are not perfect, they are usable and experimentations using this prototype as a scoring system could help determining threshold values delimiting titles considered as pertinent or non-pertinent.

Moreover, this prototype could be largely improved to augment the reliability of the Information Retrieval process used in this project. A manual analysis of the results obtained by the Information Retrieval process shows that concepts disambiguation could be improved, as well as the extraction of generic ideas from terms.

8.2 Areas of further development

The results obtained in this project can be improved in a number of ways. This section tries to summarise the techniques that could be considered to improve the accuracy of the project further, and try to obtain better results.

8.2.1 Languages specificities

Some languages have some specificities that were not encountered in the scope of this project, this project focusing only on two languages (English and French).

Some specificities might introduce new problems and challenges in the Information Retrieval process, and need to be considered when applying the techniques employed in this project to new languages.

Split named entities

In some languages, named entities can be separated by other words. This is the case with German, where verbs composed of multiple verbs are usually located partially at the beginning and partially at the end of the same sentence.

For example, in the German sentence "*Mach die tür zu*" (meaning "Close the door"), the verb "*zu machen*" is located in the first and last terms of the sentence.

Such entities are harder to detect, and an analysis requiring a deep knowledge of the language constructs might be needed.

Words concatenation

Another problem introduced by the German language is the flexibility of the language, allowing multiple words to be concatenated to form longer words, combining the meanings of a word to create a group of ideas.

A well known example is the word "*Donaudampfschiffahrtsgesellschaftskapitän*", often considered as one of the longest German word, that means "Danube steamship company captain".

In this case, all the German words for “Danube”, “Steamship”, “Company” or “Captain” might be relevant without being explicitely present in the contents. This might happen if one or more of these words are only present in the concatenated form.

A potential solution would be to use a stemmer able to split such words in multiple fragments, allowing a deeper analysis of this kind of concatenated words.

Non-Latin alphabets

French and English, two languages experimented in this project, are both languages using latin alphabets. Moving to a new language using a different set of characters might cause some trouble in words detection.

This project assumes that words are a sequence of letters separated by one or more non-alphabetic characters. This might not be the case in some languages using Kanji (Japanese)

8.2.2 Improve the content of synonymy databases

Synonymy databases used in the context of this project try to cover the basic words of a language, usually present in a dictionary.

However, synonymy can be improved to detect similar concepts and ideas beyond the simple meaning of words, and detect similar concepts such as similar geographical regions, or people belonging to the same organisational unit.

Geographical concepts

Databases of concepts exist to extract generic concept from narrower concepts. For example, “Information Retrieval” and “Information Extraction” are concepts related to “Natural Language Processing” and have to be considered as such when detecting synonymy between concepts.

The same example of concepts can be found in geographics. For example, “Tripoli” and “Benghazi” are narrower concepts for the general concept of “Libya”. Such databases of geographical concepts have to be used to be able to extract general geographical

concepts.

Important databases of such geographical data exist on the Web and could be adapted to be used in the context of this project. Such a database is Geonames [1], which contains RDF triples to represent semantically geographical locations worldwide [BHIBL08].

Organisational units

In the same way that geographical areas need to be represented in the concepts database, organisational units can be added to the databases.

Organisational units can easily be referenced and evoked by using names of members of the organisation, name of employees, or even by their geographical location.

For example, if the names "Bill Gates" and "Steve Ballmer" are present in the text of a web page, it is likely that the page is talking about the concept of "Microsoft". The same can be said if the page contains "Redmond", which is the city where Microsoft headquarters are located.

Databases of such concepts exist on the Internet, but few are really exhaustive. More over, such databases usually contains lots of informations and might need to be cleaned to fit the needs of a Natural Language Processing application.

A well known database of such information is the DBpedia project [2], which aims at extracting information found in the Wikipedia project in a semantic database [ABK+07].

Such a database could be easily reused and might offer valuable information in the context of analysing semantic concepts in the text of the web page.

[1] GeoNames, `http://www.geonames.org/`
[2] DBpedia, `http://dbpedia.org/`

9 Conclusion

In conclusion, different Natural Language Processing algorithms and techniques were experimented in order to determine which ones could be applied in the context of scoring titles for general web pages.

Experimental results showed that algorithms present similar results on different corpora and languages. It seems neither the corpora nor the language influence the accuracy of statistical algorithms.

However, the semantical analysis uses dictionaries and the quality of the whole Information Retrieval process depend on the quality of the word databases used in the project. A poor database can greatly influence results and decrease their quality.

Moreover, the area of Natural Language Processing is still a complex area in which no perfect results can be obtained. At the same time, even if not perfect, acceptable results can be obtained using the correct techniques to extract information from texts.

Finally, the results can be greatly improved by augmenting the content of the synonymy databases and including general concepts, other than simple words synonymy, inside the databases. This could make the algorithms aware of the context in which they operate and make the Information Retrieval process more accurate.

9.1 Ethical and legal issues

During the development of this project, some ethical and legal issues were overlooked due to the academic nature of this disseration.

9.1.1 Copyrights on training material

The training evaluation material was extracted in a academic research context, and none of the extracted data was republished under any form whatsoever.

However, web sites can prevent the extraction of their web pages for this kind of usage in their Terms of Usage, and the terms of usage of each and every web site have to be carefully read in order to prevent any Intellectual Property infringement.

As an example, the Terms of Service of CNN.com [1] states that the contents provided on the CNN web site is subject to the copyright laws in force in the United Stated. It also states that you "may download copyrighted material for your personal use only". Downloading and reusing the content of CNN to rank the title of pages issued on other web sites could be considered a copyright infringement, unless it is a strictly personal usage.

The Terms of Usage of a web site can also impose limits on the way their data is extracted and analysed. For example, Wikipedia prevents the usage of web site extractors on their web sites. Instead, they recommend to download the full HTML archives provided.

9.1.2 Copyrights on words databases

In the same way that web sites are subject to copyrights laws and Intellectual Property Rights, word databases can be subject to licensing terms and have restrictions in their terms of usage.

This is the case for synonyms databases, but also stop lists, concepts databases and inflections databases.

In the context of this project, Wordnet was used. The license, available on Wordnet web site [2], states that Wordnet can be freely reused in any academic or, more generally, any non-commercial project.

WOLF, the Wordnet compatible database of French words, uses a CeCILL-C license,

[1] CNN.com – Terms, `http://edition.cnn.com/interactive_legal.html`
[2] Wordnet – License and commercial use, `http://wordnet.princeton.edu/`

that allows it to be reused freely as part of a commercial project (Derivative Software clause). However, all the modifications done to the database have to be redistributed under a similar license.

When using such databases in a project, it is important to consider carefully the licenses of words databases, and to ponder the implications on the project.

9.1.3 Software patents on techniques and algorithms

Some of the techniques and algorithms discussed in this dissertation can be the subject of one or more software patents that could prevent their usage in a project, or make this usage subject to a patent fee.

When using algorithms, checking patents databases is important to avoid any Intellectual Property Infringement.

References

[ABK+07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer Berlin / Heidelberg, 2007.

[AOG98] Chinatsu Aone, Mary Ellen Okurowski, and James Gorlinsky. Trainable, scalable summarization using robust nlp and machine learning. In *Proceedings of the 17th international conference on Computational linguistics - Volume 1*, COLING '98, pages 62–66, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[BHIBL08] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web (ldow2008). In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 1265–1266, New York, NY, USA, 2008. ACM.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[Bur05] David Burdon. *The basics of Search Engine Optimisation.* Simply Clicks, 2005.

[DG06] Robert A. Day and Barbara Gastel. *How to Write and Publish a Scientific Paper – Sixth edition.* Cambridge University Press, 2006.

[EM03] Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 459–

460, New York, NY, USA, 2003. ACM.

[Fox89] Christopher Fox. A stop list for general text. *SIGIR Forum*, 24:19–21, September 1989.

[GKA05] Clifford W. Gay, Mehmet Kayaalp, and Alan R. Aronson. Semi-automatic indexing of full text biomedical articles. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, pages 271–275, 2005.

[JH01a] Rong Jin and Alexander G. Hauptmann. Automatic title generation for spoken broadcast news. In *Proceedings of the first international conference on Human language technology research*, pages 1–3. Association for Computational Linguistics, 2001.

[JH01b] Rong Jin and Alexander G. Hauptmann. Learning to select good title words: An new approach based on reverse information retrieval. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 242–249, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[JH02] Rong Jin and Alexander G. Hauptmann. A new probabilistic model for title generation. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[KH00] Paul E. Kennedy and Alexander G. Hauptmann. Automatic title generation for em. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 230–231, New York, NY, USA, 2000. ACM.

[KSN10] Martin Klein, Jeffery L. Shipman, and Michael L. Nelson. Is this a good title? *CoRR*, abs/1004.2719, 2010.

[McB94] Oliver A. McBryan. Genvl and wwww: Tools for taming the web. In *In Proceedings of the First International World Wide Web Conference*, pages 79–90, 1994.

[Mil95] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November 1995.

[NSB+09] Emmanuel Navarro, Franck Sajous, Gau Bruno, Laurent Prévot, Hsieh

ShuKai, Kuo Tzu-Yi, Pierre Magistry, and Huang Chu-Ren. Wiktionary and nlp: improving synonymy networks. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, People's Web '09, pages 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[PAD98] Glen Pringle, Lloyd Allison, and David L. Dowe. What is a tall poppy among web pages? *Computer Networks and ISDN Systems*, 30(1-7):369 – 377, 1998. Proceedings of the Seventh International World Wide Web Conference.

[PHKJ01] Ari Pirkola, Turid Hedlund, Heikki Keskustalo, and Kalervo Jrvelin. Dictionary-based cross-language information retrieval: Problems, methods, and research findings. *Information Retrieval*, 4:209–230, 2001. 10.1023/A:1011994105352.

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program*, 3(14):130–137, October 1980.

[Rij79] C. J. Van Rijesbergen. *Information Retrieval*. London: Butterworths, 1979.

[SF08] Benoît Sagot and Darja Fišer. Building a Free French Wordnet from Multilingual Resources. In *OntoLex 2008*, Marrackech, Morocco, 2008.

[SLWPC00] Tomek Strzalkowski, Fang Lin, Jin Wang, and Jose Perez-Carballo. Evaluating natural language processing techniques in information retrieval. 2000.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[TLCL06] Yuen-Hsien Tseng, Chi-Jen Lin, Hsiu-Han Chen, and Yu-I Lin. Toward generic title generation for clustered documents. In Hwee Ng, Mun-Kew Leong, Min-Yen Kan, and Donghong Ji, editors, *Information Retrieval Technology*, volume 4182 of *Lecture Notes in Computer Science*, pages 145–157. Springer Berlin / Heidelberg, 2006.

[TS06] Andrew Turpin and Falk Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information re-*

trieval, SIGIR '06, pages 11–18, New York, NY, USA, 2006. ACM.

[WBFL09] Timothy Weale, Chris Brew, and Eric Fosler-Lussier. Using the wiktionary graph structure for synonym detection. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, People's Web '09, pages 28–31, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[WM99] Michael J. Witbrock and Vibhu O. Mittal. Ultra-summarization: A statistical approach to generating highly condensed non-extractive summaries. In *In SIGIR99*, pages 315–316, 1999.

[WS92] W. John Wilbur and Karl Sirotkin. The automatic identification of stop words. *J. Inf. Sci.*, 18:45–55, January 1992.

[XHX+07] Yewei Xue, Yunhua Hu, Guomao Xin, Ruihua Song, Shuming Shi, Yunbo Cao, Chin-Yew Lin, and Hang Li. Web page title extraction and its application. *Information Processing & Management*, 43(5):1332 – 1347, 2007. Patent Processing.

[ZLD09] Chengling Zhao, Jiaojiao Lu, and Fengfeng Duan. Application and research of seo in the development of web2.0 site. In *Proceedings of the 2009 Second International Symposium on Knowledge Acquisition and Modeling - Volume 01*, KAM '09, pages 236–238, Washington, DC, USA, 2009. IEEE Computer Society.

Bibliography

[1] G. Chowdhury. *Introduction to modern information retrieval.* London: Facet, 2010.

[2] Jim Cowie and Wendy Lehnert. Information extraction. *Commun. ACM*, 39:80–91, January 1996.

Appendix 1: Glossary of technical terms

This appendix contains a list of technical terms used in this dissertation.

hypernym Term that designates a general concept broader than the considered term.

For example, "communication" is the *hypernym* of "message", the concept of messaging being a form of communication.

hyponym Term whose meaning is narrower than another term. Opposite of *hypernym*.

For example, "message" is an *hyponym* of "communication".

inflection Modification of a word to express a grammatical context.

The *inflection* of a word can be a conjugation, a plural or gender declension, or any modification of a word induced by its grammatical context.

Information Retrieval Area of *Natural Language Processing* consisting of searching relevant information from documents.

Information retrieval is often used with the aim of searching documents, extracting meta-data or mining documents.

lemma Most basic form of a word, without any *inflection*.

The *lemma* of a word represents its simplest form and is the form usually found in dictionaries.

lemmatisation The process of removing all suffixes of a term to retrieve its *lemma*.

In some cases, the *lemmatisation* process can be more aggressive and can modify the word further than a simple suffix removal.

precision In *Information Retrieval*, proportion of retrieved elements that are relevant to the original query.

recall In *Information Retrieval*, proportion of relevant elements that were retrieved during the process.

Search Engine Optimisation Set of techniques and good practices that improves web sites indexing in search engines.

The process of *Search Engine Optimisation* focuses on helping search engines index a web site, with the main aim being to improve the ranking of a web site in search results pages.

SEO See *Search Engine Optimisation*.

stemmer Algorithm executing the process of stemmisation.

A *stemmer* is generally used to denote an algorithm that extracts the root of a word by removing suffixes on the word.

stemmisation See, *lemmatisation*

The term *stemmisation* is usually employed to denote the process of *lemmatisation* using an algorithmic *stemmer*.

stop list Database of *stop words*

stop word Term appearing often in a language and carrying little or no meaning.

Examples of *stop words* in English include "of", "the" or "and".

Term Frequency – Inverse Document Frequency Term ranking formula focusing on giving scores to terms depending on their importance in a text.

TF-IDF gives a high importance to words appearing often in the considered text and are rarer in others documents.

TF-IDF See *Term Frequency – Inverse Document Frequency*

Appendix 2: Tables of experimental results

Statistical experiments

The F_1 results for statistical algorithms are shown in table 9.1

Semantic experiments

Stemmisation

The complete experimental results, after stemmisation, are shown in table 9.2.

Synonyms detection

The complete results for the experiments related to the analysis of synonyms disambiguation are shown in table 9.3.

Stop words removal

The table of complete experimental results after the removal of all the stop words from the list of extracted terms is shown in table 9.4.

Algorithm	CNN	Le Monde	Wikipedia EN	Wikipedia FR	Average
TF-IDF	0.19346	0.14568	0.16382	0.18669	0.17129
Naïve Bayes classifier	0.30741	0.30593	0.54493	0.29619	0.33809

Table 9.1: Average F_1 for statistical algorithms

Algorithm	CNN	Le Monde	Wikipedia EN	Wikipedia FR	Average
TF-IDF	0.20181	0.14609	0.16318	0.18563	0.17299
Naïve Bayes classifier	0.30094	0.31950	0.54261	0.29607	0.34072

Table 9.2: Average F_1 for statistical algorithms, after stemmisation

Algorithm	CNN	Le Monde	Wikipedia EN	Wikipedia FR	Average
TF-IDF	0.20986	0.14673	0.16638	0.18549	0.17552
Naïve Bayes classifier	0.30070	0.31951	0.54454	0.29649	0.34107

Table 9.3: Average F_1 for statistical algorithms, after synonyms detection

Algorithm	CNN	Le Monde	Wikipedia EN	Wikipedia FR	Average
TF-IDF	0.22265	0.14712	0.17891	0.18558	0.18051
Naïve Bayes classifier	0.28679	0.32114	0.45817	0.29657	0.32586

Table 9.4: Average F_1 for statistical algorithms, after synonyms detection

Appendix 3: Listings of algorithms implementations

Listing 9.1: Full implementation of TF-IDF algorithm

```
public class TFIDFWordExtractor implements WordExtractor {

    /*
     * Map storing the number of documentsin which each term appears
     */
    private Map<String, Integer> termsCount =
                            new HashMap<String, Integer>();

    /*
     * Number of documents processed during the training
     */
    private long documentsCount = 0;

    public void train(Document doc) {
        Set<Word> words = getWords(doc);

        for (Word word : words) {
            Integer count = termsCount.get(word.text);
            count = count == null ? 0 : count;
            termsCount.put(word.text, count + 1);
        }

        documentsCount++;
```

```
}

public List<String> extract(Document doc) {
    Set<Word> words = getWords(doc);

    int wordsCount = 0;
    for (Word word : words) {
        wordsCount += word.count;
    }

    Comparator<Word> comparator = new TFIDFComparator(wordsCount);
    SortedSet<Word> sortedWords = new TreeSet<Word>(comparator);
    sortedWords.addAll(words);

    List<String> returns = new ArrayList<String>();
    for (Word result : sortedWords) {
        returns.add(result.text);
    }

    return returns;
}

protected Set<Word> getWords(Document doc) {
    String text = doc.body().text();
    return getWords(text);
}

protected Set<Word> getWords(String text) {
    Map<String,Word> words = new HashMap<String,Word>();

    WordParser parser = new SynonymsParser(
            new WordnetSynonymsExtractor(),
            new StemmingParser(new NonAlphaWordParser(text),
```

```
                    new WordnetDictionaryStemmer()));
        String next;
        while ((next = parser.next()) != null) {
            String word = next.toLowerCase();
            if (!words.containsKey(word)) {
                words.put(word, new Word(word));
            }
            words.get(word).count++;
        }

        return new HashSet<Word>(words.values());
    }

    public void save(OutputStream out) throws IOException {
        ObjectOutputStream dOut = new ObjectOutputStream(out);
        dOut.writeObject(termsCount);
        dOut.writeLong(documentsCount);
        dOut.close();
    }

    public void load(InputStream in) throws IOException {
        ObjectInputStream dIn = new ObjectInputStream(in);
        try {
            termsCount = (Map<String, Integer>) dIn.readObject();
            documentsCount = dIn.readLong();
        } catch (ClassNotFoundException ex) {
            throw new RuntimeException(ex);
        }
    }

    protected static class Word {

        private String text;
```

```
    private int count = 0;

    public Word(String text) {
        this.text = text;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Word other = (Word) obj;
        if ((this.text == null) ? (other.text != null)
                    : !this.text.equals(other.text)) {
            return false;
        }
        return true;
    }

    @Override
    public int hashCode() {
        int hash = 7;
        hash = 47 * hash + (this.text != null
                    ? this.text.hashCode() : 0);
        return hash;
    }

    @Override
    public String toString() {
```

```
        return "Word{" + text + '}';
    }

}

protected class TFIDFComparator implements Comparator<Word> {

    private int wordsCount;

    public TFIDFComparator(int wordsCount) {
        this.wordsCount = wordsCount;
    }

    public int compare(Word o1, Word o2) {
        double tfidf1 = tfidf(o1);
        double tfidf2 = tfidf(o2);

        if (tfidf1 > tfidf2) {
            return -1;
        }
        if (tfidf1 < tfidf2) {
            return 1;
        }
        return 0;
    }

    protected double tfidf(Word w) {
        double tf = w.count / (double) wordsCount;
        double idf = Math.log(documentsCount
           / termsCount.get(w.text).doubleValue());
        return tf * idf;
    }
```

```
    }

}
```

Listing 9.2: Full implementation of Naïve Bayes classifier

```
public class BayesClassifierWordExtractor implements WordExtractor {

    private Map<String, Integer> titleCounts
                            = new HashMap<String, Integer>();

    private Map<String, Integer> documentCounts
                            = new HashMap<String, Integer>();

    public void train(Document doc) {
        for (String word : getWordsNoSynonyms(doc.title())) {
            if (titleCounts.containsKey(word)) {
                titleCounts.put(word, titleCounts.get(word) + 1);
            } else {
                titleCounts.put(word, 1);
            }
        }

        for (String word : getWords(doc.body().text())) {
            if (documentCounts.containsKey(word)) {
                documentCounts.put(word, documentCounts.get(word) + 1);
            } else {
                documentCounts.put(word, 1);
            }
        }
    }

    public List<String> extract(Document doc) {
        Set<String> words = getWords(doc.body().text());
```

```
        List<String> sortedWords = new ArrayList<String>(words);
        Collections.sort(sortedWords, new BayesClassifierComparator());

        return sortedWords;
    }

    public void save(OutputStream out) throws IOException {
        ObjectOutputStream oOut = new ObjectOutputStream(out);
        oOut.writeObject(titleCounts);
        oOut.writeObject(documentCounts);
        oOut.close();
    }

    public void load(InputStream in) throws IOException {
        ObjectInputStream oIn = new ObjectInputStream(in);
        try {
            titleCounts = (Map<String, Integer>) oIn.readObject();
            documentCounts = (Map<String, Integer>) oIn.readObject();
        } catch (ClassNotFoundException ex) {
            throw new RuntimeException(ex);
        }
    }

    protected Set<String> getWords(String text) {
        Set<String> words = new HashSet<String>();

        WordParser parser = new SynonymsParser(
                new WordnetSynonymsExtractor(),
                new StemmingParser(new NonAlphaWordParser(text),
                        new WordnetDictionaryStemmer()));
        String next;
        while ((next = parser.next()) != null) {
            words.add(next.toLowerCase());
```

```
        }

        return words;
    }

    protected Set<String> getWordsNoSynonyms(String text) {
        Set<String> words = new HashSet<String>();

        WordParser parser = new StemmingParser(
                new NonAlphaWordParser(text),
                new WordnetDictionaryStemmer());

        String next;
        while ((next = parser.next()) != null) {
            words.add(next.toLowerCase());
        }

        return words;
    }

    protected class BayesClassifierComparator
                        implements Comparator<String> {

        public int compare(String s1, String s2) {
            double bayes1 = bayes(s1);
            double bayes2 = bayes(s2);

            if (bayes1 > bayes2) {
                return -1;
            }
            if (bayes1 < bayes2) {
                return 1;
            }
```

```
            return 0;
        }

        protected double bayes(String word) {
            Integer titleCount = titleCounts.get(word);
            titleCount = titleCount == null ? 0 : titleCount;

            double documentCount = documentCounts.get(word).doubleValue();

            return titleCount / documentCount;
        }

    }

}
```