Automatic Construction of a Multi-Document Summarization Corpus

Master's Thesis

Ferdinand Armel Kameni Leukam
Knowledge Engineering Group
Technische Universität Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Automatic Construction of a Multi-Document Summarization Corpus

Vorgelegte Master-Thesis von Ferdinand Armel Kameni Leukam

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Markus Zopf

Tag der Einreichung:

Thesis Statement pursuant to § 22 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, den 16. May 2017

Ferdinand Kameni

Abstract

The increasing rate of information on information sites, digital books, etc., makes it difficult for readers to obtain the essential information they need in a short time. Not only is the quantity of information huge, but also the heterogeneity of the data makes it difficult to summarize them automatically because of the styles, the text structures, the level of language used and the forms of presenting the information vary from one domain and genre of text to another. That is why the automatic summary of multiple heterogeneous sources is a very interesting field of research. The *text summarization* (Nenkova & McKeown, 2011) is the process of shortening a text or a group of documents (multi-document summarization) written about the same topic, to a condensed version of a textual document In general, there are two forms of summarization: the *abstractive summarization*, which consists in expressing the ideas in a source documents by reformulating them and *the extractive summarization*, which builds the summary from some extracted sentences of the source document. A summary is useful in order to get relevant, important information about a topic without reading it completely.

Most of the existing corpus (group of files stored in electronic form and intended for a specific purpose) for multi-document summarization such as MultiLingb (Habernal, Zayed, & Gurevych, 2016) or issue from e.g. the Document Understanding Conference (DUC)(Over, Dang, & Harman, 2007), are mainly focused on homogeneous data, more prevalently in newspapers. These corpora are built manually and contain too few topics. Therefore, they are too small for a reliable analysis using machine learning. (Zopf, Peyrard, & Eckle-kohler, 2016) Propose a heterogeneous corpus, built manually and comprising 10 different genres of documents. They reverse the summarization process by starting from a summary text and create their corpus in three steps: from a summary, some important fragments of information are first extracted, then they are used to download web pages related to the topic, and finally, these web pages are classified in the 10 different genres they proposed. The goal of this thesis is to automate this process and evaluate its performance.

Two approaches, namely the part of speech and the named entities have been used to extract the information nuggets. By comparing them to the manually extracted list, the named entities based approach provided better results when considering their F-Measure values (64.76% > 42.59%). Subsequently, topic-related web pages to the information nuggets were downloaded using the Bing Web Search API. Several classifiers were tested on the training data and SVM was more efficient than the others (accuracy = 61.59 %). The causes of this average accuracy have also been explored. The training data was generated from the corpus on the one hand using the StringToWordVector filter of the WEKA tool and on the other hand by building some features related to the web pages. The classification of the newly downloaded web pages reveals that they are made up of 90% of three document genres, and the other seven genres have been represented at 10%. On average, 12 files were assigned to each summary file and split on average into four documents genres.

The automation of this process makes it possible to create the largest heterogeneous corpus for multi-document summarization, and the possibility of using the machine learning to evaluate the training set reliably. The tools used allow to extend this solution in addition to the English language, to three other languages Chinese, German, and Spanish.

Contents

1 Introduction

The *automatic text summarization* consists of reducing a text to the essentials by using a computer. Because of the information being overloaded on the net, search engines can return hundreds of results for a query. Having a summary of these results would help the surfer to find the information he needs quicker. Another scenario is, for example, the summary of a book. In this case, it helps the reader to decide whether to continue reading, to buy it or not. Researchers invest much time in learning computers to accurately produce the summary of a given text or set of documents. When only a document has to be summarized, this is called a *single document summarization* (abstract, the introduction of an article, etc.). A summarizer can also condense the content of many documents about the same topic. If the set of documents belong to the same genre (e.g. news), then this process is called *multi-documents summarization (MDS)*, and if the input documents belong to more than one genre (news, blog, etc.), this is called *heterogeneous multi-document summarization (hMDS)*.

There are two forms of summarization: the abstractive summarization and the extractive summarization. A human being often does the abstractive summarization; it consists of reading a text and resuming its contents with others words. In contrary, the extractive summarization is the fact to fragment input texts into sentences; important sentences are selected and reorganized in a coherent order. Finally, they are reduced to build the summary (Figure 1-1). Most research in automatic text summarization is based on the extractive summarization (Neto, Freitas, & Kaestner, 2002).

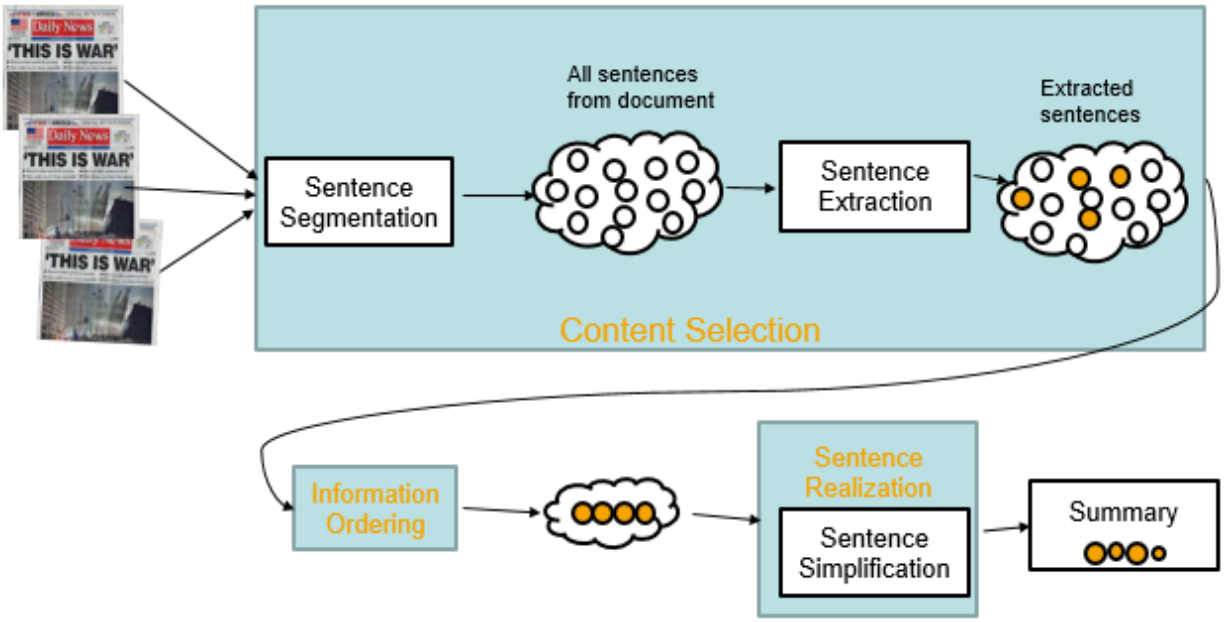


Figure 1-1: Extractive summarization approach (Source: Jurafski & Martin)

The set of documents to summarize is often taken from a corpus. A corpus (*plural corpora*) is an assemblage of documents in the electronic form for a defined purpose.

1.1 Motivation

All the existing corpora for multi-document summarization that exist so far are very small and have been manually created. These corpora are created by asking the annotators to search for and record documents that encounter certain criteria. In general, they contain less than a hundred of topics and sometimes very few documents are assigned to each topic. This small size of corpora does not allow to use the machine learning algorithms to train them efficiently on summarization tasks. The summarization is then done manually, mostly using the approaches presented in the *Document*

Understanding Conference (DUC) or the *Text Analytic Conference (TAC)* (these approaches will be described later in the related work).

If the number of documents to be summarized becomes very large, it becomes painful, or impossible for the human being to produce a good summary. A solution is to make use of *crowdsourcing* (the task is done by a crowd of people). For example, documents can be divided into sets. Several annotators are invited to summarize each set. Subsequently, the best summary of each set is chosen. These chosen summaries can be assembled again and resubmitted for a second summarization task, and so one until the desired length of the summary is reached. (Lloret, Plaza, & Aker, 2013) show that this method leads to a poor result. Then machine learning becomes very necessary to summarize them automatically. With a very small corpus, machine learning cannot produce satisfying results. Thus, the larger the corpus, the better the machine learning algorithms are trained to produce reliable results. (Neto et al., 2002) gives some advantages of an automatic summarization, for example the size of the abstract can be monitored, and it is possible to establish a relation between an element in a summary and its position in the original text.

(Zopf et al., 2016) proposes a corpus containing more than 100 subjects that were manually created, following the defined steps they established. The goal of this thesis is to study the automation of their approach. In case of success, the corpus will be extended automatically to thousands of subjects, which makes it interesting for a reliable summarization using the machine learning.

1.2 Contribution

(Zopf et al., 2016) describes a process to generate a homogeneous corpus for multi-document summarization automatically. This process consists of three main steps: *information nugget extraction* (fragment of information that offers useful information about a topic), *web page retrieval, and web page classification* into 10 document genres. In this thesis, it will be shown that part of speech (nature of words) and named entities (e.g. Person, location, time, etc.) can be used to find such information nuggets. In addition to this, experiments have revealed that SVM is a suitable classifier for the web page classification tasks and provides better results when the training datasets are generated from HTML files instead of text files. It is also noticed that a document could belong to more than one genre, making the learning difficult to the classifiers. For this reason, adding some features attributes such as text features, HTML structures, URL feature, and if possible also include the analysis of neighbor web page, will improve the accuracy. Finally, it is remarked, that some document genres (from the given 10 genres) appear very rarely in the retrieved web page using a search engine (Bing). A corpus for heterogeneous multi-document summarization has been produced automatically and can be used as silver standard in the summarization tasks.

1.3 Structure

The general topic of this thesis was introduced in Section 1. In Section 1.1, the motivation of this work is explained, and the contribution in section 1.2 will be given. In this section, an overview of the overall structure and a short insight in each of the following section is given. The rest of this document is organized as followed: the related works are presented in section 2. In section 2.1, there is a list of some related works referring to the corpora for text summarization. They are separated into related works for homogeneous corpora (section 2.1.1) and heterogeneous corpora (section 2.1.2). In section 2.2 some linked works to web page classification are given. The corpus that is used in this thesis is presented in section 3. The automation of this corpus construction will be in three steps:

The first step is the extraction of information nuggets. Section 4 highlights the methods used for this task. First, in section 4.1, some reasons to choose a Wikipedia article as the start point of this process are given. In section 4.2 the different ways used to achieve this goal will be presented. The first method includes using the part of speech (section 4.2.1). The obtained list can be refined using the ClausIE tool

that is introduced in section 4.2.2. This tool is useful if it is combined with a co-reference resolution tool. That is argued in section 4.2.3. Another process of identifying the information nuggets are by using the named entities. This process described in section 4.2.4. Finally, it will be tested if merging the two approaches can be powerful in section 4.2.5.

The second step of this work is web pages retrieval and download (section 5).

The last step of this process is web page classification. This is discussed in section 6. It is spoken about the restructuration of the corpus in section 6.1, to enable WEKA to generate the datasets automatically. The experiments that led us to the choice of the SVM classifier are introduced in section 6.2. In the following section (6.3) is given more information about it and also some achieved results in section 6.4. Some possible factors that led to the average accuracy observed are explained in section 6.4.1. After this, another approach of classification by building the attributes based on the HTML, text, and URL features is presented in section 6.4.2. Then in section 6.4.3, the chosen classifier is tested on others datasets to check its parameters configurations. After training the SVM classifier, the classification of newly downloaded web pages in highlighted in section 6.5.

Some examples of the classified web page are shown in section 7. Section 8 explain how the developed tool can be tested. The conclusion of this work is presented in section 9 with some fields for feature works.

2 Related Work

Experts may sometimes do the summarization task, but the work becomes boring and almost unfeasible as the number of documents to be summarized increases. This is why systems for automatic summarization have been developed. These systems can only be evaluated on gold standards which are represented by manually created corpora (All the corpora found for training datasets, had been manually generated, for this reason, there are not silver standards that can also be used. It is spoken about the silver standard when the training dataset is generated automatically). Most corpora that are used for multi-document summarization (MDS) tasks are homogeneous, but there also exist some heterogeneous corpora. In the following, the available corpora found for MDS are presented.

2.1 Corpora for Text Summarization

The corpus that was found are usually very small in size and cannot provide reliable results using machine learning (ML). The larger the corpus, the better the ML classifiers can be trained for the summarization tasks.(Neto et al., 2002) describes how machine learning is used in the summarization task. It exploits information like the sentence length, sentence position, similarity to title, similarity to keywords, sentence-to-sentence cohesion, sentence-to-centroid cohesion, depth in the tree, referring position in a given level of the tree, indicator of main concepts, occurrence of proper names, occurrence of anaphors, occurrence of non-essential information, and so one. However, the accuracy of the classifiers is improved with the size of the corpus. There exist some corpora for homogeneous and heterogeneous multi-documents summarization.

2.1.1 Homogeneous Corpora

Corpora issued from *DUC 2001-2003* (Over & James, 2001): DUC is the abbreviation of Document Understanding Conference. These corpora are available from September 2001. It has been designed for multi-summarization of newswire in the English language, and the provided summaries had fixed lengths of 50, 100, 200, and 400 words. This corpus is built with 60 sets of 10 documents. One-half is used for training and the other half for testing. In case of single-document summarization (SDS), 100-words softcopy summary is generated for each document and constitutes the summary (take the first 100 words in the document). For MDS, a 400-words softcopy is done for each sets a set of 10 documents (take the first 400, 200, 100, or 50 in the most recent document). A 400-words extracted summary can be then reformulated to reduce its size by half up to the desired size (200, 100, or 50). (Benikova, Mieskes, Meyer, & Gurevych, 2016) mention in their paper that another corpus which is similar to the precedent versions has been created in DUC 2004; in addition to the English language in DUC (2001) the Arabic language was added. This corpus has 50 topics, and the summarization is done by reducing the extracted sentence into a 10-100 words summary. Further, from *DUC (2005-2007)*, a new corpus only available for the English language has been created. It contains 50 topics and more than 25 documents per topic. The summary consists of a 250-words. The summaries in DUC are performed by using the abstractive summarization form.

Corpora issued from *TAC 2008- 201:* The TAC (Text Analysis Conference) summarization proceeds with the efforts made at the DUC 2007. The summarization task consists of writing a short 100-word summary average of a set of newswire articles assuming that the user has already read a given set of articles previously. 10 documents are assigned to each of its 44 topics all written in the English language. Content and readability are the points taken into account in the evaluation of the summaries. Apart from newswire, TAC has also been evaluated on blogs. Participants are provided with a list of TAC QA Track and the text snippets output by QA systems. The summaries are made by answering the questions using the text snippets or associated documents. From TAC 2014, a corpus of scientific documents has been created containing 50 topics and 10 documents for each topic. The summarization task here includes producing a 250-words summary, using the same process as previously. In conclusion, TAC has been tested on news, blogs, and scientific genres. However, each corpus only contains documents in one of

these three genres. As DUC, TAC uses the abstractive form of summarization. The first two presented corpora are the most popular used in the summarization task.

MultiLing (2011; 2013)(Giannakopoulos et al., 2011): this corpus has been conceived to evaluate the summarization algorithms on a set of languages. Its corpus contains 10 topics and each of them is assigned 10 documents. A single representative abstract must be produced from a set of documents describing a sequence of events. The size of the final summary is between 240 and 250 words. A topic should only contain documents written in one language at the same time. The languages to be used will be Arabic, Czech, English, French, Hebrew, Hindi and Greek. In 2015, MultiLing proposed another corpus that supports 10 different languages: Arabic, Chinese, Czech, English, French, Greek, Hebrew, Hindi, Romanian and Spanish. There are 10 documents in each of the 15 available topics and also a summary of 240 - 250 words has to be produced.

(Loupy, Guégan, Ayache, Seng, & Moreno, 2010): this paper presents a Corpus for Sentence Compression and Multi-Document Summarization for the French language. There are 20 documents per topic for an amount of 20 topics. The abstractive summarization form is used to produce a 200-words summary in average from news articles.

(Hirao, Fukusima, Okumura, Nobata, & Nanba, 2004): the summary is built from news articles written in the Japanese language. The extractive form of summarization is used to construct a summary containing 5% - 10% of characters. The first step consists of extracting important sentences from a given set of documents. The redundant sentences are minimized, then the result is rewritten to reduce the size of the summary to the specified number of characters or less. There is neither information about the number of topics available in the corpus, nor the number of documents for each topic.
(Goldstein, Carbonell, Kantrowitz, Carbonell, & Kantrowitzt, 2000): their corpus contains at least 10 sentences per summary from news genre in the English language. There are 10 documents in each of 25 genres. In comparison to the precedents corpus, this one is generated using the extractive summarization form.

(Carenini, Ng, & Zhou, 2007): propose a corpus for summarizing Email conversations written in the English language with Clue Words. It contains 20 topics with at least four Email conversations in each. The human summarizers directly selected sentences for each given Email. The summary is built by considering about 30% of the original sentences. These summaries had to represent the major ideas of the original E-Mail so that there is no need to read the original Email anymore. The extractive summarization form is used here.

(Zechner, 2002): presents a corpus that can be used to summarize Open-Domain Multiparty Dialogues in the English language. The extractive form of summarization is used on four different kinds of dialogues: CallHome from the Linguistic Data Consortium (LDC) collection, NewsHour (from PBS's NewsHour television show with Jim Lehrer (recorded in 1998)), CrossFire (from CNN's CrossFire television show with Bill Press and Robert Novak (recorded in 1998)), and GroupMeetings (from recordings of project group meetings in the Interactive Systems Labs at Carnegie Mellon University). This corpus contains in all 23 topics. The number of documents per topic and the size of the summaries are not given in the paper.

2.1.2 Heterogeneous Corpora

(Nakano et al., 2010): The presented corpus uses the extractive form of summarization to produce an abstract of about 2500 characters. It contains 24 topics, and 352 documents are assigned to each topic. The summaries have been manually produced by students. There were six queries sentences as the guideline. For a good quality summary, three student hat to read each summary and validate if the queries have been good answered.

(Lloret et al., 2013): Of all the corpora cited so far. The one presented here is the largest. It is a heterogeneous corpus of 310 topics in English. Each topic is assigned 10 documents. The extractive summarization is used to produce a summary of 100 - 200 words. They focus on the tourist domain and use the image collection described in (Aker & Gaizauskas, 2006). The model summaries were created manually based on image descriptions taken from VirtualTourist and contain a minimum of 190 and a maximum of 210 words.

(Benikova et al., 2016): propose a heterogeneous corpus in the German language. It contains 10 topics with about 4-14 documents per topic. They use the extractive form of summarization to build a 500-words summary in average. Their MDS process consists of three steps: In the first step, the most important units are selected (so-called identification of nuggets). They characterized a nugget as it should be important in the context of the given topic and has to contain at least one verb and one corresponding noun and can maximally span over one sentence. Next, the selected nuggets are clustered into groups with similar content, and for each cluster, the best nugget is selected. Finally, during the formulation of summaries, the best nuggets are co-reference resolved, grammatical and sorted coherently.

(Zopf et al., 2016) proposed a heterogeneous corpus with more than 100 topics in the English language. This corpus is still growing since others topics are permanently added. There are about 10 - 20 HTML documents for each topic. For each HTML file, there are three others text files obtaining by extracting the content of HTML files manually and automatically. In this Thesis, the process of automatically generate a heterogeneous corpus for the multi-documents summarization task based on their approach is studied. More detail about this corpus is given in the corpus description chapter.

2.2 Web Pages Classification

Several techniques and approaches have been used to classify web pages into different genres. In many researches, the classifiers such as *Support Vector Machine, Naïve Bayes, nearest neighbor, neural networks, J48,* and so forth have been used to classify new unseen web pages. Some features as the URL, the web page structure and the web page content have often been used for the classification task.

(Dwivedi, 2016) evaluate the performance in term of precision of web page classification using the Naïve Bayes classifier of Weka. Their approach consists of downloading some web page randomly using the Google Search engine and classifying them between news pages and non-news pages. They classify web pages using their content, URL, and HTML structure.

(Gu et al., n.d.) used the SVM classification on their work. They used the extraction method based on DOM Tree (Document Object Model Tree), which allows extracting the attitude information and text of the tags. The Jsoup Parser has been used to clean the pages of noises such as advertisement before generating the dataset for SVM classifier.

(He & Li, 2016) also used SVM classifier based on the structure of web documents in their work. Among the full text, they also used the META information and the title in the header of the HTML web pages. Moreover, they also analyze the links and the contents of image tags.

(Wen, Fang, & Guan, 2008) investigated on automatic web pages classification. They reduce the noises in the dataset by removing the head of HTML web page, which often contains JavaScript, Cascading Style Sheets (but they conserved the file title). The attributes have been generated from the word, lemma, and tag to train some WEKA classifiers (SVM, NBK, C 4.5 and Naive Bayes).

(Vidulin, 2007) considers HTML features, URL features, and text-based features to classify new unseen web page. They used a corpus of 1539 manually labeled web pages to generate 502 features. These

features have been chosen by observing the corpus and by surveying some related literature. J48 and baggin are the two machine learning algorithms chosen for training and testing the precision of the classification.

(Joachims, 1998) explore the use of Support Vector Machine (SVM) and give some arguments why the SVM classifier is suitable for text categorization in comparison to the others classifiers.

(Kan & Thi, 2005) compare in their work the web page classification using the URL features. The sets of features considered are URL components, its length, orthographic features, sequential n-grams and precedence Bigram. They conclude that in certain scenarios, their method outperforms the state-of-the-art text-based and link-based methods.

(Qi & Davison, 2009) investigated on some useful web-specific features for classification. They first give some differences between web page classification and text classification. For example, there exist connections between a web page to another through the links. They define two types of features (on-page features and neighbor's features). The on-page features are found on a web page (textual content and tags), and neighbors features are web page linked to the current web page (parent, child, grandparent, grandchild, sibling, and spouse). They affirm that considering these relationships perform the accuracy of classifiers in comparison to the bag of words method.

(Riboni, 2002) classifies web page using the NaiveBayes classifier. They conduct their experiments on a corpus containing 8000 documents to 10 Yahoo! categories. They considered the plain text and the URL. Their experiments reveal that this combination could improve classification performances.

3 Corpus Description

A corpus (corpora in plural) is an assembly of a large number of files in electronic form meeting certain criteria and intended for a specific usage. Generally, corpora are used to verify a hypothesis or also to train computers to perform tasks like the text categorization and summarization. The bigger a corpus is, the better it produces reliable results while learning the data. In the summarization tasks, the files that have to be processed must relate to the same topic.
As seen in the related works, the corpora have always been manually created. After many researches, there is no system that is used to generate the training set automatically. (Zopf et al., 2016) propose a new approach to generating heterogeneous corpora for the multi-documents summarization task (Figure 3-1). This process consists of three steps: information nuggets extraction (fragments of text, which give important information about the topic), web page retrieval, and web page classification. The purpose of this thesis it to study how this process can be automated.

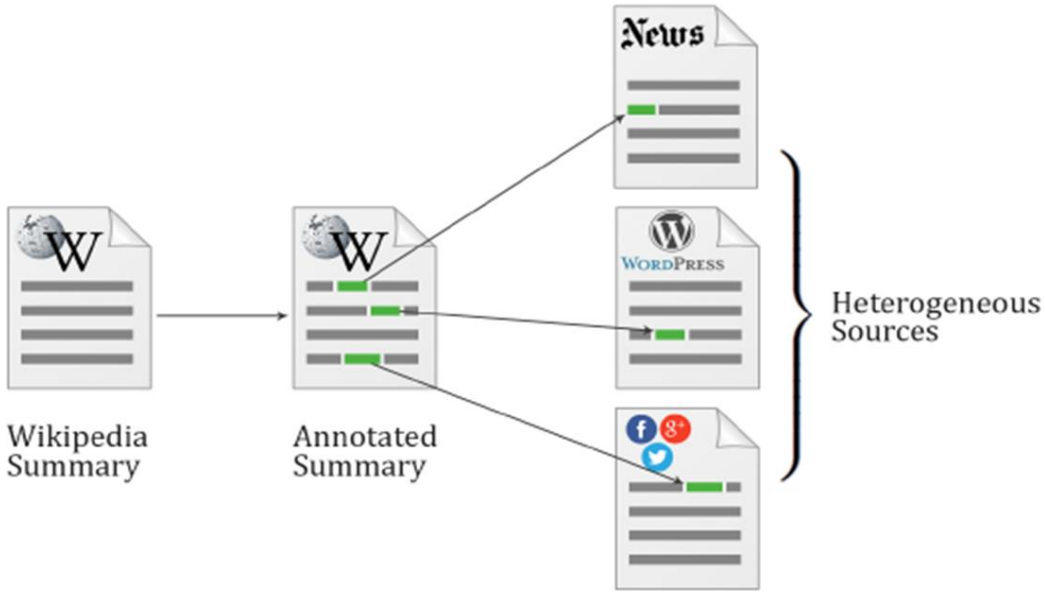


Figure 3-1: Corpus construction approach (Zopf et al., 2016)

The Wikipedia summaries are collected from articles of three domains:

- Domain 1: Art, Architecture, and Archaeology (D01).
- Domain 2: History (D02).
- Domain 3: Law, Politics, and Government (D03).

For each domain, a set of running number have been assigned to the topics: Topic 01 (T01), topic 02 (T02), and so one.
In the corpus, the folder D01T23 will materialize the topic 23 of Domain 01. These files have been assigned to the folders, following a guideline[1] illustrated with examples, and consisting of 12 steps.

- *Step 1*: create a folder for the topic that is going to be treated.
- *Step 2*: The lead (first paragraph of the Wikipedia article) of the given topic have to be extracted and stored in a summary file.

[1] https://github.com/AIPHES/hMDS/blob/master/Guidelines.md

- *Step 3*: Both the URL and the version-ID of the Wikipedia article must be saved. It is important to save the version-Id since the Wikipedia articles are regularly updated.
- *Step 4*: the most important information (information nuggets) must be extracted and saved in a file called nugget.txt. It has been asked to extract 10 to 20 of such fragments of information.
- *Step 5*: A web search engine (Google or Bing) can be used to retrieve the web pages. It has been recommended to at first log off from his account. Many criteria have been given the web pages to be selected (no PDF, no book, must be written in the English language, plain text can be must fully contain an information nugget, can be achieved in the web archive, extracted, and so one.). The quotes can be used to force an expression to appear in the responses returned by a search engine.
- *Step 6*: The URL of each selected web page must be saved in http://archive.org/web and also in a text file.
- *Step 7*: The information nugget used to retrieve web pages must be saved in a new line in the file of step 6.
- *Step 8*: Assign a document genre in a new line to the saved URL. There are 10 different documents genres taken into account in the corpus: article, forum post, twitter, organization, encyclopedic short, encyclopedic long, social media, scientific, education, and dialogues. How to distinguish between different genres is explained in (Zopf et al., 2016).
- *Step 9*: Extract and saved the title of the downloaded web page in the same text file that in step 6.
- *Step 10*: The plain text is extracted from the web page and stored in others files to enable further the training of classifiers with the plain text too (the extraction has been done manually and automatically using the Boilerplate tool).
- *Step 11*: the created folder in step 1 is stored in the repository
- *Step 12*: Check the topic in the list as done or skipped

This corpus is publicly available and can be downloaded at the URL mentioned in (Zopf et al., 2016). The downloaded corpus will look like Figure 3-2.

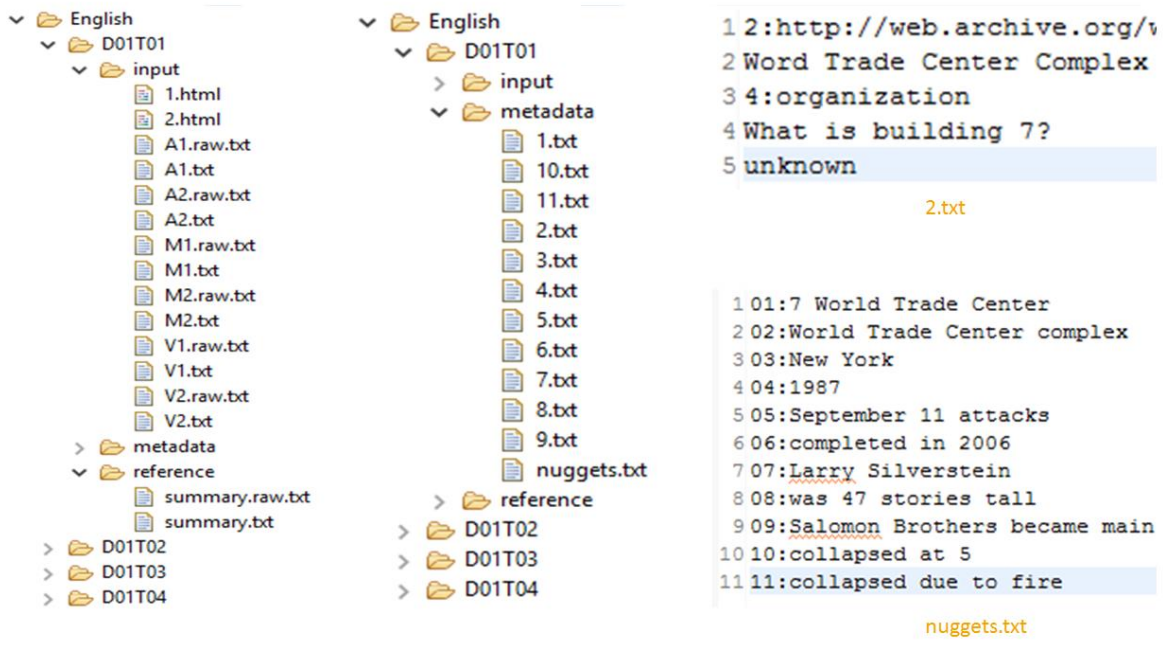


Figure 3-2: Structure of the corpus

As mentioned previously, the folder D01T23, for example, is short for topic 23 of Domain 01. Each of these folders contains three sub-folders:

Input: here there are files further used to train WEKA classifiers for the web page classification purpose. It contains downloaded HTML files and three types of extraction of its content: the manually extracted content (*Mx.txt*), the automatically extracted content (*Ax.txt*) through the *Boilerplate tool*[2] and the extraction of the visible text (*Vx.txt*) by setting an option in the Boilerplate tool. For each of these text files, another version is provided by writing only one sentence in a line (*file.raw.txt*).

Reference: contains the normal summary text file and another version (*summary.raw.txt*) with a single sentence pro line.

Metadata: It embodies many files among with a file called nuggets.txt. This file contains a list of information nuggets manually extracted from the *summary.txt* file. Each of these information nuggets is attached to a web page described in the above text files. For example, the *nuggets.txt* file in Figure 3-2 contains 11 information nuggets, that is the reason while there is 11 text file, named by the order number of each information nugget. Also, the second nugget is attached to the file *2.txt* which informs that: the information nugget *World Trade Center Complex* has been used to find the web page *2.html* located in *(http://web.archive.org/web/20160306083026/http:/rememberbuilding7.org/7-facts-about-building-7/)* This web page has for document genre *organization*, has the title "*what is building 7?*" and the copyright in unknown.

The corpus contained about one hundred topics on December 2016. These have been used to train some classifiers and used the generated model to classify newly downloaded web pages.

The present structure of the corpus cannot directly be used by WEKA to generate the training dataset and must be restructured. How the corpus needs to be restructured will be explained later.

[2] https://html5boilerplate.com/

4 Identification of Information Nuggets

The approach to be automated is described in Figure 3-1. As said in the corpus description, the summary file is built from the lead of Wikipedia articles

4.1 Wikipedia Summary as the Start Point

According to the Wikipedia featured article criteria[3], Wikipedia articles are well written, comprehensive, well-researched, neutral and stable (Zopf et al., 2016). The lead is the text written before the table of contents. It introduces the topic and gives the most important information from the rest of the document. Each sentence is supposed to be informative. Wikipedia specified that the lead must contain at most four paragraphs. Given that the Wikipedia articles are often written by many authors from different countries, different cultures, different knowledge, and are also updated regularly, they can be considered as an agreement of several authors on a given subject. Therefore they are considered as a high-quality summary. Because of these particularities, it can be concluded that they include all the important information, even the most recent ones. One can conclude that the retrieved web page will not bring new information anymore since Wikipedia articles are regularly updated. This corpus-building approach is therefore valid. Currently, only the files written in the English language has been taken into account for the creation of the corpus. Instead of a Wikipedia summary, any high-quality summary can be taken as the starting point of this process. This approach can be extended to other languages because of the simplicity of this model. Figure 4-1 give the structure of a Wikipedia article.

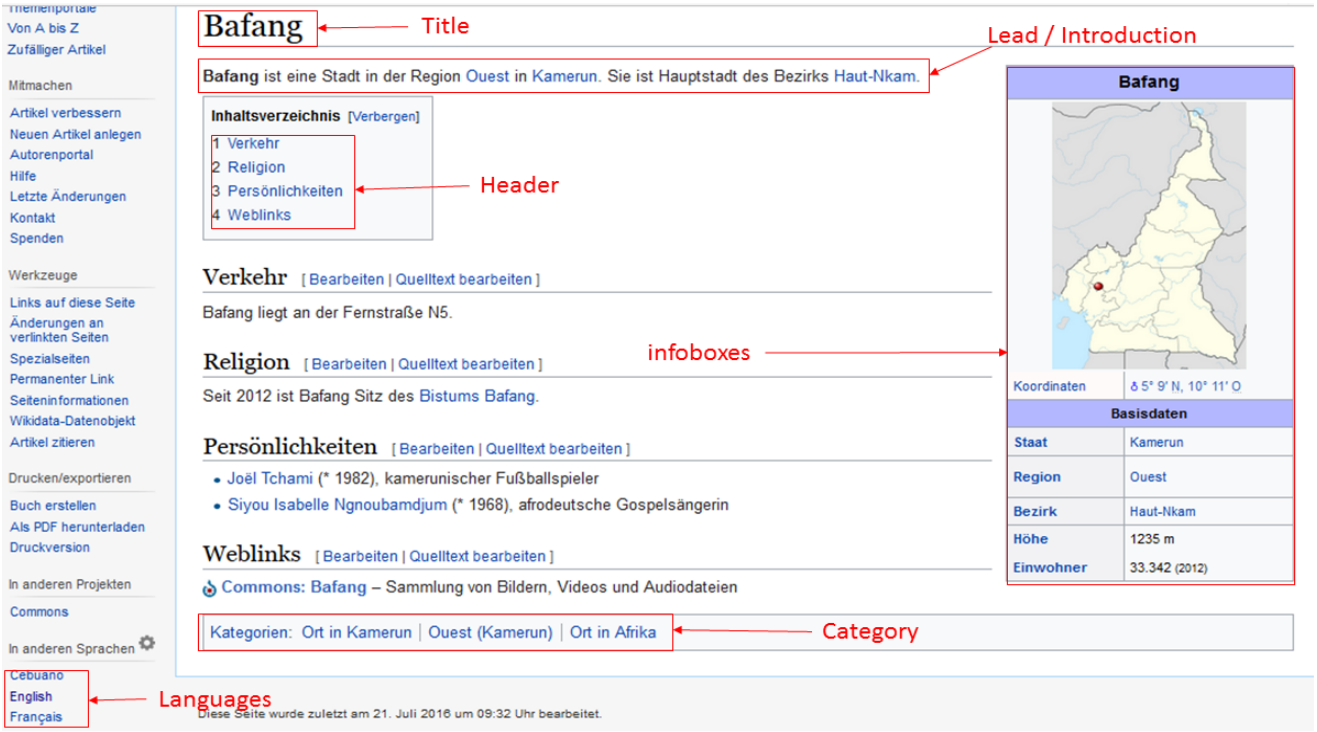


Figure 4-1: Wikipedia article features

Generally, at least one category is assigned to each Wikipedia article. This was useful to select the topic which belongs to the three domains considered for the corpus. Although Wikipedia articles are often written in many languages, the majority is written in English. This makes the research of the summaries easier because for each topic for which a Wikipedia article exist, it is likely to get an English written summary.
In the next section, it will be explained how the information nuggets can be programmatically identified.

[3] https://en.wikipedia.org/wiki/Wikipedia:Featured_article_criteria.

4.2 Extraction of Information Nuggets

7 World Trade Center is a building in the World Trade Center complex in Lower Manhattan, New York City. The current incarnation is the second building to bear that name and address in that location. The original structure was completed in 1987 and was destroyed in the September 11 attacks. The current building opened in 2006. Both buildings were developed by Larry Silverstein, who holds a ground lease for the site from the Port Authority of New York and New Jersey. The original 7 World Trade Center was 47 stories tall, clad in red exterior masonry, and occupied a trapezoidal footprint.

01: 7 World Trade Center **02:** World Trade Center complex **03:** New York **04:** 1987
05: September 11 attacks **06:** completed in 2006 **07:** Larry Silverstein **08:** was 47 stories tall

Figure 4-2: An example of a manually extracted information nugget list manually

Information nuggets are fragments of texts that give important information about a topic. They can be a word or an expression and must appear entirely in a requested web page (Figure 3-1). That means that they are supposed to represent atomic information (only the web pages on which they appear entirely are considered). Furthermore, they must be related to the topic. Recognizing such expressions is difficult because some words with the same semantical function (e.g. proper noun) are not necessarily information nuggets. One can remark from this short list in Figure 4-2 that they can be a date (1987), a proper noun (Larry Silverstein), a location (World Trade Center Complex), also another type of expression containing verbs, adjectives, (was 47 stories tall) and so forth. How can a computer be trained to extract such expressions? Two approaches based on the Part of Speech tags (POS) and on the named entities (NE) have been evaluated. There are presented in the followings subsections.

4.2.1 Information Nuggets from Part Of Speech (POS) Tags

The Stanford POS Tagger tool has been used to read the Part of Speech of each word of the summary files. A part of speech of a word represents its lexical function: this can be a preposition, a verb, a proper noun, and so one. The tool is implemented in Java, is based on the log-linear POS taggers described in (Toutanova, Klein, & Manning, 2003), and requires java 1.8+ to run. Depending on the model complexity, plenty of memory can be needed to train a tagger. The Stanford group suggests a minimum of 1GB. The first implementation of POS has been done by Kristina Toutanova and has been further extended by many other authors to support many other languages than English, and also to improve its performance, its speed, and its usability. The actually supported languages by this tool are: English, Arabic, Chinese, French, German, and Spanish. Furthermore, the tool contains a training file (with the extension .tagger) for each supported language. This training file includes all the information the tagger needs to tag a string. Figure 4-3 shows the result of the POS tagging on a short extract of a summary file from the core corpus (…\hMDS\English\D01T01\reference\summary.txt). A list of different functions words and their signification can found in the following paper (Marneffe & Manning, 2010).
The explanation of the abbreviations on the picture is:
NN = Noun; NNS = Noun, plural; NNP = Proper noun, sing.; IN = Preposition or subordinating conjunction; CD = Cardinal number; VBN = Verb; JJ = Adjective and TO = to).

CD = cardinal number, NNP = proper noun, sing. JJ = adjective, NN = noun, sing., VBD = verb past tense, TO = "to", IN = preposition/sub-conj

Extraction of expression building from the following tags: NN [S, P, PS], IN, CD, JJ.

Figure 4-3: Identification of Part Of Speech tags in a summary text.

Eight manually extracted information nuggets that has been provided with the corpus are highlight in blue in Figure 4-3. Some of them have been analyzed to discover how they are built:

- *World trade Center* = NNP + NNP + NNP
- *Port Authority of New York* = NNP + NNP + IN + NNP + NNP
- *September 11, 2001* = NNP + CD + CD
- *collapsed due to fire* = VBN + JJ + To + NN

With many other experiences, one can remark that the information nuggets are often built using a consecutive appearance of one of the following function words NN, NNS, NNP, NNPS, IN, CD and JJ. They can be extracted in a Java program using the following four regex patterns NN(.)*, IN, CD, and JJ. The function word IN is not extracted alone, it must always connect two of the three other function words. The solution based on this approach will be called in the rest of the paper by Compound POS (CPOS). Applying this regex pattern on a summary file in core corpus *D01T01\reference\summary.txt* (the whole summary from Figure 4-2), 52 expressions that fulfill these conditions are obtained (Figure 4-4).

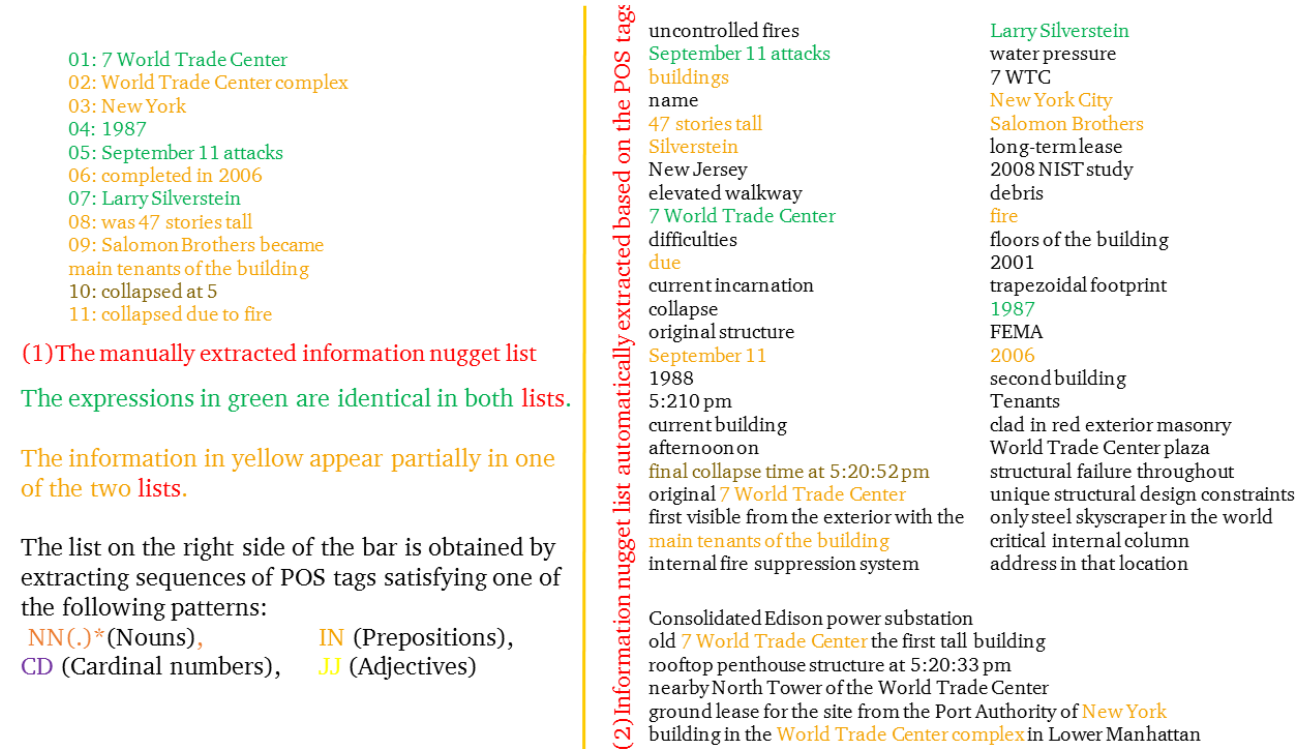


Figure 4-4: Extraction of information nuggets from the POS tags

From this result, some remarks can be made:

- Some expressions appear exactly or partially in the both lists.
- The original is not complete because it has been asking only to extract 10 to 20 information nuggets manually.
- Some nuggets have been incompletely extracted (collapsed at 5 # collapse time at 5:20:52pm)
- Many expressions in the automatically extracted list are information nuggets, but they do not appear in the manually extracted list (e.g. 7WTC, New Jersey, 2008 NIST study, etc.).
- Also, many elements on the automatically extracted list cannot be considered as information nugget (e.g. name, difficulties, due, etc.)
- In reality, the precision, recall, and F-Measure are higher since the manually extracted list of information nuggets in incomplete.

To evaluate the proposed method, it has been considered that an expression is automatically found if it appeared entirely on both lists, or if it appears partially in one of the lists. For Instance, it has been assumed that *"Port Authority"* and *"Port Authority of New York"* are identical. In the same way *"World Trade Center"* and *"North Tower of the World Trade Center."* In Figure 4-4, one can see that:

- Four expressions appear exactly in both lists (*7 World Trade Center, 1987, September 11 attacks, and Larry Silverstein*)
- Three expressions in the manually extracted list appear partially in another list (*New York, 7 World Trade Center, ground lease for the site from the Port Authority of New York*)
- 10 expressions in the retrieved list appear in the original list (*buildings, 47 stories tall, Silverstein, September 11, main tenants of the building, Salomon Brothers, fire, 2006, due, tenants*).
- It has been considered that the information nugget *collapsed at 5* has not be found because it has been modified by the manual extraction and cannot entirely be found any more in the summary file.

If the Recall (R) is defined as the percentage of manually extracted information nuggets, which have been found by this approach, then $\boldsymbol{R = \frac{10}{11} = 90.91\ \%}$ since 10 expressions over 11 from the left list (Figure 4-4) appear on the right side (green and yellow colored). The precision (P) can be defined as the percentage of the information nugget available in the automatically retrieved list:
$\boldsymbol{P = \frac{(4+10)}{52} = 26.92\ \%}$, since four information nuggets appear entirely in both lists and 10 nuggets from the retrieved list are part of the manually extracted nugget list. The F-Measure (F) is define as follow: $\boldsymbol{F = \frac{2PR}{P+R} = 41.54\ \%}$. These values have been used to evaluate the performance of the proposed method, but they are higher in the reality, since the manually extracted list is incomplete (some others information nuggets can be added). This solution has been evaluated on the whole corpus and got following average results: *Precision: 27.39 %, Recall = 95.74 % and F-Measure = 42.59 %* (Figure 4-5). The low precision is predictable since the retrieved list is three times longer than the list of manual extraction. That means that a maximum of 34 % from precision can be reached in average for the corpus. The precision can be ameliorated by adding more information nuggets in the manually extracted list or by reducing the size of the retrieved list (or by eliminating the false information nuggets).

Folder	Precision	Recall	Nuggets	CPOS
D01T06	0,6666667	1	10	12
D02T12	0,25	0,85714287	14	12
D02T22	0,30769232	0,8181818	11	13
D03T03	0,23076923	0,8181818	11	13
D03T15	0,5	0,84615386	13	14
D03T33	0,14285715	0,72727275	11	14
D03T18	0,29411766	0,72727275	11	17
D03T25	0,3888889	0,9285714	14	18
D02T11	0,15789473	1	10	19
D01T13	0,8	1	17	20
D02T24	0,5714286	1	10	21

Average Nugget Length	13,86813187
Maximum	20
Minimum	10
Average NE Length	42,96703297
Maximum	117
Minimum	12
Average Precision	0,273885641
Average Recall	0,957427473
Average F-Measure	0,42592844

Figure 4-5: Evaluation of the part of speech based solution on the corpus

By removing some elements in the retrieved list without taking into account if they are true information nuggets or not, the precision may grow while the recall may decrease. It is, therefore, better to identify expressions that are likely false information nuggets and remove them. How to identify these false expressions is difficult. The idea is to consider sentences where the topic of the summary appears. ClausIE is a clause-based open information extraction tool that can be used for this purpose. It is presented in the next subsection.

4.2.2 ClausIE to perform the CPOS List

ClausIE is a powerful tool that can help to detect clauses within a sentence. A clause is a part of a sentence that has its own subject and its own verb. A clause must have at least these two parts. Optionally, it can contain a complement (C), an indirect object (O_i), a direct object (O), and one or more adverbials (A). (Del Corro & Gemulla, 2013) describe ClausIE as "*based on dependency parsing for clause detection in a sentence and a small set of domain-independent lexica to detect clause types.*"[4]. In their research, they compare ClausIE to others approach performing the same task (OLLIE, Reverb, WOE) and as a result, ClausIE outputs more propositions, and its precision was similar or higher.
Furthermore, (Del Corro & Gemulla, 2013) mention that ClausIE doesn't require any training data to split sentences and it treats sentences one after the other without post-processing.
For the moment, this tool can only analyze texts written in the English language. It uses grammatical rules to determine the clauses from an input sentence and then determines the type of clause given the grammatical functions of each component. The tool can be tested using the online demo or by

[4] Dependency-based methods for syntactic parsing.

downloading it as source code and binaries on their website[5]. With the online demo, just one sentence has to be given every time for clause derivations. With the binaries, one can specify an input file containing a set of sentences to be analyzed. But, only one sentence muss be written per line in the file. The difficulty here to differentiate from a summary text a *point* punctuation from a decimal number. This can be done by using the Regex. Some patterns to the output of ClausIE are shown in Table 4-1. The abbreviations refer to:
S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial,
V_i: Intransitive verb, V_c: Copular verb[6], V_c: Extended-copular verb, V_{mt}: Monotransitive verb,
V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb.

Pattern	Clause type	example	Derived clauses
SV_i	(SV)	Wolfgang Goethe died.	"Wolfgang Goethe" "died."
SV_eA	(SVA)	WG remained in Frankfurt.	"WG" "remained" "in Frankfurt."
SV_cC	(SVC)	WG is German.	"WG" "is" "German."
$SV_{mt}O$	(SVO)	WG learn many languages.	"WG" "learned" "many languages."
$SV_{dt}O_iO$	(SVOO)	Strbrg university gave WG the doctor title.	"Strbrg university" "gave" "WG the doctor title."
$SV_{ct}OA$	(SVOA)	She showed his husband to the office.	"She" "showed" "his husband to the office." "his" "has" "husband."
$SV_{ct}OC$	(SVOC)	WG declare the meeting closed.	"WG" "declare" "the meeting closed" "the meeting" "closed."
SViAA	(SVAA)	WG died in Weimar in 1832.	"WG" "died" "in Weimar." "WG" "died" "in 1832." "WG" "died."
SV_eAA	(SVAA)	WG remained in Weimar until his death.	"WG" "remained" "in Weimar until his death." "WG" "remained" "in Weimar." "his" "has" "death"
SV_cCA	(SVCA)	WG is a novelist of the 18th century.	"WG" "is" "a novelist of the 18th century." "WG" "is" "a novelist."
$SV_{mt}OA$	(SVOA)	WG learnt English in 1758.	"WG" "learned" "English in 1758."
$ASV_{mt}O$	(ASVO)	in 1758, WG learnt English.	"WG" "learned" "English in 1758." "WG" "learned" "English."

Table 4-1: Clauses extraction with ClausIE

1- 7 World Trade Center is a building in the World Trade Center complex in Lower Manhattan, New York City.
"7 World Trade Center" "is" "a building in the World Trade Center complex in Lower Manhattan"
"7 World Trade Center" "is" "a building"
"Lower Manhattan" "is" "New York City"

2- An elevated walkway connected the building to the World Trade Center plaza.
"An elevated walkway" "connected" "the building to the World Trade Center plaza"
"An elevated walkway" "connected" "the building"

3- The original structure was completed in 1987 and was destroyed in the September 11 attacks.
"The original structure" "was completed" "in 1987"
"The original structure" "was destroyed" "in the September 11 attacks"

Figure 4-6: Example of generated clauses with ClausIE

The idea is to split the sentences of the summary file into clauses and then consider only those where the topic appears. If the topic appears in the subject part of a clause, the object gives more information

5 http://resources.mpi-inf.mpg.de/d5/clausie/

6 A copular verb *is a* special kind of verb used to join an adjective or noun complement to a subject.

about it (sentence 1 in Figure 4-6). Similarly, if the object contains the topic, then the subject would bring interesting information about it (sentence 2 in Figure 4-6). The problem with this solution is that the topic can appear a few times in the summary file, leading to a short list with the CPOS solution.
Most of the time, the topic is replaced by synonyms, pronouns or others expressions that refer to the same thing (sentence 3 in Figure 4-6). They are called the referents of the topic. If such referents in a summary file can be identified, then the new input file may be built from clauses containing the topic or one of its referents. This may help to eliminate some false expressions considered as information nuggets from out CPOS lists. A referent of a keyword can be found using a co-reference resolution tool. Some of them are presented in the following section.

4.2.3 Co-reference Resolution Tools to improve the Extraction of Information Nugget

The Stanford group define the co-reference resolution as "*the task of finding all expressions that refer to the same entity in a text.*"[7]. Many tools have been explored in order to find out which one is able to detect the referents of the topic in each summary file more accurately. In the following some are presented:

The Stanford Co-reference Resolution tool[8] (Recasens, Marneffe, & Potts, 2013)**:** This tool is often used and cited in the papers dealing with the co-reference resolution. It is available for the English and Chinese language and implements both pronominal and nominal co-reference resolution. They propose three approaches to detect the referents of an expression in a text:

- Deterministic: this system is multi-pass sieve rule-based. It also includes pronominal resolutions, string match, and the rules. (Recasens et al., 2013) says that there are 10 rules in all, but they have not been mentioned in the paper.
- Statistical: The system iterates through a document and creates a co-reference link between each current and a preceding mention.
- Neural: this system is based on the neural network and also operates with the mention-ranking model. Experiences on the Stanford homepage show, that this system is very slow, but produces better results.

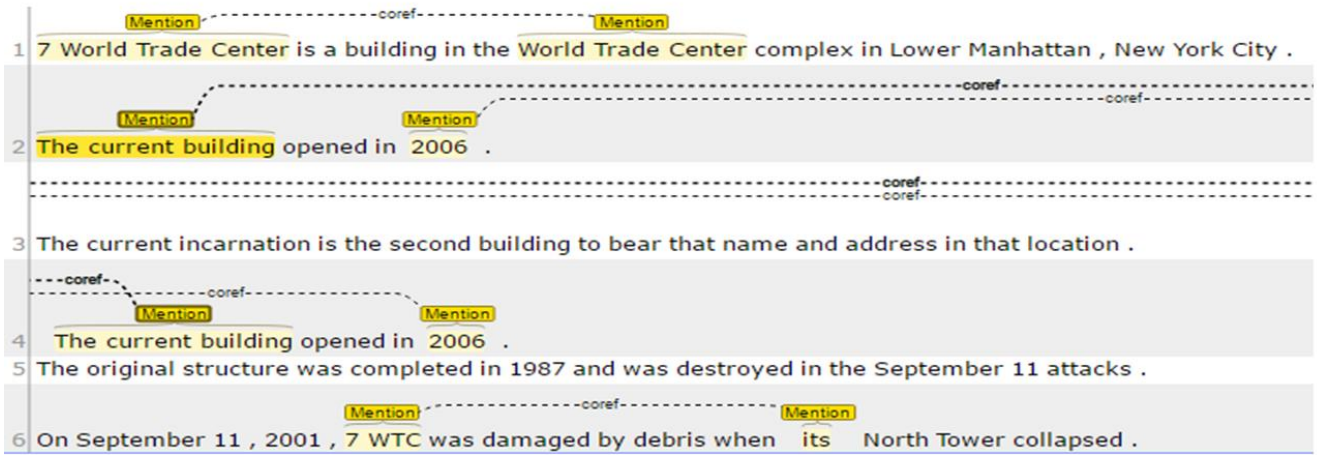


Figure 4-7: Co-reference resolution with the Stanford tool

This tool has been tested on some summary files, and the results were not satisfying for all the topics. Figure 4-7 shows the example of four sentences that are processed using their online Demo tool[9]. There is not any option to switch between the different systems (deterministic, statistical and neural). The

[7] *https://nlp.stanford.edu/projects/coref.shtml*

[8] http://stanfordnlp.github.io/CoreNLP/coref.html

[9] http://corenlp.run/

following referents have been identified: *World Trade Center = World Trade Center, The current building = the current building, 2006 =2006, and 7 WTC = its*. As one can remark, the tool was only able to link the same expressions or pronouns that refer to an expression. The tool can neither identify *7 WTC* as a referent of *7 World Trade Center*, nor *the current building* as another referent. A good co-reference resolution tool may catch following referents for the topic World Trade Center (from the sentences on Figure 4-7):
World Trade Center = {*the current building, the current incarnation, the original structure, 7 WTC, its*}.

BART (Beautiful Anaphora Resolution Toolkit)[10](Versley et al., 2008): This tool includes a variety of ML approaches and can use several ML toolkits, including WEKA, the Berkeley parser, the Stanford NER(Named Entity Recognition), and an included MaxEnt implementation. It can be tested on a Linux operation system. For larger quantities of text, their REST-based web service must be used. The results with this tool were similar to the online demo of the Stanford co-reference resolution tool. With the same text as above (Figure 4-7), the same referents have been returned. This tool is open source, and 9 people (listed on their web page) have been involved in its creation and evolution.

Reconcile[11] (Stoyanov et al., 2010)**:** it uses the Stanford Named Entity Recognition System, the Berkeley Parser and supervised machine learning classifiers to handle co-reference resolutions. It outperforms the Stanford co-reference resolution tool on the same datasets, but cannot be able to find enough referents in many other summaries.

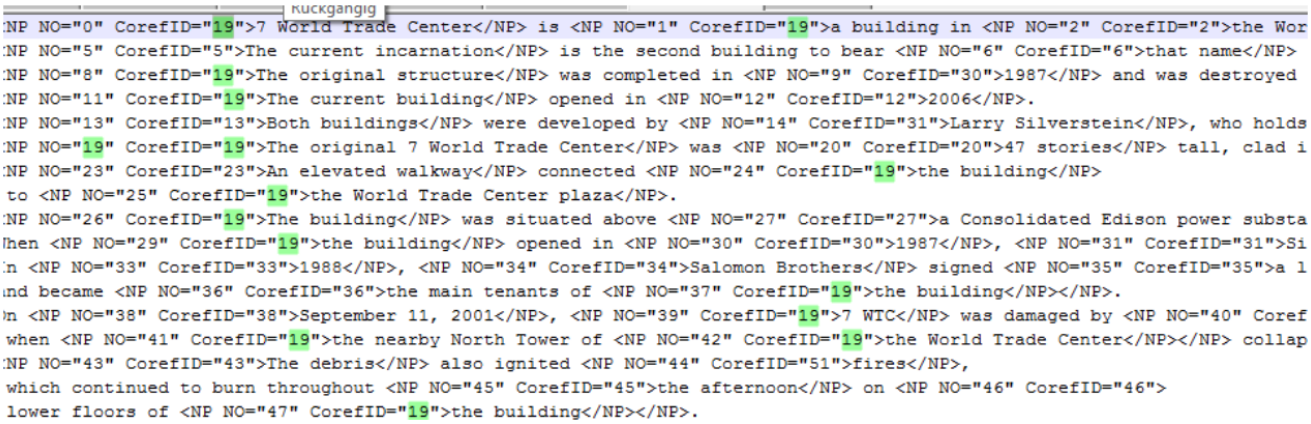


Figure 4-8: Co-reference resolution with Reconcile

In the same summary, this tool has been able to identify following referents for the topic *World Trade Center = {a building in, The original structure, The current building, The original 7 World Trade Center, the building, the World Trade Center plaza, 7 WTC, the old 7 World Trade Center, the first tall building, the nearby North Tower of*}. The tool achieved good results on some summaries but also identified too few on none referents in some other summaries. For this reason, it cannot be combined with ClausIe to filter the CPOS lists better.

Others tools for the co-reference resolution have been tested: cherrypicker[12] (Ng & Cardie, 2002), OpenNLP[13] and GuiTAR[14], but the results were worse than the three tools that were mentioned above.

[10] http://www.bart-coref.org/

[11] http://www.cs.utah.edu/nlp/reconcile/.

[12] http://www.hlt.utdallas.edu/~altaf/cherrypicker.html

[13] http://incubator.apache.org/opennlp/.

[14] http://cswww.essex.ac.uk/Research/nle/GuiTAR/.

It has been said earlier that the precision of the CPOS solution could be improved by adding more information nuggets in the manually extracted nugget list or by removing some false expression from the automatically retrieved list. After some experiments, the conclusion is that it is difficult to recognize these false expressions in order to remove them. This is because not any trustworthy co-reference resolution tool has been found, that can return good results for each of the summary files. If a perfect co-reference resolution tool is found, then the CPOS lists can be filtered using the followings steps:
At first, the sentence composing the summary file must be split into clauses using the ClausIE tool. Secondly, the referents of the topic are identified by using the co-reference resolution tool. Finally, only the clauses containing the topic or at least one of its referents are selected and performed in the same way as in section 4.2.1 to generate the CPOS lists. This process is highlighted on a sentence in Figure 4-9. ClausIE can cut the sentence "*In 1988, Salomon Brothers signed a long-term lease, and became the main tenants of the building*." into five clauses. A good co-reference resolution tool may identify the *building* as a referent of *World Trade Center,* and for this reason, only the clause containing it (the fourth sentence in Figure 4-9 will be processed using the POS solution. As a result, the expressions *Salomon Brothers* and *the main tenants of the building* will remain in the retrieved list while *long-term lease* and *1988* are removed (Figure 4-4).

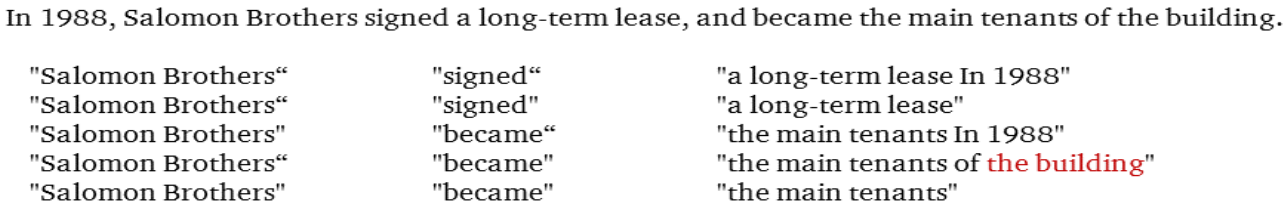


Figure 4-9: How ClausIE and a co-reference resolution tool can be applied on a sentence

Considering the manually extracted information nuggets lists, it is remarked that most of them are Named Entities (NE). Another solution of information nuggets detection based on them is presented in the next subsection.

4.2.4 Nuggets from Named Entities

The named entities (NE) refer to entities like person, location, time and so forth. The Stanford group proposes a tool named Stanford Named Entity Recognizer[15] that can help to catch such entities in a text. The tool is written in Java and can be used with their proposed online Demo tool or in a java program. In a java program, the following three classifiers can be used:

- classifiers/english.all.3class.distsim.crf.ser.gz: this classifier can identify the person, location, and organization entities in the text.
- classifiers/english.conll.4class.distsim.crf.ser.gz: available in four languages (English, German, Spanish and Chinese), this classifier proposes in addition to the first one the Miscellaneous type. An entity is classified as miscellaneous when it belongs to more than one entity type, depending on the context it is used in (in Figure 4-10, *World Trade Center* is detected as an *organization* and as a *location* in another context, for this reason, it is a miscellaneous entity).
- classifiers/english.muc.7class.distsim.crf.ser.gz: this classifier can cover four additional entities (time, date, percent and money) in addition to the first classifier.

[15] http://nlp.stanford.edu/software/CRF-NER.html

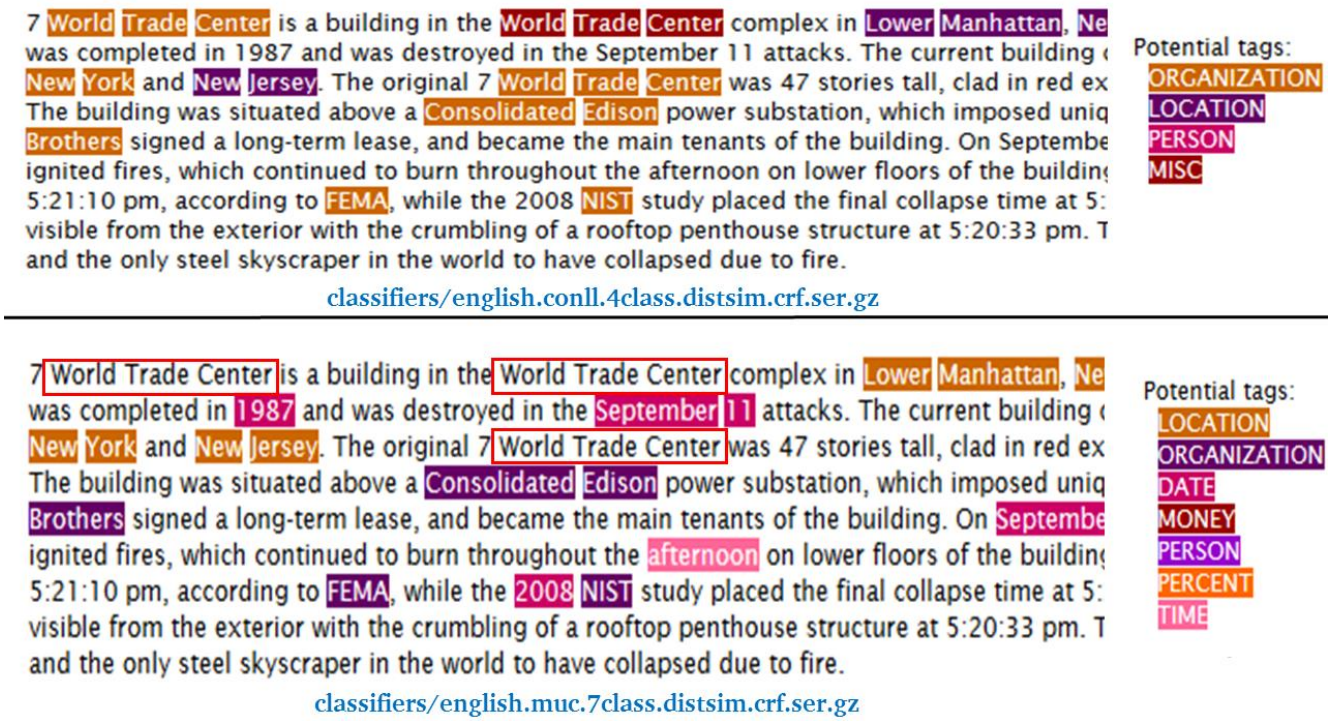


Figure 4-10: Advantages of combining the classifiers to extract the named entities

The two last mentioned classifiers have been combined in a java implementation to extract all the eight named entity types they can solve: person, location, organization, time, date, percent, money, and miscellaneous. Another advantage of this combination is that for the same entity type, a classifier can cover an expression missed by another. In Figure 4-10, one can see that both classifiers can recognize the organization entity type. However, *World Trade Center* as an organization is missed by the second classifier. Another important remark is that the information nugget *World Trade Center* is built by concatenating the named entities *World, Trade, and Center*. In a java program, the classifiers return them separately. Because they belong to the same entity type and are separated by a space, they can be concatenated to build a unique information nugget. Figure 4-11 shows the result of this approach on the *…\hMDS\English\D01T01\reference\summary.txt* file

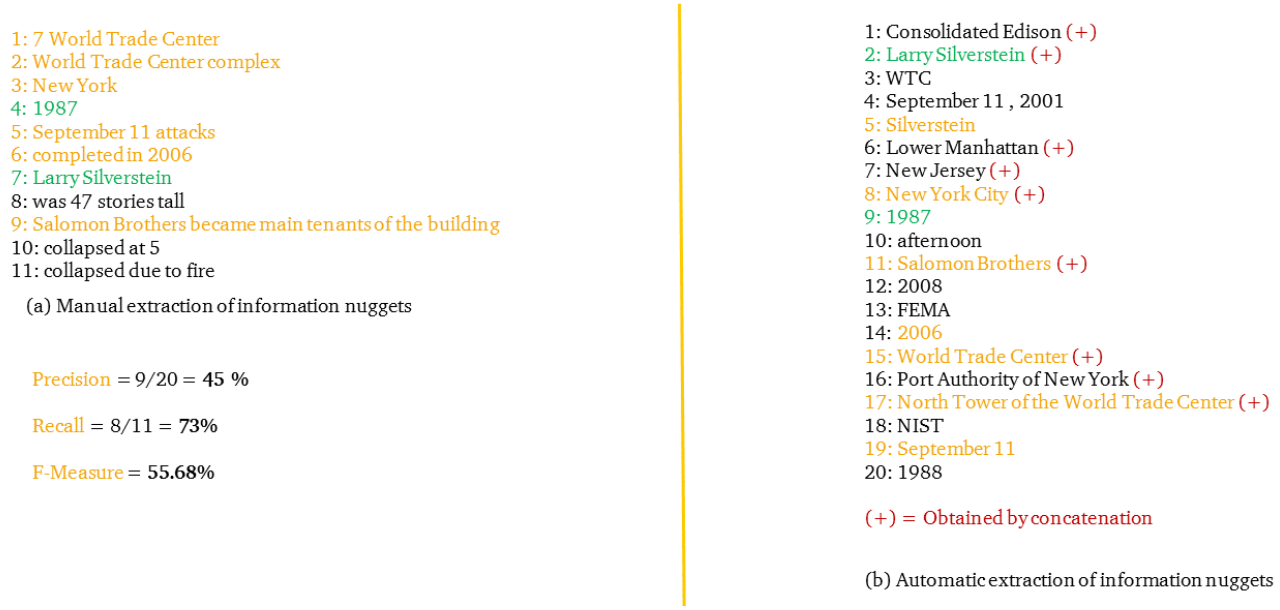


Figure 4-11: Information nugget extraction from named entities

Some remarks can be made from this result:

- As mentioned in the proposed solution using the part of speech, the manually extracted list of information nuggets is incomplete because it has been asked only to extract 10 to 20.
- The retrieved list is shorter than the CPOS list (20 < 52)
- Information nuggets can be obtained by concatenating the named entities
- Many expressions in the automatically extracted list are information nuggets, but they were omitted in the manually extracted list (e.g. WTC, September 11, 2001, New Jersey, etc.).
- A few elements in the automatically extracted list are not information nuggets (e.g. afternoon – correspond to the named entities type *time*)
- Some information nuggets do not contain named entities (was 47 stories tall, collapsed at 5, collapsed due to fire), and cannot be found with this approach.
- The F-Measure is better than by the solution based on POS (55.68 % > 42.59 %).
- In reality, the precision, recall, and F-Measure are higher since the manually extracted list of information nuggets in incomplete.

The result in Figure 4-12 represents the whole corpus. The precision, recall, and F-Measure have been calculated in the same manner as with the CPOS solution: *Precision: 58.47 %, Recall = 72.57 % and F-Measure = 64.76 %*. The precision has doubled compared to the one obtained with the part of speech solution, and the F-Measure has increased too. Following results have been obtained with the CPOS solution: *Precision: 27.39 %, Recall = 95.74 % and F-Measure = 42.59 %* (Figure 4-5).

One remark from the results in Figure 4-12, that the recall is lower than the precision when the number of named entities (considered as information nuggets) is less than the number of the information nuggets (manually extracted). If a summary file only contains only one named entity ad if it has been manually extracted, then the precision of 100 % is achieved, since the retrieved list contains a unique element, which appears in another list. A consequence is that only one web page will be downloaded for this topic because only one web page must be assigned to an information nugget.

Folder	Precision	Recall	Nuggets	Named-Entity
D01T01	0,35	0,72727275	11	20
D01T02	0,5925926	1	16	27
D01T03	0,61538464	0,5294118	17	13
D01T04	0,5294118	0,9	10	17
D01T05	0,6	0,9230769	13	20
D01T06	0,8	0,4	10	5
D01T07	0,4827586	0,8235294	17	29
D01T08	0,6363636	0,8235294	17	22
D01T09	0,33333334	0,4	10	12
D01T10	0,61538464	1	17	26
D01T11	0,6666667	0,6666667	12	12
D01T12	0,68421054	0,8235294	17	19
D01T13	0,75	0,47058824	17	8
D01T14	0,5714286	0,8	10	14

Average Nugget Length	13,8681319
Maximum	20
Minimum	10
Average NE Length	19,1098901
Maximum	65
Minimum	2
Average Precision	0,58467414
Average Recall	0,72565024
Average F-Measure	0,64757847
Recall < Precision	
N° Named-Entities < N° Nuggets	

(b) Automatically extraction of information nuggets

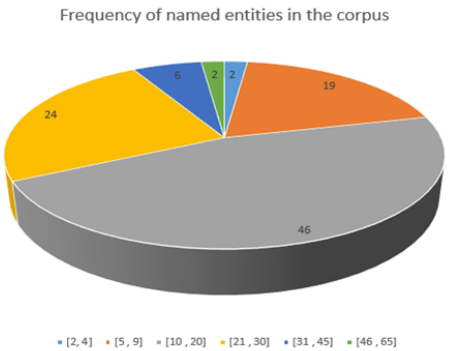


Figure 4-12: The performance of the information nugget extraction using named entities

The repartition of named entities in the whole corpus have been evaluated, and the result is presented in the diagram on the right in Figure 4-12. From this diagram, it can be read that:
Less than 5 NE is found in 2 files, 5 and 9 named entities appear in 19 files, 46 files have been found containing 10 to 20 NE (this correspond to the interval given for the manual extraction), 24 files contain 21 to 30 NE, 6 files contain 31 to 45 NE and more than 45 NE have to be found in 2 files. The conclusion is that 21.21 % of the summaries in the corpus contain less than 10 NE. Another idea is to replace the NE list by the CPOS list when they contain too few elements.

4.2.5 Information Nuggets by Merging the Named Entities and the CPOS List

The F-Measure was higher by considering the solution based on the named entities. However, about 21 % of the summaries contain less than 10 named entities. It was asked to extract 10 to 20 information nuggets manually. This condition can be satisfied when the lists of named entities which contain less than 10 items are replaced by the list of part of speech (always at least 13 expressions have been returned for each summary file using the part of speech). The behavior of the F-Measure when this constraint is applied is presented in (Table 4-2).

	Precision	**Recall**	**F-Measure**
Nuggets per files: minimum = 2; maximum = 65	58.47%	72.57%	64.76 %
Nuggets per files: minimum = 2; maximum = 20	58.15%	65%	61.38 %
Nuggets per files: minimum = 5; maximum = 20	56.87%	66.50%	61.31 %
Nuggets per files: minimum = 10; maximum = 20	53.45%	71.45%	61.15 %

Table 4-2: Comparison of precision and recall when replacing some nuggets files by the POS files

The first line of the table represents the result achieved by considering the named entities as information nugget. The results in the second line represent the case where the maximum of 20 NE is considered. They are selected randomly. The result in the third line is achieved t by considering the CPOS list instead of the NE list, only in the case where the NE file contains less than five elements. The last line corresponds to the scenario where 10 and 20 information nuggets have to be extracted. The same rule as in the third line of the table has been used. It consists of replacing the named entities lists when they contain less than 10 items by the Part of Speech lists. A remark is that the F-Measure decreases when the NE files are gradually replaced with the CPOS files.
Next, the solution based on the named entities is considered. So, the information nuggets of a summary correspond to the named entities person, location, organization, time, date, percent, money, and miscellaneous that it contains.

5 Retrieval of Web Pages

There exist many search engines that can be used to incorporate search functionality into an application: e.g. the Bing Web Search API[16] or the Custom Search Engine of Google. The Custom Search Engine of Google allows 100 search queries per day for free and additional queries at the cost $5 per 1000 queries. JSON format is the default output to the request. Because many search queries for free have been needed at the beginning of the tests, the Bing Web Search API has be chosen. It allows 1000 calls free per month, up to 3 months and then will cost $3 every month (S1 Standard). One can then request web pages related to 1000 information nuggets at the beginning of the tests and complete them in the following month. In December 2016, the corpus contained 99 topics. As in the manually extracted information nuggets lists, a maximum of 20 named entities (taken as information nuggets) per summary has been considered. That corresponds to a maximum of 1980 information nuggets. Web pages for the first 1000 information nuggets can be downloaded in the first month of the registration to this API and the rest completed at the beginning of the following month. Other options are available, but the S1 standard already allows us to retrieve a web page for all the information nuggets freely. Moreover, the Bing Web Search engine is easy to configure:

- One can specify the language of returned web page (the English language has been chosen).
- More than 10 URLs can be retrieved for each query (the default value 10 can be changed).
- The results can be filtered so that only images, news, videos, web pages are returned. They can also be combined within a single query (the web page option has been chosen).
- The data formats for the response are JSON and XML leading that these responses can be used in many programming languages.
- The response object contains important information for each returned web page such as the displayed URL, document title, a snippet of information about the web page.

The search has been performed using the named entities lists (considered as the information nuggets lists) with a maximum of 20 selected expressions. The search for an information nugget muss be atomic; that means an information nugget must appear entirely in each returned website. An information nugget could be a word or an expression (e.g. *Silverstein* or *September 11 attacks*). While searching, each information nugget has to be combined with the topic it belongs to because the requested web page has to be closed to it. For example, if the topic is "World Trade Center" then it must be combined with the first information nugget (*e.g. "September 11 attacks"*) to request the first 10 pages. Then the same topic is combined with a second information nugget (*e.g. "Larry Silverstein"*) and request another 10 pages and so one. The + sign can be added before an expression to force it to appear in the query (*e.g. +"World Trade Center"*). If for a summary file, 15 information nuggets are extracted and that 10 URLs are asked for each, then theoretically 150 HTML web pages would be downloaded. In reality, fewer files are download, because PDF file has to be discarded and some stored URLs were truncated, leading to a *malformed URL Exception*. There is also many duplications because the same files can be downloaded several times if they appear in the search response of others information nuggets. To each information nugget must be assigned at most one web page.

If the number of nuggets is greater than the number of document genre (more than 10), then several information nuggets may have HTML files of the same document genre (e.g. the nuggets N2 and N5 in Figure 5-1(a)). If on the other hand, the number of nuggets is less than the number of document genres (less than 10), it will be tried to assign a different document genre for each nugget (there is no document for the genre G2 in Figure 5-1(b)). In this case, all the document genres cannot be covered. The goal of this classification is to cover as many document genres as possible (*article, forum post, microblog, organization, encyclopedic-short, encyclopedic-long, social media, scientific, education, and dialogues*).

[16] https://www.microsoft.com/cognitive-services/en-us/bing-web-search-api.

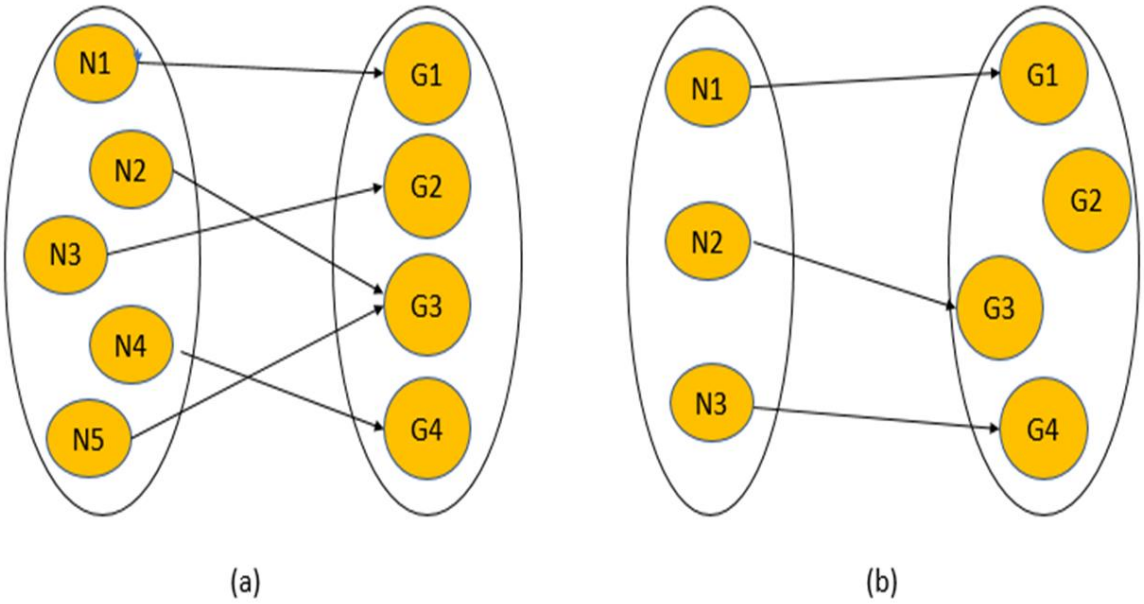


Figure 5-1: Webpage assignment

After downloading some web pages for each information nugget, they are going to be classified and for each information nugget, at most one web page as explained earlier must be assigned.

6 Classification of Source Documents

As per the Machine Learning Group at the University of Waikato WEKA is described as a collection of machine learning algorithms developed to solve data mining problems. These algorithms can be imported into a java program or in a graphical way using the tool. We use this tool in this thesis for pre-procession and classification. It offers others functionalities as regression, clustering, association rules and visualization. WEKA is well-suited for the experiments we are going to perform. It has been used to generate the training dataset from the corpus. However, the corpus needs to be reorganized beforehand. All the classifications results have been obtained using the 10 folds cross-validation.

6.1 Restructuration of the Corpus for WEKA Usage

The corpus is provided with different extraction types of a web page (downloaded HTML files, text files for the manual extraction of the content and text files for the automatically extraction of visible text using the Boilerplate tool). WEKA is used for training the corpus and determining the best classifier and the best dataset, which leads to the best classification of newly downloaded web pages. Firstly, the corpus is restructured so that WEKA can read the documents and generate the ARFF files (ARFF is the file data format use in WEKA).

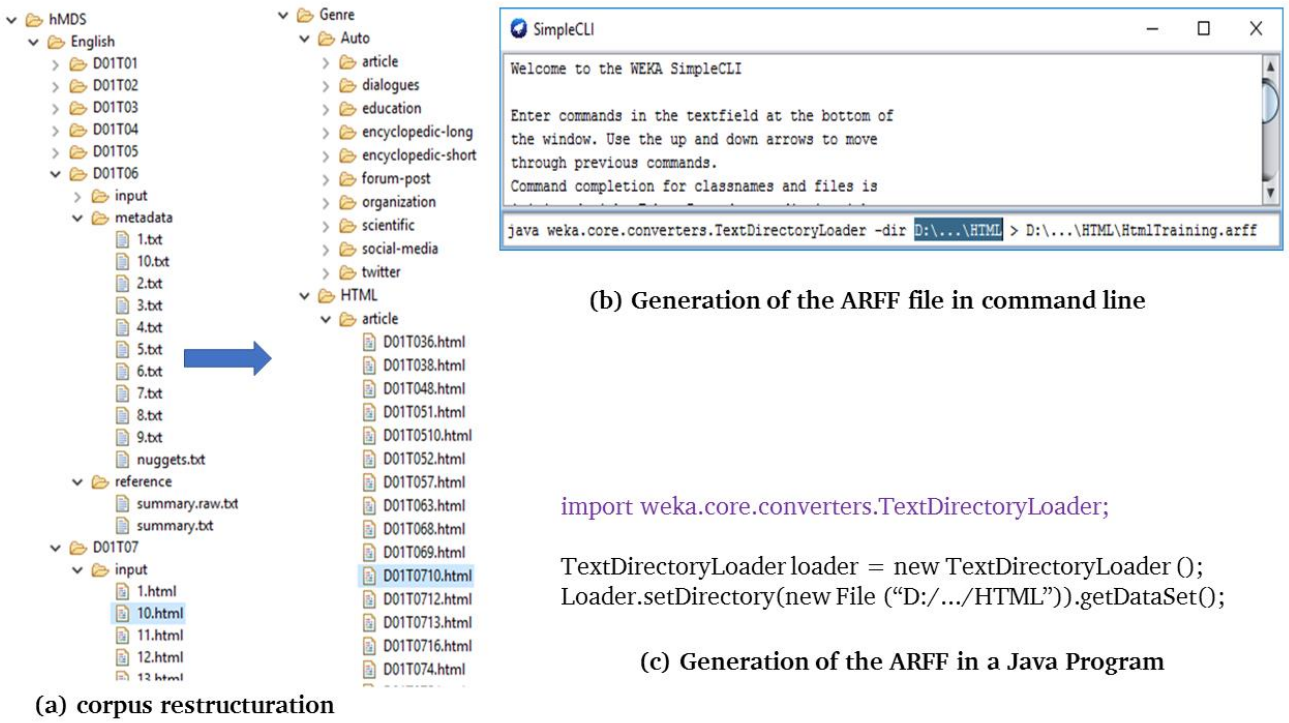


Figure 6-1: Corpus Restructuration for WEKA use

WEKA assigns to each file, its parent folder as the class attribute. For this reason, a new root folder is created (c.f. example of *Genre* in Figure 6-1 (a)). This root folder contains three subfolders for the three extraction types that are trained in WEKA. These three folders are *Auto* for the automatic extraction, *Manual* for the manual extraction of HTML contents and *HTML* for the downloaded web pages. In the following example the process of generating the dataset HTML.arff is explained:

- Create in the folder HTML 10 subfolders and rename them with the 10 documents genres.
- Copy each HTML file from the corpus under the folder that corresponds to its document genre. For example, the document genre of the file "*hMDS\English\D01T07\input\10.html*" can be read

in the file "*hMDS\English\D01T07\metadata\10.txt*". It is an *article*. That is the reason while it is classified under the subfolder *article* in Figure 6-1(a). In short, the folder HTML will then contain 10 subfolder corresponding to the 10 documents genres, and each subfolder will only contain HTML files assigned to this genre (Figure 6-1 (a)).

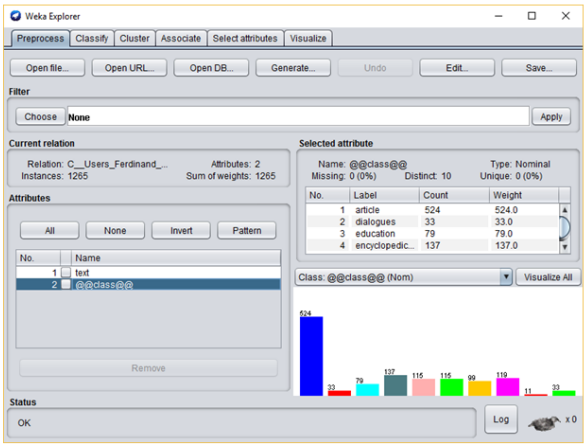
- This is followed by generation of the dataset for the HTML files using the WEKA command line interface (Figure 6-1(b)) or a Java program (Figure 6-1(c)).

The generated ARFF file contains only two attributes. A text attribute that represents the contents of a whole file, and a class attribute whose values are the 10 documents genre (Figure 6-2). Each line under the *@Data* section represents an HTML file as the first attribute and the class it belongs to, as the second attribute.

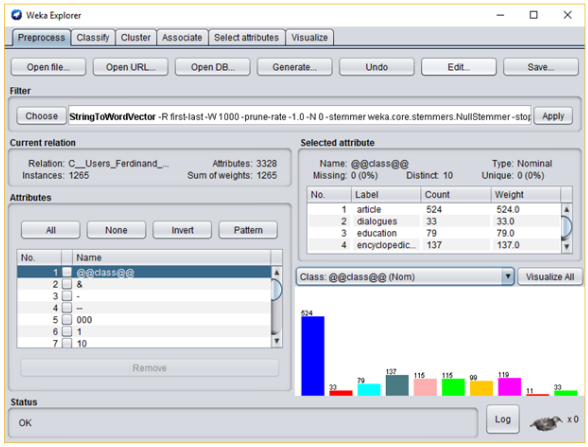
```
@relation HTML-Training

@attribute text string
@attribute class-Html {article,dialogues,education,encyclopedic-long,encyclopedic-short,forum-post,organization,scientific,

@data

'<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\" \"http://www.w3.org/TR/html4/loose.dtd\"><!-- 0.3.39 -->\
'<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\" \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
'<!DOCTYPE html>\n<html lang=\"en\" class=\"webkit chrome cursor\">\n<head>\n\n\n<script type=\"text/javascript\" src=\"/st
'<!DOCTYPE html>\n<html lang=\"en\" dir=\"ltr\" class=\"client-nojs\">\n<head>\n\n\n<script type=\"text/javascript\" src=\'
```

Figure 6-2: Overview of an ARFF file obtained using the StringToWordVector filter

In the same way, the others datasets Auto.arff and Manual.arff, respectively are built for the automatically extracted and the manually extracted content of the HTML files. WEKA's explorer interface allows to have a graphical view of the generated ARFF files, the number of attributes, and the number of instances pro class (Figure 6-3(a)).



(a) Auto.arff file without applying a filter



(b) Auto.arff after applying the StringToWorldVector filter

	Auto.arff	Manual.arff	Html.arff
Instances	1265	1264	1265
Attributes	3328	3902	4234

(c) Number of attributes after applying the StringToWorldVector on each of the datasets

Figure 6-3: Graphical overview of a generated dataset in Weka

Many classifiers cannot be trained on the datasets since they have only one attribute (the *Text* attribute). WEKA proposes the StringToWordVector filter (STWV) which generates string attributes from a text attribute (*Weka.filters.unsupervised.attribute.StringToWordVector*). This filter is applied with the default settings on the Auto.Arff dataset and as a result, more than 3000 attributes were obtained (Figure

6-3(b)). Figure 6-3 (c) shows the number of attributes for each dataset. With the datasets in the ARFF format, many classifiers can be trained and the one which achieves a better accuracy is then selected.

6.2 Choice of the Classifier and the Dataset

WEKA proposes a large number of classifiers that use different approaches to classify a new instance. Since there were 10 documents genres in the corpus used, a multi-class classifier is selected. From some related papers to the text classification, the most used classifiers have been selected and their accuracy is performed on the selected datasets. The chosen classifiers belong to five different groups in WEKA: *Bayes (Naive Bayes), function (Logistic, SVM), Lazy (IBk), Rules (JRip), and Tree (J48)*.

- NiveBayes (Edu, John, & Edu, 1995) is a simple probabilistic classifier based on applying Bayes' theorem $Posterior = \frac{Prior * Likelihood}{evidence}$ with strong independence assumption between the features. An attribute does not have any influence on the others. An advantage of this classifier is that it needs few training set to estimate good parameters for the classification.
- Logistic (Society, 2017): This classifier generalizes the logistic regression to multiclass problems. Logistic regression measures the relationship between the categorical dependent variable (classes) and one or more independent variables (attributes) by estimating probabilities using a logistic function (Wikipedia).
- Support Vector Machine (SVM) (Platt, 1998): SVM is a generalization of linear classifiers and can be used both in a regression as in a discrimination task. It will be described in this chapter later.
- Instance-based learning (IBk) (Aha, Kibler, & Albert, 1991): NBk, which belongs to the family of instance based classifiers has been used in the experiments. A new instance is classified by comparing it with the instances presented in the training set. It becomes the majority class of its neighbors. *K* is the number of neighbors to be considered ($k = 3$ has been used in the experiments)
- JRip (Cohen, 1995) is a propositional rule learner. An instance is classified into a document genre only if it fulfills at least one of the generated rules for this class.
- J48 (Ross, 1993): it is the implementation of the C4.5. The training dataset is trained to build a decision tree. To classify a new instance, it is at first compared with the root of the tree. The result of this comparison determines the branch of the tree to follow. This is repeated for each node until finding a leaf. The instance becomes the class value of this leaf.

The tables below give the results after applying the StringToWordVector filter to generate string attributes and then use the AttributeSelection filter on the three datasets (Auto.arff, Manual.arff, and Html.arff). The AttributeSelection filter allows selecting attributes that fulfill a given criterion. It has been set with the followings configuration: Evaluator = InfoGainAttributEval, Search method = Ranker with Threshold = 0.

	SVM	NaiveBayes	NBk	J48	Logistic	JRIP
StringToWordVector (2570 attr)	**49.17 %**	35.34 %	34.62 %	38.26 %	---	45.61 %
AttributeSelection (972 attr)	**48.46 %**	30.751 %	34.23 %	36.21 %	---	46.25 %

Table 6-1: Performance of some classifiers on the Auto.arff dataset

	SVM	NaiveBayes	NBk	J48	Logistic	JRIP
StringToWordVector (2933 attr)	34.97 %	14.08 %	27.69 %	23.50 %	---	**40.82 %**
AttributeSelection (2 attr)	**41.38 %**	41.30 %	41.22 %	41.22 %	41.22 %	**41.38 %**

Table 6-2: Performance of some classifiers on the Manual.arff dataset

	SVM	NaiveBayes	NBk	J48	Logistic	JRIP
StringToWordVector (3422 attr)	**54.55 %**	45.06 %	50.99 %	48.30 %	---	51.54 %
AttributeSelection (2037)	**54.87 %**	42.6087 %	50.75 %	47.67 %	---	53.12 %

Table 6-3: Performance of some classifiers on the Html.arff dataset

The AttributeSelection filter does not improve the results, and with the same Threshold (t = 0), only two attributes were retained for the dataset Manual.html. The table shows that the logistic classifier does not support a large number of attributes and that the best results were achieved with SVM on the Auto.arff and Html.arff datasets.

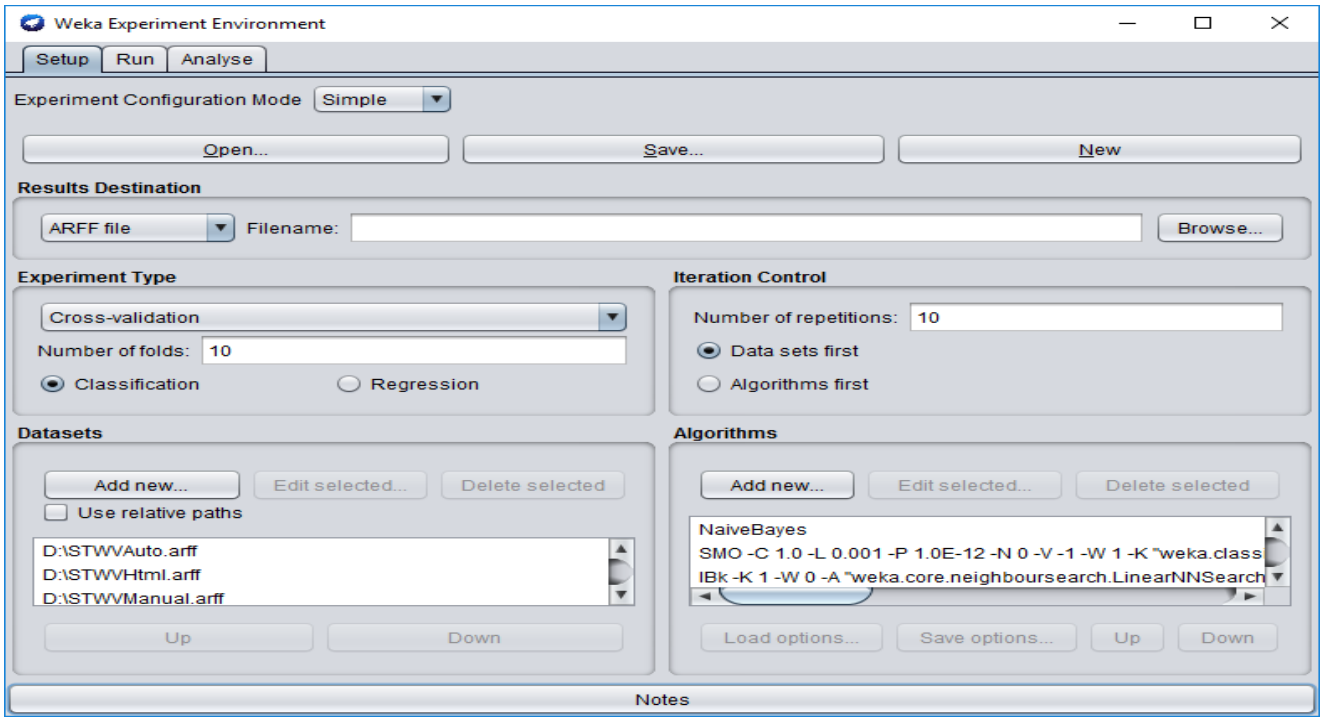


Figure 6-4: Selection of the best classifier using the experiment environment in WEKA

The experimenter panel in WEKA (Figure 6-4) can also be used to evaluate many classifiers on many datasets simultaneously. To do that, open Weka and choose the *Experimenter* panel. On the Tab *Setup,* Click on *New* and upload your datasets in the panel *Datasets*. In the picture, the three datasets Auto.arff, Manual.arff, and Html.arff have been upload. The next step is to choose the classifiers that are going to be compared (*Naive Bayes, SVM, NBk, Jrip, PART, J48)*) in the panel *Algorithms*. There exist many other options like the number of folds, the classification or the regression type of experiment, etc. The comparison can be started by clicking the *start* button on the *Run* tab. At the end of this process, jump to the *Analyse* tab, click on the *Experiment* button and then on the *perform test* button in the panel *Actions*. Figure 6-5 shows the results of the comparison.

Dataset	(1) bayes.Na |	(2) funct	(3) lazy.	(4) rules	(5) rules	(6) trees
Auto	39.24 |	50.24 v	36.39	46.78 v	38.50	41.54
Html	51.48 |	60.68 v	54.82 v	52.22	48.38	49.60
Manual	39.93 |	48.85 v	33.69 *	44.87 v	36.36	38.54
	(v/ /*) |	(3/0/0)	(1/1/1)	(2/1/0)	(0/3/0)	(0/3/0)

Key:
(1) bayes.NaiveBayes '' 5995231201785697655
(2) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.Pd
(3) lazy.IBk '-K 3 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.Euclid
(4) rules.JRip '-F 3 -N 2.0 -O 2 -S 1' -6589312996832147161
(5) rules.PART '-M 2 -C 0.25 -Q 1' 8121455039782598361
(6) trees.J48 '-C 0.25 -M 2' -217733168393644444

Figure 6-5: Performance of the classifiers o the datasets using the experiment environment in WEKA

The accuracies are higher on the picture since some defaults parameters of the StringToWordVector filter were changed. The best results are achieved with the HTML dataset, and it is still the SVM classifier that performs the higher accuracy. For this reason, the dataset Html.arff is used for the generation of the model, which is used to classify new instances with the SVM classifier. It was predictable that SVM would produce better results compared to other classifiers because, in other works, that was also the case in many others works.

6.3 The SVM Classifier

Support Vector Machine (SVM) are supervised learning models with learning algorithms associated, analyzing the data used for classification and regression analysis. For a two-class classification problem, an SVM algorithm uses a training dataset to generate a model, and this model is further used to classify new instances in one of the two classes. SVM is basically a linear classifier. That means given a d-dimensional vector; one can separate them using (d-1)-dimensional hyperplane (Figure 6-6). The best hyperplane is the one who has a maximal distance (margin) to the two classes.

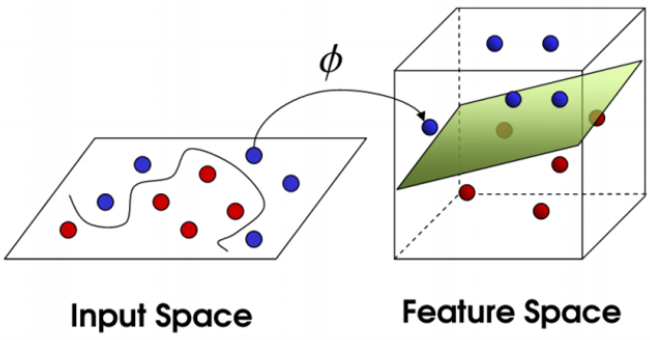


Figure 6-6: Transformation of the input space into feature space (source[17])

This picture shows how SVM can transform a linear non-separable problem into a linearly separable problem by changing the dimensionality of the feature space. It is often said that SVM generalizes the classifiers for linear separable problems.

For a point x_i that belongs to the class u_i, the following system is obtained:

$$\begin{cases} w^t x_i + w_0 > 0 \; if \; u_i = 1 \\ w^t x_i + w_0 < 0 \; if \; u_i = -1 \end{cases} \quad (1)$$

[17] https://perso.telecom-paristech.fr/~clemenco/Course_M2MO_files/M2MO_Seance5.pdf (page 251)

The hyperplane has the following equation $h(x) = w^t x + w_0$ and the distance for a point to the hyperplane is $\frac{h(x)}{||w||}$. To get the largest marge as possible between the hyperplane and the vector support, the following equation has to be minimized: $\frac{1}{2}||w||^2$ with the constraint $u_i(w^t x_i + w_0) > 1$ for $i = 1 \ldots m$. This equation is difficult to solve and can be transformed using the Lagrange theorem to get to the following:

$$h(x) = w^t x + w_0^* = \sum_{i=1}^{m} \alpha_i^* . u_i(x . x_i) + w_0^* \textbf{ (2)}.$$

α_i^* are Lagrange multipliers and w_0^* are solution for the system (1).

When the classes are linearly nonseparable, then the constraints have to be relaxed.

The constraint $u_i(w^t x_i + w_0) > 1$ becomes $u_i(w^t x_i + w_0) > 1 - \varepsilon_i$ **(3),**

and the new function to minimize is: $\frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \varepsilon_i \; with \; C > 0$ **(4)**.

The separation method presented in (2) is limited, since the scalar $(x . x_i)$ must be a point of a hyperplane. The introduction of the kernel function solve this limitation:
In the equation $h(x) = w^t x + w_0^* = \sum_{i=1}^{m} \alpha_i^* . u_i(x . x_i) + w_0^*$, the scalar product $(x . x_i)$ can be replaced by any kernel function $K(x . x_i)$ provided that $K(x . x_i)$ performs a scalar product.

The new function is then: $h(x) = w^t x + w_0^* = \sum_{i=1}^{m} \alpha_i^* . u_i . K(x . x_i) + w_0^*$ **(5)**.

There exist many kernel functions. Among them:

- Linear kernel: $K(x . x_i) = x . x_i$
- Radial Basis Function (RBF): $K(x . x_i) = \exp(-\gamma \left| |x - x_i| \right|^2)$
- Polynomial kernel: $K(x . x_i) = (x . x_i + c)^2$
- Sigmoidal kernel: $K(x . x_i) = \tanh(x . x_i + c)$

In Weka, following kernel functions are implemented: *RBFKernel* (Radial Basic Function), StringKernel, *NormalizedPolyKernel, PrecomputedKernelMatrixKernel, Puk* (the Pearson VII function-based universal Kernel).In Figure 6-7 the linear kernel is compared with the RBF kernel on two datasets.

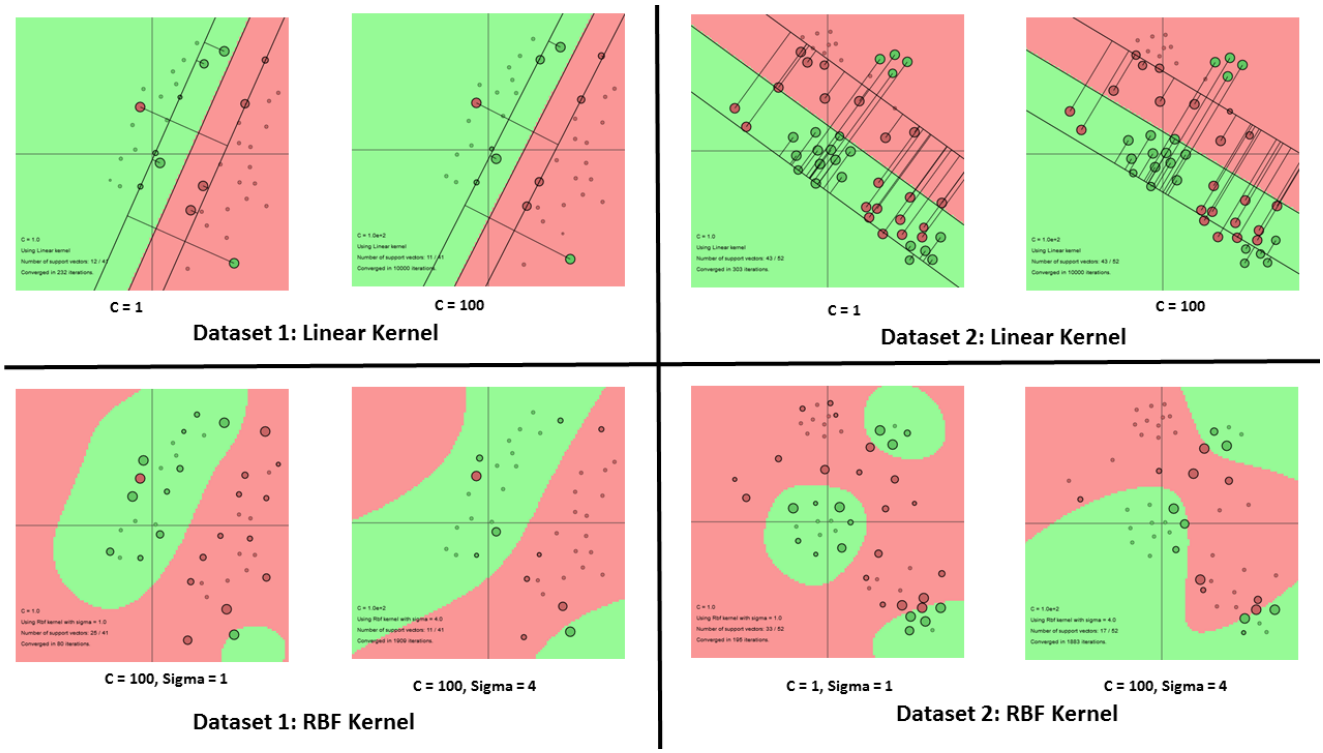


Figure 6-7: Impact of the SVM kernel on the datasets

On these two datasets, the RBF kernel performs as well as the linear kernel. This example illustrates the flexibility of SVM to train data whose scatter plots are random. Parameter C is a complex parameter, and sigma is another parameter to the RBF kernel. Getting the best parameters for the training datasets is difficult to find because of the high dimensionality. The linear kernel has been used in the experiments. For 2-class problems, experiments in many papers reveal that the RBF kernel performs well on the datasets as the others kernel functions. SVM has been developed for binary classification problems. However, it is extended to multiclass problems using some methods as 1-against-all, 1-against-1, etc. All of these properties make the SVM a good candidate for text categorization.

(Joachims, 1998) explore the SVM classifier and gives some arguments, why it should work for the text categorization problems:

- **High dimensional input space:** By the text categorization, many string attributes are generated from the documents. SVM can support more than 10000 features.
- **Too few features are irrelevant:** based on some experiments, (Joachims, 1998) concludes that also the features ranked lowest still contain useful information and are for this reason relevant. He concludes that a good classifier should support a large number of feature and that discarding some of them for example with the attribute selection filter may result in a loss of information.
- **Robustness:** on all experiments, it performed good results in the text classification problem.
- **Text classification problems are often linearly separable:** The default kernel function of SVM is linear. Its kernel can be changed to transform a nonlinearly separable problem into a linearly separable problem.

6.4 Classification Results on the Original Datasets

The data preprocessing using the StringToWordsVector filter (STWV) has a great influence on the training data. Its path in WEKA is *Weka.filters.unsupervised.attribute.StringToWordVector*[18]. As explained in the documentation, it is a filter that converts a text into string attributes. The tokens that can delimit a word can be set in its configuration. Each generated word constitutes an attribute and its value is its number of occurrences. In the default configuration, for example, this filter will keep the thousands most frequent words for each file. The parameters are verified to find those which lead to a better accuracy on the datasets using the SVM classifier. The best results were obtained with the following settings:

Parameter	Value	Property
IDFTransform	true	If this parameter is set as true, then the word frequencies in a document have to be transformed into w_{pq}*log(num of Docs/num of Docs with word *p*) where w_{pq} is the frequency of word *p* in document (instance) *q*. (see the StringToWordVector filter documentation)
TFTransform	true	Sets that the word frequencies have to be transformed using the rule log(1+w_{pq}) where w_{pq} is the frequency of word *p* in document (instance) *q*. (see the StringToWordVector filter documentation)
OutputWordCounts	true	Set if the occurrences of a word should be counted off or if only his presence should be reported using a Boolean variable.
Stemmer	snowballStemmer	A wrapper class for the Snowball stemmers (http://weka.wikispaces.com/Stemmers)
StopwordsHandler	Rainbow	Stopwords list based on Rainbow (shttp://www.cs.cmu.edu/~mccallum/bow/rainbow)
Tokenizer (wordTokenizer)	= \n \r \t . , ; : ' " ( ) ? ! / - _ > < & #	Split words by these characters
WordsToKeep	4000	Number of words to keep after tokenization

Table 6-4: The best parameters for the StringToWordVector vector on the Html.arff dataset

This configuration is applied to the ARFF files. STWV extracts the 4000 most significant words from each file I according to others settings and merges them all to form the set of attributes. Table 6-5 shows the impact of the number of words to keep per file on the accuracy of the SVM classifier. Each cell contains the number of attributes and the achieved accuracy. It is on this basis that 4000 words were chosen for the continuation of the experiments on the HTML.arff dataset. A better accuracy has not be achieved. By observing the confusion matrix, a large number of files are mistakenly classified into the *article* class (Figure 6-8). It is observed for example that out of 99 files of the genre *organization*, only 33 are well classified, and among 66 files that are misclassified, 49 have been attributed to the *article* class. The article class is the largest class in the corpus with 41% of the files. Therefore, this class has a high weight and may influence the categorization of certain similar files to the article genre.

Words to keep	**500**	**1000**	**3000**	**4000**	**5000**
Auto.arff	1876, 51.86 %	3156, 51.70 %	**9181, 53.99 %**	16461, 51.07 %	16351, 52.49 %
Manual.arff	2273, 48.22 %	3398, 49.17 %	**9486, 52.25 %**	16164, 48.62 %	17608, 51.38 %
Html.arff	1923, 57.31 %	4279, 57.71 %	14213, 59.84 %	**19896, 61.59 %**	25841, 60.63%

Table 6-5: Impact of number of words to keep on the accuracy. The number of attributes and the archived accuracy is reported in the cells

[18] http://weka.sourceforge.net/doc.stable/weka/filters/unsupervised/attribute/StringToWordVector.html

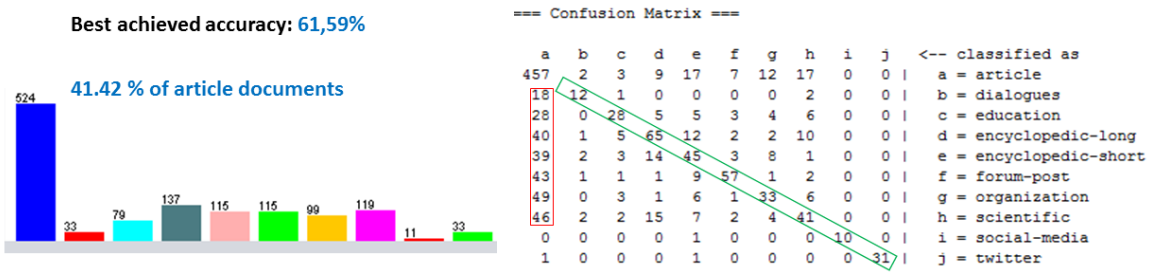


Figure 6-8: Confusion matrix by the training

In case only two group of document genres for the corpus are defined: *Genre1* made up of HTML files of the *article* class (41.42%), and *Genre2* made up of the rest of the files (58.58%), an accuracy of 73.28% is obtained. This result shows that the *article* documents genre are not easily separable from other. If the *article* documents genres are removed from the corpus, and the corpus is trained with the nine remaining document genres (with the same STWV configuration), 56.41% of accuracy is obtained. That mind that others document genre are not easy to separate each other and that the high weight of the article class does not influence the classification results very much. The document genres that are readily separable from each other were examined next, and the result are reported in the following table:

	Article	**Dialogue**	**Education**	**Encyclopedic-long**	**Encyclopedic-short**	**Forum**	**Organization**	**Scientific**	**Social-media**	**Twitter**
Article	Achieved Acc. Distribution	96.05 % 94.07 %	90.38 % 86.90 %	87.29 % 79.27 %	86.85 % 82 %	85.91 % 82 %	86.19% 84.11 %	86.62% 81.50 %	99.62 % 97.94%	99.64 % 94.08 %
Dialogue	33.39 %	The amelioration	85.71 % 70.53 %	94.70 % 81.59 %	91.21 % 77.70 %	87.16 % 77.70 %	86.36 % 75 %	86.18 % 78.29 %	97.72 % 75 %	96.96 % 50 %
Education	26.56 %	51.51 %		68.05 % 63.43 %	67.52 % 59.28 %	72.68 % 59.28 %	71.91 % 55.62 %	74.74 % 60.10 %	98.89 % 87.78 %	97.32 % 70.54 %
Encyclopedic-long	38.69 %	**71.21 %**	12.63 %		73.41 % 54.36 %	82.14 % 54.36 %	78.81 % 58.05 %	72.65 % 53.52 %	98.64 % 92.57 %	98.23 % 80.59 %
Encyclopedic-short	26.94 %	**60.58 %**	20.24 %	41.74 %		84.78 % 50. %	71.03 % 53.75 %	85.04 % 50.85 %	97.62 % 91.27 %	97.30 % 77.70 %
Forum	21.72 %	42.42 %	32.91 %	**60.87 %**	**69.56 %**		71.50 % 53.75 %	76.50 % 50.85 %	97.62 % 91.27 %	96.62 % 77.70 %
Organisation	13.09 %	45.44 %	36.71 %	49.49 %	37.36 %	38.38 %		74.31 % 54.59 %	98.18 % 90 %	97.73 % 75 %
Scientific	27.68 &	36.34 %	36.69 %	41.16 %	**69.56 %**	52.19 %	43.43 %		96.92 % 91.54 %	98.03 % 78.29 %
Social-media	**81.55 %**	**90.88 %**	**90.92 %**	**81.70 %**	**72.74 %**	**72.74 %**	**81.80 %**	**63.59 %**		93.18 % 75.0 %
Twitter	**93.92 %**	**93.92 %**	**90.90 %**	**90.88 %**	**87.89 %**	**84.84 %**	**90.92 %**	**90.93 %**	**72.72 %**	

Figure 6-9: Pairwise comparison of the accuracy of the 10 documents genres

At the upper diagonal of the table, there is the distribution of the two classes and the achieved accuracy after the training. Because of lack of space, only the percentage of the largest class is represented. The lower diagonal indicates the improvement achieved by training the two classes. The table can be read as follows: By training the *article* classes (*94.07 %*) and the *dialogue* class (100 % - 94.07 % = *5.98%*), *96.05 %* of accuracy is achieved, this corresponds to an improvement of *33.39%* [(96.05 - 94.07) / (100 - 94.07) %]. If the files are classified as *articles*, then the minimum accuracy of 94.07 %, which corresponds to the percentage of the biggest class, will be reached. The *twitter* and *social media* classes are easily dissociable from the other classes. An improvement of more than 60% is observed by each comparison (in bold). The *organization* and *article* class, for example, are difficult to train pairwise; only an improvement of *13.09 %* has been observed. Few files were examined in details in the corpus to detect some sources of misclassification.

6.4.1 Possible Causes of the Low Accuracy

The web page classification task is not evident even for the human being.

- Some web pages may belong to multiple document genres at the same time. For example, a web page can be classified at the same time in the encyclopedic or in the article category. In this case, two similar web pages that have been categorized differently, make it difficult for classifiers to train them. The classifier would then consider one of them as misclassified. It is, therefore, important to have specific features for each category.
- It has also been noticed that some very long documents were classified as encyclopedic-short. The difference between the encyclopedic-short and encyclopedic-long classes cannot be established strictly from the number of words or the number of sentences. Because the corpus has been manually annotated, the presence of many pictures can influence the classification. A document may appear to be long because it contains many images (*e.g. D03T28/input/2.html, D03T14/input/1.html, D02T24/input/10.html, D02T26/input/6.html, D01T31/input/16.html, etc.)*
- The corpus also includes some empty web pages such as: *D01T06\input\M1.html (scientific), D02T13\input\M12.html (Encyclopedic-long), D01T25\input\M6.html (Encyclopedic-short), D01T33\input\M8.html (Twitter), D01T08\input\M16.html (article), D01T15\input\M13.html (organization)*. Because these empty files belong to different document genres, they constitute noises and must be removed from the corpus or re-downloaded again. It could be that empty web pages occur if a tool is used to download them and that the URL given as parameter is malformed.
- The Twitter class contains comments in several languages: Spanish, French, German, Arabic, Chinese, etc. This feature is exploited by classifiers which use them to dissociate the Twitter web pages from others quickly. It is rather a good feature.

The remarks mentioned above are some possible causes of this average accuracy (61.59 %).
Another way of training the corpus is to generate the dataset by self-defining the attributes. This approach is described in the next paragraph.

6.4.2 Classification Using Self-Created Attributes

The corpus is evaluated with the attributes generated by the STWV of WEKA. It builds attributes from a bag of words. In this paragraph, the process of creating an HTML dataset based on the characteristic of an HTML web page to generate a set of attributes. Datasets such as the iris dataset of WEKA were surveyed, and it is noticed that these datasets contain attributes that are specific to only one of the classes. It is deduced, that there may be words which can characterize each of the 10 documents genres (*article, forum post, microblog, organization, encyclopedic short, encyclopedic long, social media, scientific, education, and dialogues*) individually. For example, the words *Bibliography* and *Abstract* would refer to scientific documents genres. The high frequency of the word *forum* will be higher in the forum document genre as well as the words *twit* for the document genres twitter. It was required to understand how such of words could be generated. Rules based learners and trees based-learners classifiers in WEKA show the rules (or the tree) used to classify the instances into their corresponding classes (For example, the rules used by the rule-based learner *PART* or the tree-based learner *J48* to classify new instances are printed on the classifier output panel of WEKA). Some attributes can be extracted there, and some others can be provided by the HTML tags, to build the set of the features. This method to generate the dataset is not good because many attributes were selected manually (from the rules) and there are specific to the actual dataset. The achieved accuracy was less than 50 % on the corpus with this process.

Another idea is to consider the most frequents words from each document genre in the set of attributes of the datasets. Figure 6-19 shows the more frequents words of the *article* and *twitter* class:

the=53777, of=30791, and=22913, to=21691, in=17698, a=16694, was=8991, The=8941, that=7029, on=6697, for=6472, is=6369, by=6018, as=5323, with=4935, at=4432, from=4080, his=3955, it=3405, be=3380, were=3161, this=2979, an=2938, had=2844, he=2790, not=2761, are=2528, have=2378, -=2369, I=2336, or=2334, =2285, which=2171, In=1943, their=1938, they=1914, but=1908, =1904, &=1813, been=1810, who=1747, has=1660, you=1654, would=1571, one=1546, its=1382, more=1336, all=1331, about=1309, It=1308, her=1294, |=1285, A=1271, your=1248, This=1222, into=1193, also=1161, when=1157, 2016=1143, after=1135, out=1096, 2015=1064, other=1037, first=1035, up=1029, He=984, will=967, over=966, ? =956, History=940, new=937, we=933, 16=932, two=924, only=887, no=887, so=886, May=882, our=879, there=862, time=861, News=857, can=830, some=807, Castle=807, could=805, than=805, years=804, says:=793, most=792, made=787, British=772, she=766, New=740, what=724, them=724, any=717, people=705, said=701, during=697, where=694, On=690, 2014=684, if=681, many=679, John=673, like=661, then=656, –=651, before=650, through=650, under=649, being=647, my=632, him=619, About=614, against=605, Help=605, American=604, Close=603, such=602, between=600, may=590, You=575, April=572, do=561, these=560, just=556, Home=549, 1=547, used=545, because=537, found=534, did=531, National=530, government=525, captures=525, Court=524, us=524, 2017=518, very=507, those=504, 2013=503, right=501, even=501, back=498, December=495, But=489, They=488, now=488, As=487, World=485, South=484, Edward=483

Most frequents words in the article class

Tweet=552, to=314, @=282, Close=277, 1=260, 0=253, ?=220, the=202, Twitter=199, this=195, ·=177, a=173, -=138, Retweet=137, in=126, of=124, é–‰ã?˜ã‚<=120, Ñ?Ðμ=118, Embed=115, 2016=115, account=110, Verified=109, 2=107, ÐiÐ²Ð¸Ñ”Ð°=104, Ð¼Ð¸=104, retweets=103, SchlieÃŸen=100, =97, Copy=96, gusta=94, 40404=93, »=93, you=90, on=88, and=87, link=84, for=84, Pendle=82, your=77, Like=76, 16=75, Retweeted=74, More=74, me=73, Reply=73, Liked=72, Witch=71, Walk=69, Sign=65, GefÃ¤llt=64, up=64, 3,=63, Vodafone,=62, Bahasa=62, English=62, O2=62, mir=62, Indonesia=61, 3=61, more=60, Ñ‚Ð²Ð¸Ñ,=60, 2012.=60, an=59, ©=58, @pendlewitchwalk=56, Help=56, ÐšÐ¾Ð¿Ð¸Ñ€Ð°Ñ˜=54, Ð ÐμÑ‚Ð²Ð¸Ñ‚Ñ˜=54, è ª?è¨¼æ¸ˆã?¿ã‚¢ã‚«ã‚¦ãƒ³ãƒˆ=54, with=54, likes=54, Ñ‚Ð²Ð¸Ñ‚Ð°=54, Ð£Ð³Ñ€Ð°Ð´Ð¸=53, MÃis=53, Mar=53, is=53, ÐˆÐ¾Ñˆ=53, Ð»Ð¸Ñ‡Ð½Ð¾=53, Ð ÐμÑ‚Ð²Ð¸Ñ‚Ð¾Ð²Ð°Ð¾Ñ‡Ð½Ð¾=52, ÐžÐ´Ð³Ð¾Ð²Ð¾Ñ€=52, del=52, May=51, location=50, Du=50, list=49, Insertar=49, nlee69=49, Me=49, account?=48, Copiar=48, 2014=48, enlace=48, Retwittear=48, Ñ?Ð²Ð¸Ñ‚Ð°Ñšð°=47, Responder=46, Mehr=46, Gustado=46, Retwitteado=46, Account=45, Verifizierter=45, 2015=45, zu=45, Twitter?=45, from=44, Cerrar=42, by=42, Tweets=42, can=41, kopieren=41, Learn=41, zum=41, de=41, was=40, Apr=40, Have=39, Cookies=39, Cancel=38, tweet=38, The=37, Retweeten=36, searches=36, Saved=36, below.=36, Remove=36, Log=36, users=36, Link=36, Suggested=36, Video=36, Add=36, captures=33, 2017=33, be=33, Privacy=33, @asiance=32,

Most frequents words in the twitter class

Figure 6-10: Most frequents words in the article and twitter class

The most frequent words for the article class are regular words such as articles, prepositions, pronouns, and verbs. In all, 439660 different words for the article class have been enumerated in the corpus. They cannot be considered all in the set of attributes in WEKA, especially that some others new words may come from others (e.g. many numbers appear and many words from others languages in the *twitter* class). If among them, only the most frequent words are selected, then those which appear rarely and which can be very important to distinguish some classes from others, are omitted. The fact of selecting the words according to their frequencies in a file or in a corpus are already implemented in the TF-IDFTransform functions that have been used as a configuration parameter in the StringToWordVector filter. There is no need to re-implement it here anymore. If a set of attributes, only based on some words appearing in the corpus has to be built, it is preferable to use the STWV of KA which takes into account the frequency of a word in each file and each document genre of a corpus.

Many types of research on web page classification using the self-selected attributes have to be done and (Vidulin, 2007) for example proposes to build attributes from some URL features, HTML features, and test features. Figure 6-11, Figure 6-12, and Figure 6-13 present the features he proposes.

Feature
Average number of characters per word
Average number of words per sentence
Number of characters in hyperlink text / Total number of characters
Number of alphabetical tokens (alphabetical token is a sequence of letters) / Total number of tokens
Number of numerical tokens (numerical token is a sequence of digits) / Total number of tokens
Number of separating tokens (separating token is a sequence of separator characters (space, return. . .)) / Total number of tokens
Number of symbolic tokens (symbolic token is a sequence of characters excluding alphanumeric and separator characters) / Total number of tokens
Number of content words / Total number of content words for 321 content words (stemmed by Porter stemming algorithm)
Number of function words / Total number of function words for 50 most common function words in the corpus
Number of punctuation symbols / Total number of punctuation symbols for 34 punctuation symbols
Number of declarative sentences / Total number of sentences
Number of interrogative sentences / Total number of sentences
Number of exclamatory sentences / Total number of sentences
Number of other sentences (in most cases list items) / Total number of sentences
Number of date named entities / Total number of words
Number of location named entities / Total number of words
Number of person named entities / Total number of words

Figure 6-11: A set of text features (Vidulin, 2007)

Feature	Description
Https	Indicates whether the scheme is https.
URL depth	Number of directories included in the path.
Document type	Described by four Boolean features, each indicating whether the document type is *static HTML* (document extensions html and htm), *script* (document extensions asp, aspx, php, jsp, cfm, cgi, shtml, jhtml and pl), *doc* (document extensions pdf, doc, ppt and txt) or *other* (the other document extensions).
Tilde	Appearance of "/~" in the URL.
Top-level domain	Described by ten Boolean features, each indicating whether the top-level domain is *com, org, edu, net, gov, biz, info, name, mil* or *int*.
National domain	Indicates whether the top level domain is a national one.
WWW	Indicates if the authority starts with www.
Year	Indicates the appearance of year in the URL.
Query	Indicates the appearance of query in the URL.
Fragment	Indicates the appearance of fragment in the URL.
Appearance of 54 most commonly used words in URL	Indicates the appearance of common content words in URL: *about, abstract, adult, archiv, articl, blog, book, content, default, detail, download, ebai, english, error, fanfic, faq, forum, free, fun, funni, galleri, game, help, home, index, joke, kid, legal, librari, link, list, lyric, main, member, music, new, paper, person, poem, poetri, product, project, prose, pub, public, quiz, rule, search, port, stori, stopic, tripod, user, wallpap*

Figure 6-12: A set of URL features (Vidulin, 2007)

Feature
Number of hyperlinks to the same domain / Total number of hyperlinks
Number of hyperlinks to a different domain / Total number of hyperlinks

Number of tags / Total number of tags for 5 tag groups:
1. **Text Formatting** – *<abbr>, <acronym>, <address>, <b>, <basefont>, <bdo>, <big>, <blockquote>, , <cite>, <code>, <del>, <dfn>, , <font>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <i>, <ins>, <kbd>, <pre>, <q>, <s>, <samp>, <small>, <strike>, , <style>, <sub>, <sup>, <tt>, <u>, <var>*
2. **Document Structure** – *
, <caption>, <col>, <colgroup>, <dd>, <dir>, , <dl>, <dt>, <frame>, <hr>, <iframe>,   , <menu>, <noframes>,   , <p>, , <table>, <tbody>, <td>, <tfoot>, <th>, <thead>, <tr>,   *
3. **Inclusion of external objects** - *<applet>, <img>, <object>, <param>, <script>, <noscript>*
4. **Interaction** – *<button>, <fieldset>, <form>, <input>, <isindex>, <label>, <legend>, <optgroup>, <option>, <select>, <textarea>*
5. **Navigation** – *Counting href attribute of tags <a>, <area>, <link> and <base>*

Figure 6-13: A set of HTML features (Vidulin, 2007)

URL features: The URL associated with each HTML file in the corpus must be found in order to analyze it. Attributes that can be derived from an URL are presented in Figure 6-12.
HTML features (Figure 6-13)**:** the first two propositions (the domain of the hyperlinks) have been omitted in the set of attributes taken into account in the implementation.
Text feature: This feature is the most difficult feature to exploit. In fact, one should only consider text contained in HTML files. Unfortunately, web pages are saturated with tags, scripts that can be in the header or the body of a file. To have only the text content of a file, the Jsoup API can be used. Scripts are removed (JavaScript (JS), cascade style sheet (CSS)) and now each point (punctuation) can be considered as the marker of a declarative sentence. It is, therefore, possible to evaluate the percentage of declarative, interrogative, and exclamatory sentences in the text. One can also analyze the average characters per word, the average words per sentence and the average number of the sentences per file.
Many proposed attributes of the text feature such as the occurrence of each function word compared to the most common function words in the corpus, have not be implemented because of the high execution time (Figure 6-11) ". One reason is the very long execution time on an Intel core i3 3GHz and 4 GB RAM. The *DKPro pipeline tool*[19] has been used for extracting the function words. As seen in their implementation code in Figure 6-14, the method to generate the part of speech tags take each file individually in input and run a pipeline of functions. It needs more than a minute to process each file. As a consequence, it will take more than two days to train the whole corpus (The newly downloaded files from the named entities are more than 9000 and will take three days by the classification with this tool).

```
public static void getPOS(String file) throws ResourceInitializationException, UIMAException, IOException{
        runPipeline(
                createReaderDescription(TextReader.class,
                        TextReader.PARAM_SOURCE_LOCATION, file,
                        TextReader.PARAM_LANGUAGE, "en"),
                createEngineDescription(OpenNlpSegmenter.class),
                createEngineDescription(OpenNlpPosTagger.class),
                createEngineDescription(LanguageToolLemmatizer.class),
                createEngineDescription(MaltParser.class),
                createEngineDescription(Conll2006Writer.class,
                        Conll2006Writer.PARAM_TARGET_LOCATION, "."));
}
```

Figure 6-14: POS extraction using the DKPro tool (Source in footnote)

[19] https://dkpro.github.io/dkpro-core/pages/java-intro/

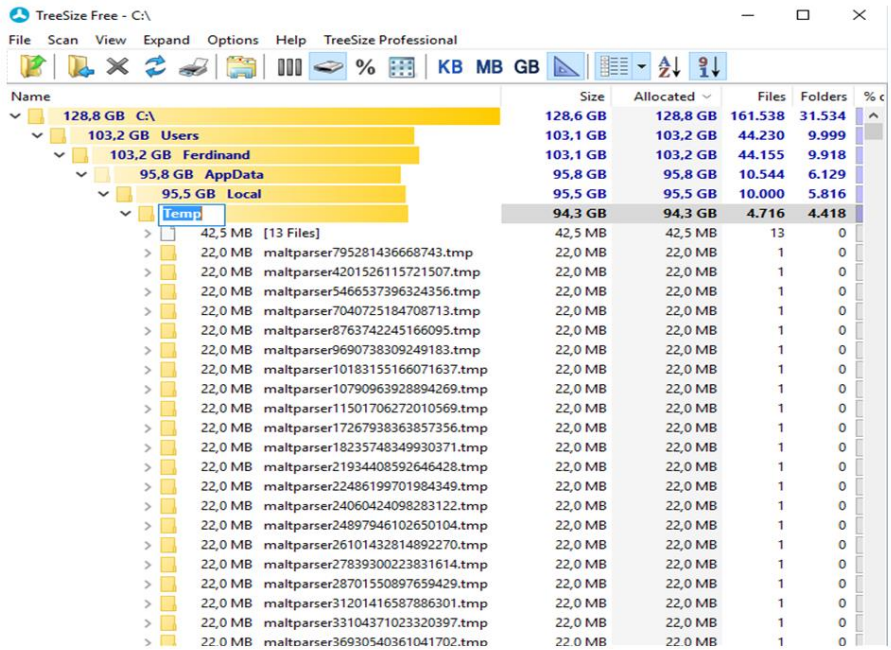


Figure 6-15: Hard disk usage of the DKPro tool

Another Drawback of this tool is that it saturates the hard disk, which can lead to a program shutdown when the partition is full (Figure 6-15). During the processing, many gigabytes of data are stored in the temporary folder and must be deleted after each run. The program is interrupted if this folder is emptied before it runs out.

Due to the constraints of the execution time and the hard drive's occupancy, it was not possible to perform several runs after a new set of attributes has been added. The accuracy of 55.02% has been achieved by only considering the list of the most frequent words in the URLs, the percentage of some function words as attributes, and the percentage of the sentence types (Figure 6-16):

Some text attributes: numberOfWords, numberOfSentence, percentDeclSent, percentInterSent, percentExclSent, avgNumCharWord, avgNumWordSentence.

Some frequent word in a URL (Vidulin, 2007): about, abstract, account, archiv, articl, biblio, blog, book, categori, close, comment, content, copyright, default, description, detail, dialog, download, education, encyclop, english, faq, forum, free, galleri, game, help, home, index, joke, kid, legal, log, librari, link, list, litterat, main, member, menu, new, Organization, paper, person, poem, poetri, private, product, project, pub, public, quiz, reply, rule, search, port, scientif, share, social, stori, stopic, subscribe, topic, tweet, url, user, wallpap.

Function words: nn, expl, csubjpass, rcmod, predet, cop, aux, conj, num, preconj, dep, xcomp, appos, advmod, neg, number, det, ccomp, partmod, possessive, rel, prep, advcl, punct, nsubj, iobj, cc, csubj, nsubjpass, auxpass, pcomp, purpcl, parataxis, infmod, poss, amod, complm, pobj, prt, measure, acomp, quantmod, tmod, abbrev, dobj, mark.

Figure 6-16: the function words attributes obtain from the corpus using the DKPro tool.

If these attributes are supplemented with other text characteristics such as the frequency of certain named entities (Miscellaneous, Time, Date, Percent, Money, Organization, Location, Person) per file, the nature of the neighbor web page (page referenced in the file), the percentage of external links and all other features (text, URL and text) presented above, a better accuracy than what has been obtained so far using the STWV filter can be reached.

6.4.3 Document Genre Classification on Others Datasets

Others datasets from WEKA and on the internet have been carried out, to test the performances of the used configurations on others datasets:

- ReutersCorn [20](WEKA dataset): 2234 attributes, 2 classes, **98.51%** accuracy.
- ReutersGrain [21](WEKA dataset): 1485attributes, 2 classes, **98%** accuracy.
- SMS[22] (Almeida, Hidalgo, & Yamakami, 2011): 1385 attributes, 2 classes, **90%** accuracy.
- 0h10.mat[23]: 3239 attributes, 10 classes, **74.85%** accuracy.
- C50 (M. Lichman, 2013): 10860 attributes, 50 classes, **78.24%** accuracy.

The data 0h10.mat has the same number of classes as the core corpus with substantially the same number of attributes; it nevertheless provides better results. The dataset C50 has 50 classes and 50 instances per class. Despite the high number of classes, the accuracy is better than on the core corpus. These experiments show that with a high number of classes, one can always obtain a good accuracy and that the dataset (HTML.arff) generated from the corpus is difficult to train.

[20] http://www.cs.waikato.ac.nz/ml/weka/datasets.html

[21] http://www.cs.waikato.ac.nz/ml/weka/datasets.html

[22] http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/

[23] http://tunedit.org/repo/Data/Text-wc

6.5 Classification of Newly Downloaded Web Page

After new web pages have been downloaded with the Bing API, they must be classified into the corresponding document genres. The first step is to train several classifiers on the corpus in order to select the best. Above, it has been shown that SVM works better on the corpus and that some default parameters of the StringToWordVector filter need to be modified. These configurations have been used to classify the new files. The second step is to classify newly downloaded HTML files. Several challenges are to be overcome at this level:
The class attribute in the test set and the training set should be identical. This can be done by creating a similar folder hierarchy for the test set and the training set.

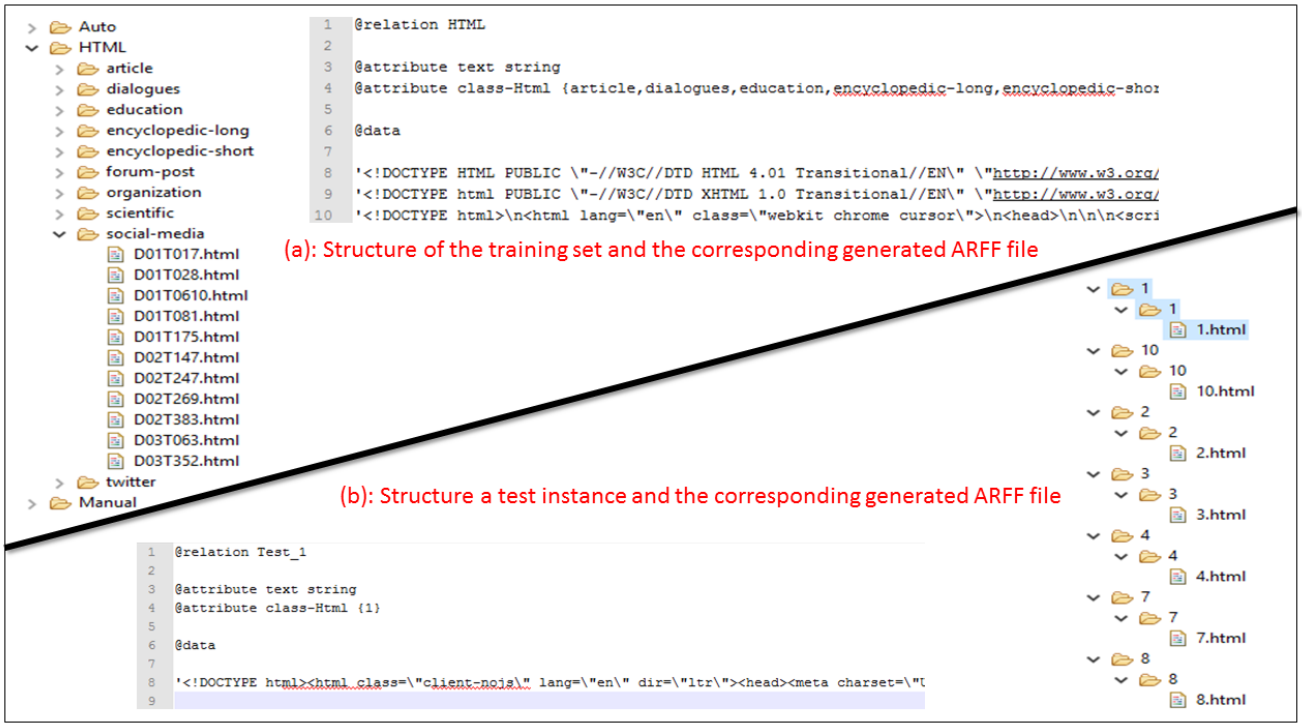


Figure 6-17: Classification of new instances

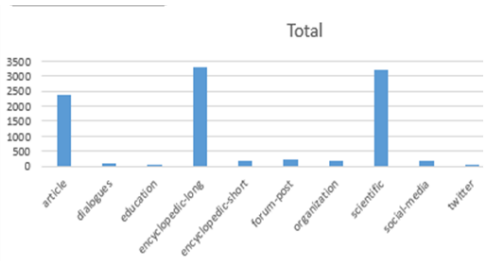
Weka generates the class attribute by considering all the folder names in the grandparent folder (Figure 6-17(a)). This implies that the downloaded files must be classified into folders of the same hierarchy as in Figure 6-17(a). Each newly downloaded file has been classified separately. This implies that an ARFF file is to be generated for each downloaded HTML file. This can be achieved if each file is structured like in Figure 6-17(b) to enable generating the test instance though the Java class TextDirectoryLoader. The problem with the structure in Figure 6-17(b) is that the generate ARFF file will contain only one value for the class attribute (*@attribute class-Html {1}*). It is possible to modify the attribute class of an ARFF file within a Java program by using the method *Weka.core.Instance.insertAttributesAt (trainingSet.classAttribute (), instance.numAttributes ())*. Each test instance attribute class has been replaced by this way by the attributes class of the training set (*@attribute class-Html {article, dialogues, education, encyclopedic-long, encyclopedic-short, forum-post, organization, scientific, social-media, twitter*}). Another problem is that a test instance has new attributes than those present in the training set. This can be solved by using a filtered classifier.

When requesting 10 URLs for each information nugget

Figure 6-18(a) shows the class distribution of the new instances after classification. As mentioned earlier, a web page can be downloaded many times for example if it is retrieved by different information nuggets. The colored files in Figure 6-18 (b) are the same file. The line "*D01T01\Output\1\4.html encyclopedic-*

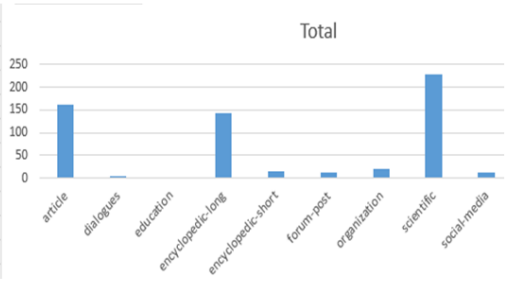
long" can be interpreted as: to the *information nugget 1* has been assigning the file *4.html* which has been classified as *encyclopedic-long*. This information nugget has been extracted from the summary file of the *topic 1 (T01)* of the *domain 1 (D01)*.

D01T01\Output\1\10.html	scientific	article	2386
D01T01\Output\1\2.html	scientific	dialogues	71
D01T01\Output\1\3.html	article	education	4
D01T01\Output\1\4.html	encyclopedic-long	encyclopedic-long	3295
D01T01\Output\1\7.html	scientific	encyclopedic-short	192
D01T01\Output\1\8.html	scientific	forum-post	213
D01T01\Output\10\1.html	encyclopedic-long	organization	186
D01T01\Output\10\10.html	article	scientific	3198
D01T01\Output\10\2.html	scientific	social-media	172
D01T01\Output\10\3.html	encyclopedic-long	twitter	5
D01T01\Output\10\4.html	scientific	(vide)	
D01T01\Output\10\5.html	article	**Total général**	**9722**



(a) Class distribution of the downloaded files (many files are redundant)

D01T01\Output\8\2.html	organization	article	161
D01T01\Output\15\9.html	forum-post	dialogues	4
D01T01\Output\7\9.html	article	education	1
D01T01\Output\15\8.html	scientific	encyclopedic-long	143
D01T01\Output\20\7.html	organization	encyclopedic-short	16
D01T01\Output\5\8.html	article	forum-post	13
D01T02\Output\11\6.html	encyclopedic-short	organization	21
D01T02\Output\9\10.html	encyclopedic-long	scientific	227
D01T02\Output\9\7.html	encyclopedic-long	social-media	11
D01T02\Output\9\9.html	encyclopedic-long	(vide)	
D01T02\Output\11\7.html	encyclopedic-long	**Total général**	**597**



(b) Class distribution after selecting one file per information nugget.

Figure 6-18: Document genre distribution of the downloaded web page

As one can see, there is a large number of documents belonging to the *article*, *scientific*, and *encyclopedic-long* classes. *Dialogue* and *twitter* classes appear very rarely. For instance, only four occurrences of the *dialogue* class (with the possibility that the same file is duplicated –downloaded from different nuggets) are presents in Figure 6-18. In conclusion, for many topics, dedicated *twitter* web pages and *dialogue* web pages are very rare, or if they exist, then they may have a low ranking (they do not appear in the first requested URLs).
After selecting for each information nugget a unique HTML file, the distribution in Figure 6-18(b) is obtained. One can remark that there is no file for the *Twitter* class and only one file for the class *dialogue*.

The files assignment to the information nuggets has been implemented using a chained hash table as shown in Figure 6-19: the documents genres are sorted in ascending order of their appearance in the corpus (e.g. G2-G3-G5-G4-G1). There are five nuggets (N1... N5) and 5 documents genres (G1... G5). The process starts by looking for the document genre with the lowest occurrence (G2), it is found in the node N1. To the information nugget whose node corresponds to N1 is assigned a file of class G2 N1(G2). The node N1 is deleted from the chained hash table, and the search continues by the next document genre with the lowest occurrence G3. It cannot be found because the node N1 has already been deleted (Maximum a page must be assigned to an information nugget that is the reason a node must be deleted after a file in its chain has been selected). This explains the absence of the *twitter* class in Figure 6-19(b)). As conclusion, there will be no files with the document genre G3 for this topic. The search follows respectively with the document genres G5, G4, and G1 and the following results N2(G5), N3(G4) and N5(G1) are got. Nx(Gy) means that to the information nugget whose node correspond to Nx, is assigned a file whose the document genre is Gy.

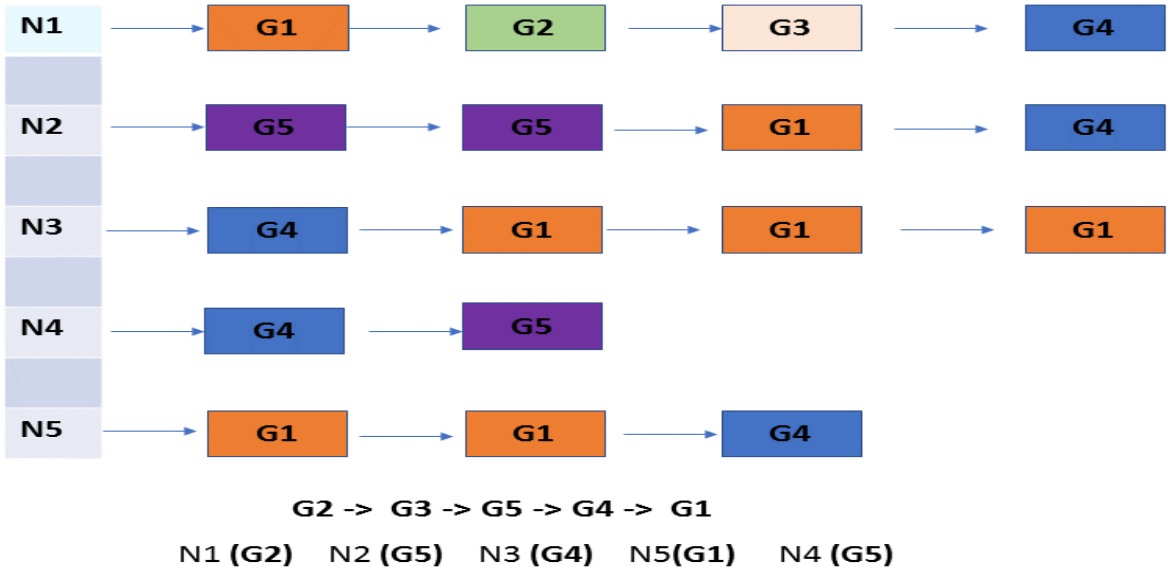


Figure 6-19: Process of file selection for each information nugget

At this point, the hash table only contains the Node N4. After searching for a file of document genre G1, the process continues again with the document genre G2 (because of the priority list G2-G3-G5-G4-G1). The search will fail for the genres G2 and G3 (the node N4 contains only two files of the genres G4 and G5) but will find a file of the document genre G5. If this file is different from the one found at the node N2 (different URL), the assignment N4(G5) is obtained. Else, the search continues with the next document genre G4 (G2-G3-G5-G4-G1). The search will return a file that is compared to the one found in the node N3. If there are the same (same URL), then no file is assigned to the information nugget whose key is N4.
This scenario clearly explains:

- why there is no file for the class Twitter in Figure 6-19,
- why to some nuggets are not assigned a file
- why to more than a file can be assigned the same document genre (each file retrieved with a different information nugget).

Figure 6-20 shows the repartition of the downloaded files after automatically selecting the files for each information nugget. There are in average 7 file spreads in about 3 documents document for each topic. The graphic (Figure 6-20) also shows that a maximum of 10 web pages per topic has been reached. A short information nuggets list for a given summary involves that few files are downloaded for the corresponding topic. Some documents type are rare among the downloaded files; by increasing the number of requested URL (perhaps 30 instead of 10) the number of document genres, as well as the number of the files downloaded per topic, may increase.

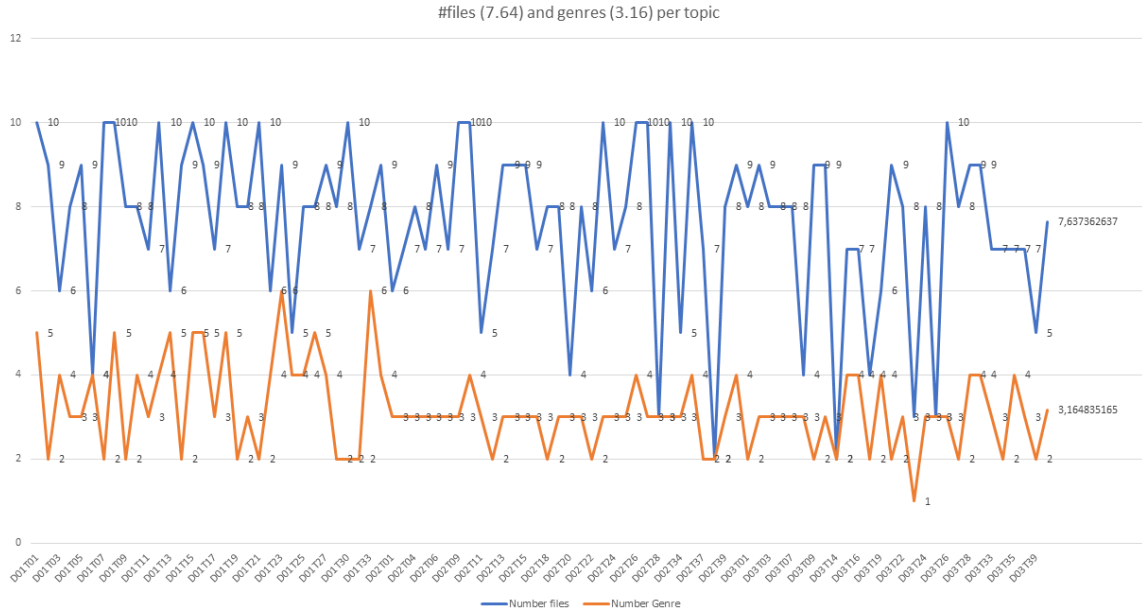


Figure 6-20: Repartition of document genres per topic with 10 URLs requested per Information nuggets

When requesting 30 URLs for each Information nugget

The same experiments have been further performed on 61 topics (30 topics from domain 1, 30 topics from domain 2, and 7 topics from domain 3). A Bing Web search parameter has been changed to request 40 URLs for each information nuggets. More than 21,000 files have been downloaded and classified. It has been remarked that it can only return a maximum of 33 URLs per request. After selecting a unique file for each information nugget, this new corpus contains 818 files (For 67 topics). As in the first experiments, to some nuggets has been not assigned any web page.

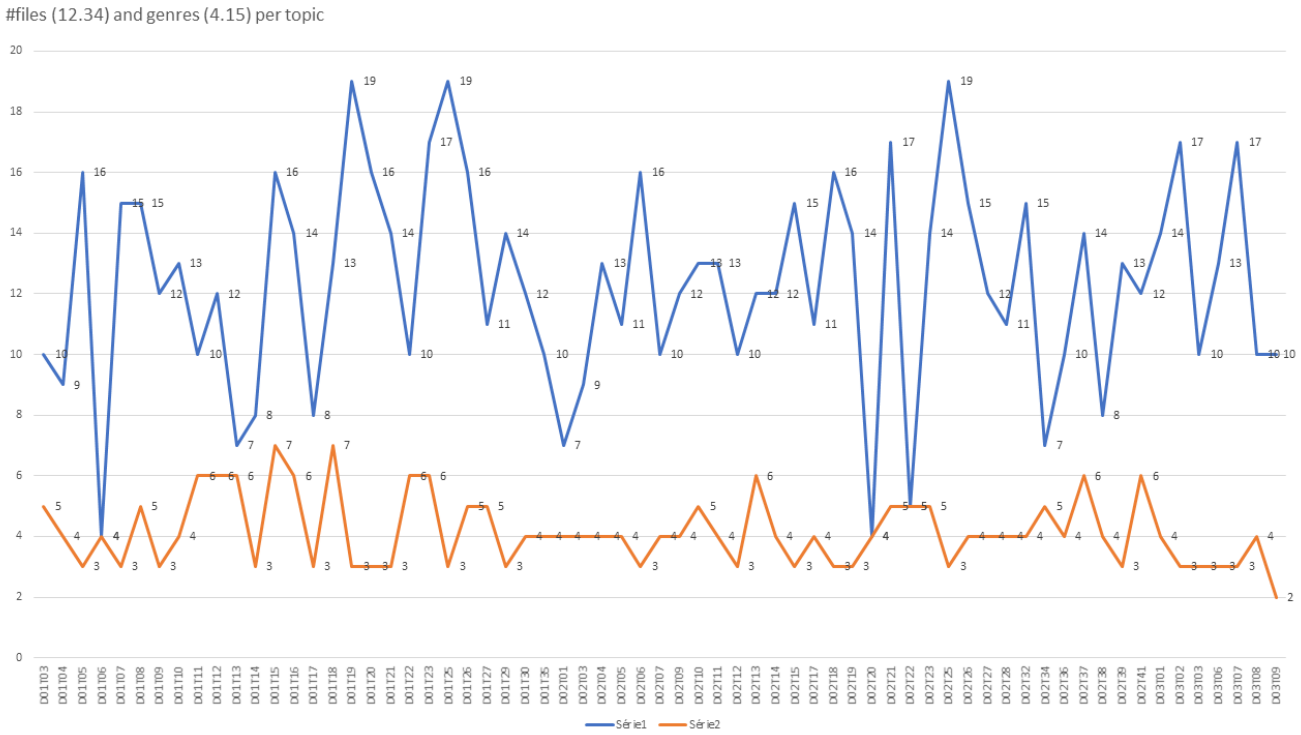


Figure 6-21: Repartition of document genres per topic with 30 URLs requested per Information nuggets

Figure 6-21 shows the achieved results when requesting 33 URLs for each information nugget. The number of files per topic increases and an average 12 files per topic has been achieved. These files are distributed in 4 documents genre in average per topic. The files of the Twitter class are absent, and those of the education class remain rare (Figure 6-22 (a)). Surely, the twitter conversations about the chosen domain are rare. Figure 6-22 (b)-(c) presents the repartition of the document genres of the first five topics. The topics D01T01, D01T02, D01T03, D01T04, and D01T05 contain respectively 13, 14, 10, 9, and 16 files and are classified respectively into 6, 3, 5, 4, and 3 documents genres.

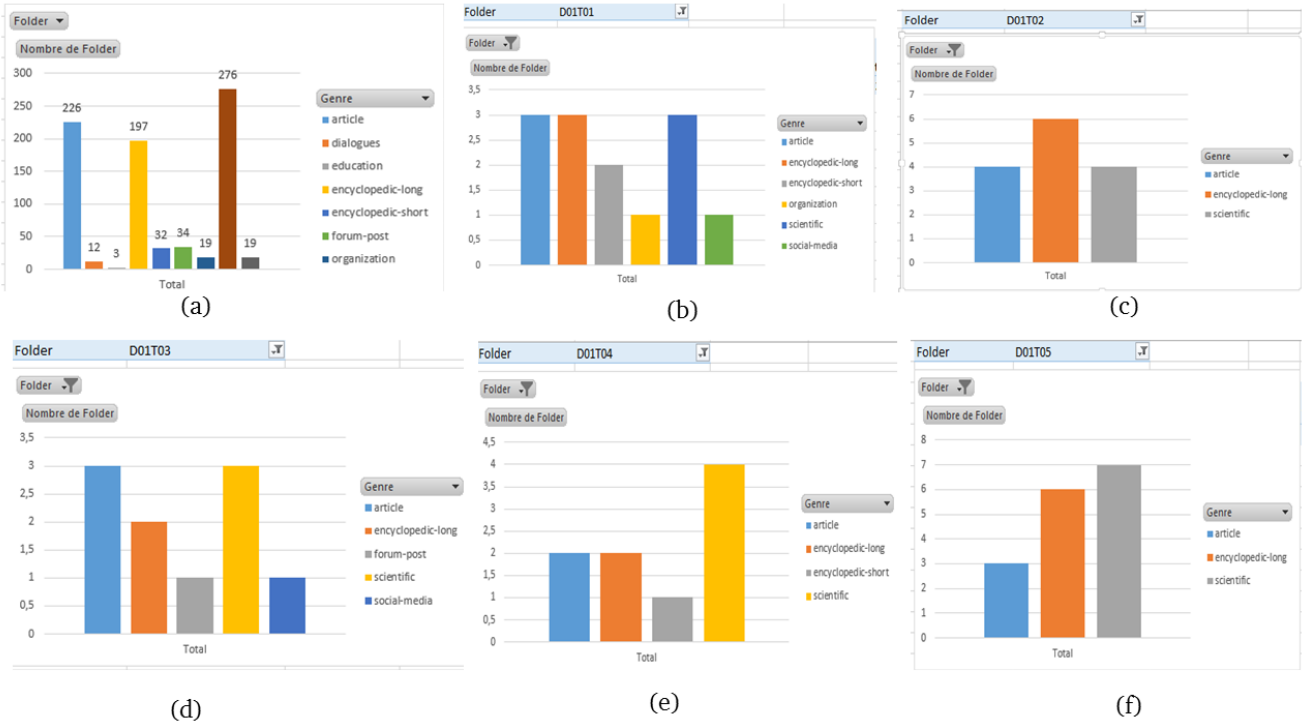


Figure 6-22: Some distribution of document genres per topic

7 Some Results: Example of the World Trade Center Topic

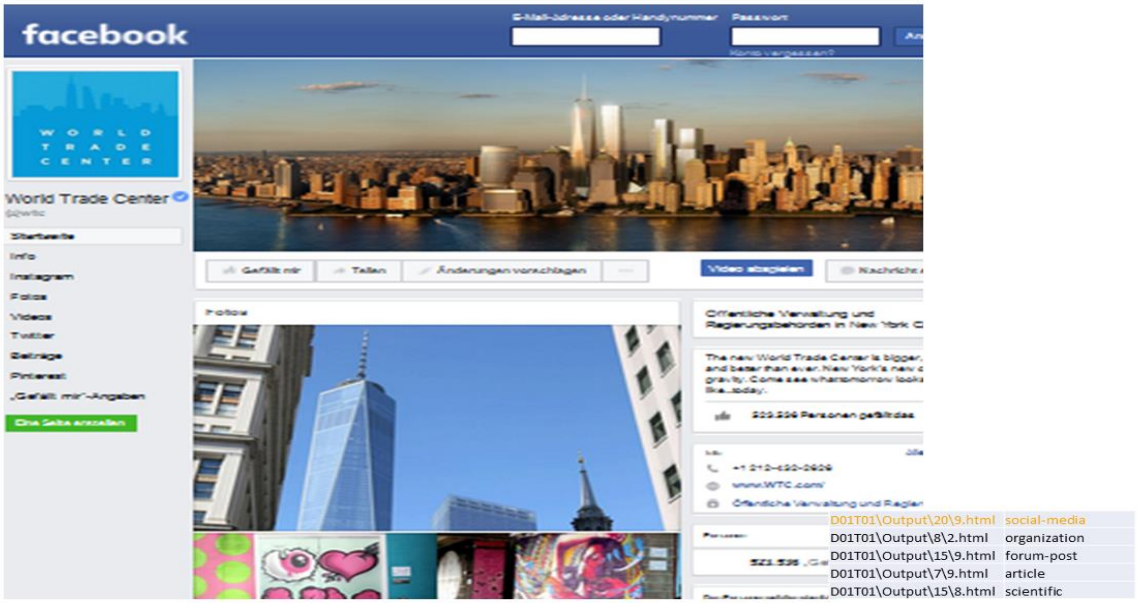


Figure 7-1: Example of a social media web page for the topic D01T01

This web page is good classified because it corresponds to the description given for the social media web pages (**social media**, e.g. Facebook, google+)

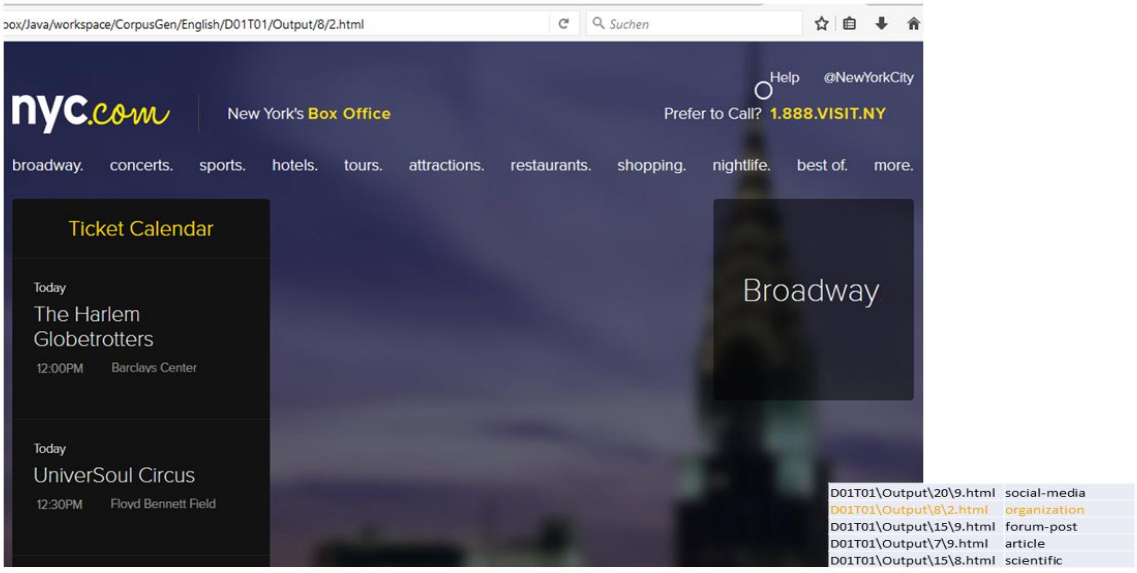


Figure 7-2: Example of an organization web page for the topic D01T01

This web page is also good classified because it contains announcements as described for the organization genre (**organization**, e.g. announcements (company, NHL, ...), press releases, and other texts which have been written with the purpose of advertisement, also including commercial descriptions, e.g. for a book or a picture)

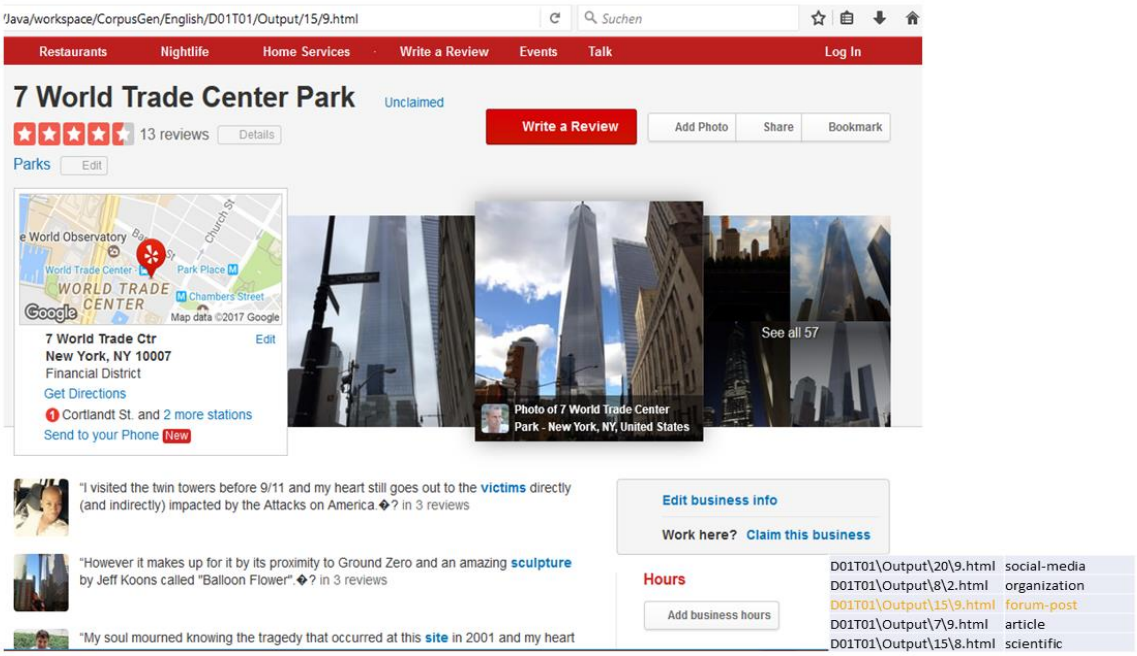


Figure 7-3: Example of a forum post web page for the topic D01T01

This web page is also good classified then it contains comments as specified for the forum genre: (**forum post**, e.g. blog/forum post (e.g., Reddit,), question answering site (e.g. quora, StackOverflow), youtube comments (low quality due to missing context))

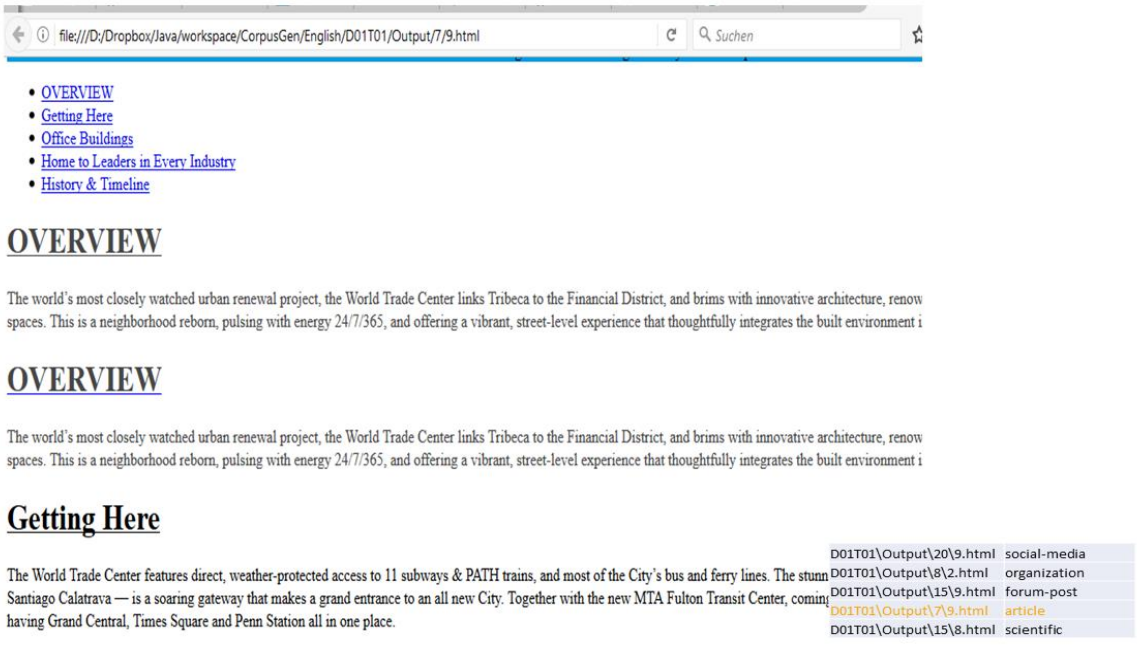


Figure 7-4: Example of an article web page for the topic D01T01

Since this page is a high-quality blog, it is well classified as an article (**article**, e.g. news/high quality (blog) article (e.g. bbc.com, spiegel.de, etc.)

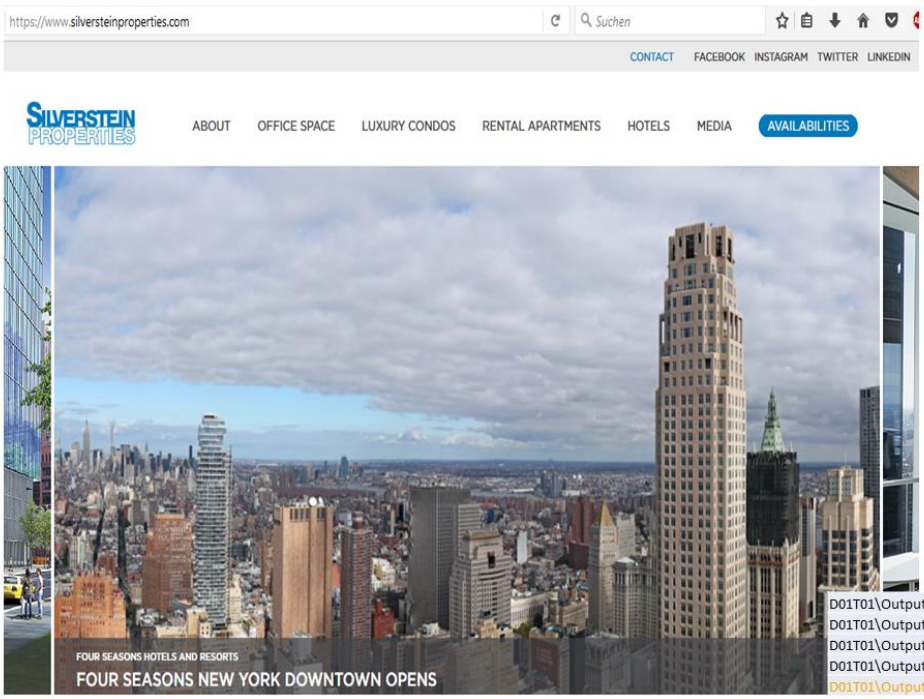


Figure 7-5: Example of a scientific web page for the topic D01T01

This web page does not contain any of the specified characteristics for a scientific article (**scientific**, e.g. scientific articles (containing citations and a bibliography), book reviews). It should be classified as an organization because it presents the Silverstein properties (Larry Silverstein is the architect of the World Trade Center).

8 How to Run the Developed Tool

To run this tool, just care about following points:

- The path to the corpus must be like "../hMDS/English" case sensitive.
- The file containing the summaries title "topics.txt" must be in the root of the English directory.

You must apply for a Bing Search key[24], then go to the dke.classification.Web pageearch.java class and search for the method bingSearch(). Update the instruction request.setHeader ("ocp-Apim-Subscription-Key, "YOUR-API-KEY") with your API Key and then uncomment the call to the bingSearch method (located in the processNamedEntityFile method of the same class).

Architecture Eclipse-Implementation

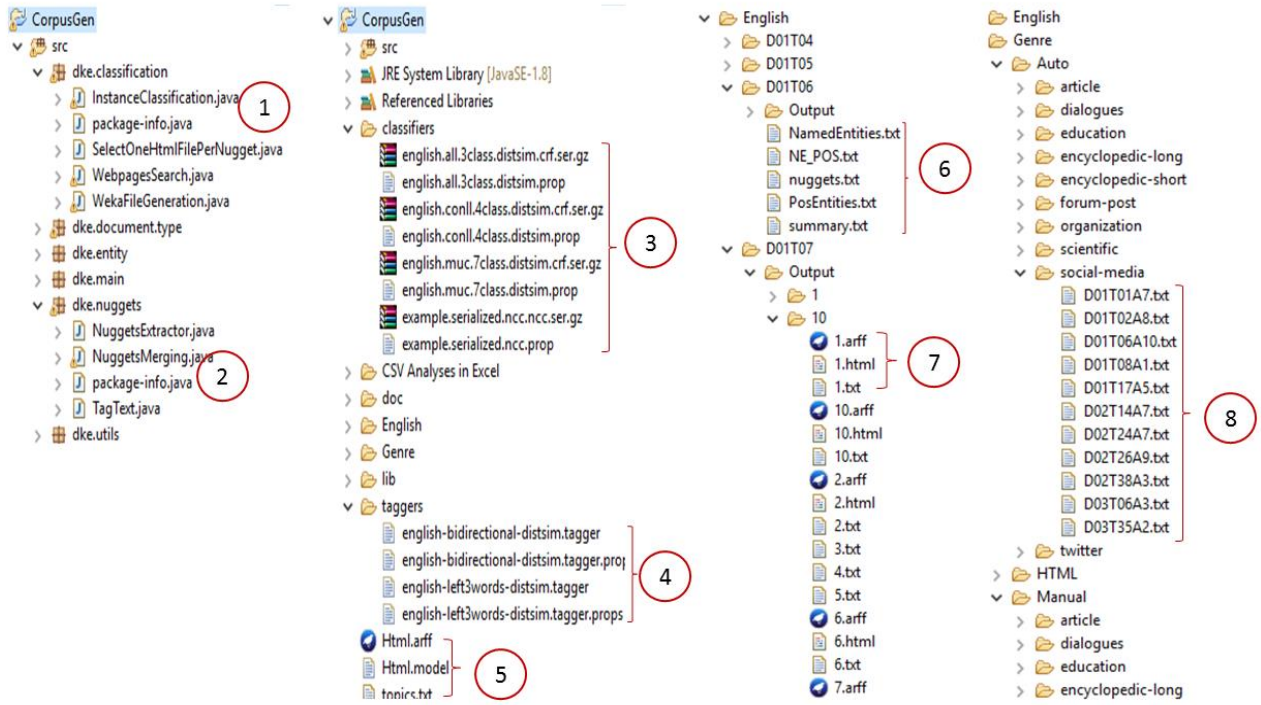


Figure 8-1: Workspace description

- 1&2 are code source organized in packages. The main class is located in dke.main package and can be call through the command line or eclipse. The absolute path to the corpus is given as parameter and must end with "/hMDS/English" (case sensitive).
- The dke.nuggets package (2) contains three independent classes corresponding to the three approaches evaluated for the nuggets extraction. The deletion of a class does not have an impact on the others.

[24] https://www.microsoft.com/cognitive-services/en-us/bing-web-search-api.

- Three Named-Entity classifiers (3) from the Standford web page have been downloaded and used. They are able to recognize following entities type in a text: person, location, organization, time, date, money, percent and miscellaneous. The example.serialized.ncc.ncc.ser.gz contain the three others classifiers.

- The taggers folder (4) contains two classifiers for the Part of Speech tags (PoS) extraction.

- Generated ARFF file from the HTML files available in the corpus (5). Afterward, the SVM classifier is trained, and the model is saved into Html.model. This model is further used to classify new instances.

- The topics.txt file is provided with the corpus and contains the list of all the summary titles in the corpus. Its name is case sensitive, and it should be located in the root of the English directory (../hDMS/English).

- The summary.txt and the nuggets.txt file (6) are provided with the corpus. The PosEntities.txt contains a compound-PoS list. The POS lists have been built by observing how the information nuggets in the manually extracted list are built. The NamedEntities.txt file contains the available Name-Entities from the summary file. The NE_POS.txt file is a copy of the NamedEntities.txt file when it contains more than t elements (t is a threshold) and is a copy of the PosEntities.txt file in the other case (this corresponds to the proposed solution in 4.2.5). If more than nine nuggets per summary file have to be provided, then just set the parameter t = 9 in the program code. The PosEntities file always contains more elements than the NamedEntities file. The maximum number of nuggets par file was limited by twenty. The precision and recall were performed by comparing the three generated files to the nuggets file.

- The Bing Web Search API has been used to retrieve web pages from the information nuggets. The search is done by giving an information nugget and the corresponding summary title as a parameter in order to become topic related web pages. For each web page, its URL and its title have been extracted and saved in the text files (7). Then the HTML files have been downloaded from the URLs. Many features of these HTML files have been exploited to build to build the ARFF files. This ARFF is considered as a new instance and is classified using the Html.model that has been previously generated. The class genres are then written in the TXT files after the classifications. This corresponds to the solution in 6.2. At the end of this process, each text file contains an URL, the document title, the nugget used, and the assigned class.

- To be able to evaluate the corpus using WEKA classifiers, the files have to be reorganized as in (8). There is a directory of the HTML files, a directory for the manual extraction of visible texts, and a directory for the automatic extraction of visible texts. Each of these directories contains ten folders renamed by the document genres in which, the files have been classified (described in 5.1).

9 Conclusion and Future Work

In this thesis, the automatic creation of a corpus for the heterogeneous Multi-document summarization tasks has been evaluated. This work is based on the model proposed by (Zopf et al., 2016) and consists of three phases: extraction of information nuggets, retrieval of web pages, and classification of the downloaded files.

The first phase consists of the automatic extraction of information nuggets, the Part of Speech and the Named Entities have been used for this purpose. The best result has been obtained using the named entities (Precision: 58.47 %, Recall = 72.57 % and F-Measure = 64.76 %.). These percentages are normally higher because the automatically extracted lists have been compared to that manually extracted and these manually extracted lists are incomplete because they should contain at most 20 information nuggets.

The second phase consisted of downloading related web pages to the given topics that contain these information nuggets. The Bing Web Search API has been used for this purpose.

The last phase consisted of classifying the downloaded web pages into the 10 different genres that the corpus has. To do this, classifiers have been trained on the datasets supplied with the corpus (saved web pages, the text versions obtained by manual extraction and another version obtained by automatically extracting the visible content from the HTML pages). After training with six classifiers (NaiveBayes, Logistic, SVM, NBk, JRip, and J48), the best performance is achieved when using the SVM classifier on the HTML file type.

The achieved accuracy is 61.59 %, which is lower than on other datasets such as Reuter or C50 (paragraph 6.4.3). This led us to consider another classification approach, which consisted of considering the text features, the URL features, and structure of HTML files to generate the attributes. Some of these features have been used and not any improvement have been observed. However, if others characteristics are added as for example taking into account the nature of the linked web pages, then this accuracy may be improved.

The corpus has also been analyzed to find some causes of the misclassification. It is noticed that some downloaded web pages were empty, that many short documents had been manually classified in the encyclopedic-long class, which does not make easy for the classifiers to separate the encyclopedic-long and encyclopedic-short classes by their size. Another remark is that the twitter web pages often contain others languages outside the English language. In addition to these remarks, the files of different document genres have been compared pairwise to discern those that are easily distinctive. The results revealed that some of them are hardly separable as for example the article genre to the organization genre or the Encyclopedic-long genre to the education genre. However, the Twitter and social media genres are easily differentiable from the rest of the genres in the pairwise tests. This analysis showed us that some files could belong to several documents genres at the same time. This is the case of many files that have been misclassified as an article, which caused this average accuracy.

After classifying the new files, an evaluation gave 90% of the downloaded HTMP web pages belong to the scientific, article, and encyclopedic-long documents genres. The remaining 7 document genres are represented in 10% of the downloaded files. This leads to the conclusion that the topics of the domain of Art, Architecture, Archaeology, History, Law, Politics, and Government, are mostly written in the documents genres scientific, article, and encyclopedic (long). A manual search of the documents makes it possible to have more diversity in the genre than an automatic search because one can search a file of a specific genre in the following pages returned by a search engine. The first 33 pages retrieved by the search engine for each nugget have been considered. After the classification, in average 12 files have been downloaded for each topic and they were distributed into 4 different documents genres.

All in all, the process described by (Zopf et al., 2016) can be automated. Some documents genres for the chosen domains are rare. Nevertheless, a heterogeneous corpus with an average of 4 genres per topic has been created.

In a later work, the ClausIE tool can be used in combination with a good co-reference resolution tool to filter the list of the automatically retrieved information nuggets as explained in paragraph 4. The accuracy of the classifiers would be performed by considering the neighborhood of the web pages (Parent, child, sibling, and Spouse) and some others text, URL, and HTML features by the classification to improve the accuracy of the algorithms. The classification based on the Jensen-Shannon (JS) divergence has not been explored. It could make an improvement on other types of files (The texts files obtaining by extracting the contents of HTML pages). Another question is how to get more files of rare documents genres in the results of search engines?

List of Figures

List of Tables

List of Abbreviations

MDS	Multi-Document Summarization
hMDS	heterogeneous Multi-Document Summarization
NLP	Natural Language Processing
POS	Part Of Speech
CPOS	Compound Part Of Speech
NER	Named Entity Recognition
OIE	Open Information Extraction
DUC	Document Understanding Conference
TAC	Text Analysis Conference
STWV	StringToWordVector Filter
JS	JavaScript
CSS	Cascade Style Sheets
D01	Domain 1 (Art, Architecture, and Archaeology)
D02	Domain 2 (History)
D03	Domain 3 (Law, Politics, and Government)
TF	Term Frequency
IDF	Inverse Document Frequency
NE	Named Entity

References

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, *6*(1), 37–66. https://doi.org/10.1023/A:1022689900470

Aker, A., & Gaizauskas, R. (2006). Model Summaries for Location-related Images. *Text*, 3119–3124.

Almeida, T. a, Hidalgo, J. M. G., & Yamakami, A. (2011). Contributions to the study of SMS spam filtering: new collection and results. *Proceedings of the 11th ACM Symposium on Document Engineering*, 259–262. https://doi.org/10.1145/2034691.2034742

Benikova, D., Mieskes, M., Meyer, C. M., & Gurevych, I. (2016). Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources, 1039–1050.

Carenini, G., Ng, R. T., & Zhou, X. (2007). Summarizing Email Conversations with Clue Words.

Cohen, W. W. (1995). Fast effective rule induction. *Twelfth International Conference on Machine Learning*, 115–123. https://doi.org/10.1.1.50.8204

Del Corro, L., & Gemulla, R. (2013). ClausIE: Clause-based Open Information Extraction. *Proceedings of the 22Nd International Conference on World Wide Web*, (i), 355–366. https://doi.org/10.1145/2488388.2488420

Dwivedi, S. K. (2016). News Web Page Classification Using Url Content and Structure Attributes, (October), 317–322.

Edu, S., John, G. H., & Edu, S. (1995). No Title, 338–345.

Giannakopoulos, G., El-Haj, M., Favre, B., Litvak, M., Steinberger, J., & Varma, V. (2011). TAC 2011 MultiLing Pilot Overview, (November). Retrieved from http://eprints.pascal-network.org/archive/00009325/

Goldstein, J., Carbonell, J. G., Kantrowitz, M., Carbonell, J., & Kantrowitzt, M. (2000). Multi-Document Summarization By Sentence Extraction Multi-Document Summarization By Sentence Extraction i l i i, 40–48.

Gu, M., Zhu, F., Guo, Q., Gu, Y., Zhou, J., & Qu, W. (n.d.). Towards Effective Web Page Classification.

Habernal, I., Zayed, O., & Gurevych, I. (2016). C4Corpus: Multilingual Web-size corpus with free license. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, 914–922. Retrieved from http://www.lrec-conf.org/proceedings/lrec2016/pdf/388_Paper.pdf

He, K., & Li, C. (2016). Structure-Based Classification of Web.

Hirao, T., Fukusima, T., Okumura, M., Nobata, C., & Nanba, H. (2004). Corpus and evaluation measures for multiple document summarization with multiple sources. *Proceedings of the 20th International Conference on Computational Linguistics - COLING '04*, *2*(in 2002), 535–es. https://doi.org/10.3115/1220355.1220432

Joachims, T. (1998). 1 Introduction 2 Text Categorization 3 Support Vector Machines. *Machine Learning*, *1398*(LS-8 Report 23), 137–142. https://doi.org/10.1007/BFb0026683

Kan, M.-Y., & Thi, H. O. N. (2005). Fast webpage classification using URL features. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management - CIKM '05*, 325. https://doi.org/10.1145/1099554.1099649

Lloret, E., Plaza, L., & Aker, A. (2013). Analyzing the capabilities of crowdsourcing services for text summarization. *Language Resources and Evaluation*, *47*(2), 337–369. https://doi.org/10.1007/s10579-012-9198-8

Loupy, C. De, Guégan, M., Ayache, C., Seng, S., & Moreno, J. T. (2010). A French Human Reference Corpus for Multi-Document Summarization and Sentence Compression The Multi-Document Summarization. *LREC Workshop*, (1), 3113–3118.

M. Lichman. (2013). UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. Retrieved from http://archive.ics.uci.edu/ml

Marneffe, M. De, & Manning, C. D. (2010). Stanford typed dependencies manual. *20090110 Httpnlp Stanford*, *40*(September), 1–22. https://doi.org/10.1.1.180.3691

Nakano, M., Shibuki, H., Miyazaki, R., Ishioroshi, M., Kaneko, K., & Mori, T. (2010). Construction of Text Summarization Corpus for the Credibility of Information on the Web . Construction of Text

Summarization Corpus. *Lrec*, (October), 3125–3131.
Nenkova, A., & McKeown, K. (2011). Automatic Summarization. *Foundations and Trends® in Information Retrieval*, *5*(3), 235–422. https://doi.org/10.1561/1500000015
Neto, J. L., Freitas, A. A., & Kaestner, C. A. A. (2002). Automatic Text Summarization using a Machine Learning Approach. *SBIA '02 Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, (i), 205–215.
Ng, V., & Cardie, C. (2002). Improving Machine Learning Approaches to Coreference Resolution, (July), 104–111. https://doi.org/10.3115/1073083.1073102
Over, P., Dang, H., & Harman, D. (2007). DUC in context. *Information Processing and Management*, *43*(6), 1506–1520. https://doi.org/10.1016/j.ipm.2007.01.019
Over, P., & James, Y. (2001). Introduction to duc-2001: an intrinsic evaluation of generic news text summarization systems. *In Proceedings of DUC 2004 Document Understanding Workshop, Boston.* Retrieved from http://www-nlpir.nist.gov/projects/duc/pubs/2001slides/pauls_slides/sld001.htm
Platt, J. C. (1998). Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. *Advances in Kernel Methods*, 185–208. https://doi.org/10.1.1.43.4376
Qi, X., & Davison, B. D. (2009). Web page classification. *ACM Computing Surveys*, *41*(2), 1–31. https://doi.org/10.1145/1459352.1459357
Recasens, M., Marneffe, M. De, & Potts, C. (2013). The Life and Death of Discourse Entities: Identifying Singleton Mentions. *Proceedings of NAACL-HLT*, *0*(June), 627–633. Retrieved from http://www.aclweb.org/anthology-new/N/N13/N13-1071.pdf
Riboni, D. (2002). Feature selection for web page classification. *EURASIA-ICT 2002 Proceedings of the Workshop*, 473–477. https://doi.org/10.4018/978-1-60566-196-4.ch012
Ross, Q. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.
Society, R. S. (2017). Short-Term Sales Forecasting Author (s): P . J . Harrison Source : Journal of the Royal Statistical Society . Series C (Applied Statistics), Vol . 14 , No . Published by : Wiley for the Royal Statistical Society Stable URL : http://www.jstor.org/stabl, *14*(2), 102–139.
Stoyanov, V., Cardie, C., Gilbert, N., Riloff, E., Buttler, D., & Hysom, D. (2010). Coreference resolution with reconcile. *Proceedings of the {ACL} 2010 Conference Short Papers*, (July), 156–161. Retrieved from http://dl.acm.org/citation.cfm?id=1858871
Toutanova, K., Klein, D., & Manning, C. D. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. *In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL '03),* 252–259. https://doi.org/10.3115/1073445.1073478
Versley, Y., Ponzetto, S. P., Poesio, M., Eidelman, V., Jern, A., Smith, J., … Moschitti, A. (2008). BART : A Modular Toolkit for Coreference Resolution. *Lrec*, (June), 1–4. https://doi.org/10.3115/1564144.1564147
Vidulin, V. (2007). Training a Genre Classi fi er for Automatic Classi fi cation of Web Pages, 305–311.
Wen, H., Fang, L., & Guan, L. (2008). Automatic Web Page Classification Using Various, (1), 368–376.
Zechner, K. (2002). Automatic Summarization of Open-Domain Multiparty Dialogues in Diverse Genres. *Computational Linguistics*, *28*(4), 447–485. https://doi.org/10.1162/089120102762671945
Zopf, M., Peyrard, M., & Eckle-kohler, J. (2016). The Next Step for Multi-Document Summarization : A Heterogeneous Multi-Genre Corpus Built with a Novel Construction Approach. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, (i).