Finding optimal text queries to cover Subjects in a taxonomy of a news article database

Joshua Ratha (295208)

Bachelor 3 Thesis Economics & Informatics
Major Business Information Systems
Supervisor: Uzay Kaymak

August 26, 2010

Contents

Chapter 1

Introduction

1.1 Introduction

In this thesis we investigate a news article database. The news articles in this database are classified in a taxonomy. A taxonomy is a collection of entities organized into a hierarchical structure. Each entity in a taxonomy is in one or more parent/child (broader/narrower) relationships to other entities in the taxonomy. In this case are dealing with a 3 level deep taxonomy. The first level of the taxonomy is called 'Main Heading' with 60 headings, the second level of the taxonomy is called 'Sub Heading' with 190 headings and the third and final level of the taxonomy is called 'Subject' with 3194 subjects. All articles belong to at least one Subject and they can belong to multiple. The taxonomy is visualized in a pyramid form in figure 1.1 and in a table form in figure 1.2.

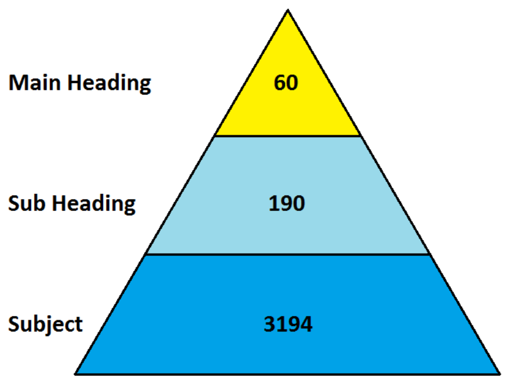

Figure 1.1: Taxonomy representation by Pyramid

The users of this database can find their articles by using the taxonomy and by entering free text queries. A query can be defined as a request for information from a database.

1.2 Problem Statement

MD Info is the company that owns the article database and the taxonomy. They deliver business information and news to various customers in different branches. MD Info face the problem of having a too large taxonomy, containing 3194 subjects. This has some disadvantages for both MD Info and their customers. For MD Info it is very labor-intensive to classify articles into over three thousand different Subjects. For the customer it is difficult to choose which subjects they want to receive news from.

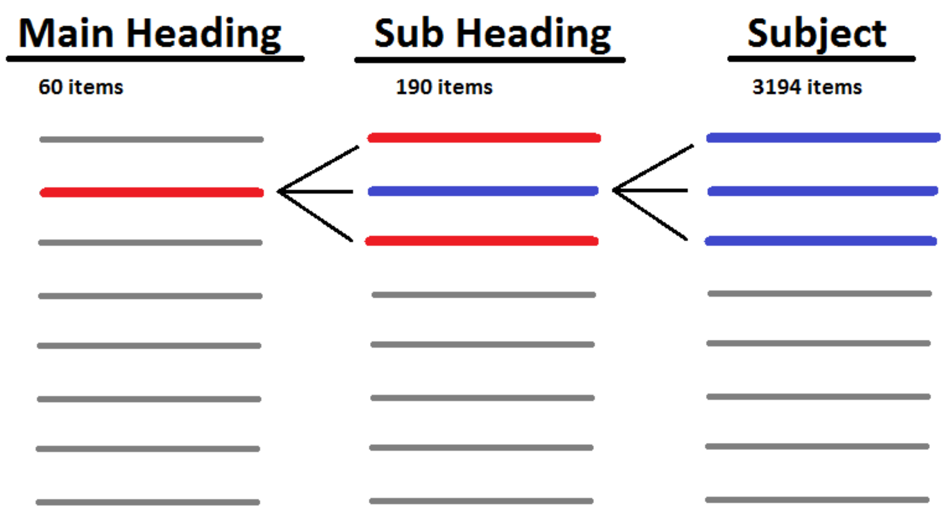

Figure 1.2: Taxonomy representation by table

In this thesis we consult the company by showing which subjects are retrievable by a free text queries and which are not. A subject is retrievable when the article set that corresponds with the subject can be retrieved satisfyingly using a free text query and not the taxonomy subject. Therefore we need to find the optimal query for each subject to find the article set that corresponds best with the original article set of the subject. In this experiment, optimal free text queries inside the Sub Heading and optimal free text queries inside the complete article database are retrieved. So per Subject, two optimal free text queries are retrieved.

1.3 Relevance

The firm would benefit from a smaller taxonomy. But the database may not lose it's functionality and original intention: bringing relevant news to the customer. That is why we are investigating the retrievability of the subjects inside the taxonomy. The theory that I have for this approach is that if the set of articles that corresponds with a certain subject can be retrieved by a free text query, that subject is less needed. In order to consult the MD Info decision maker we reveal which subjects are retrievable with free text queries and therefore can be changed or fused with other subjects or maybe be removed from the taxonomy. The relevance of the research to the problem is that this research will let come to light which subjects are retrievable by a query without the use of the taxonomy subject.

1.4 Goal of the Research

The retrieval of articles can be done in two ways, making use of the taxonomy or a free text query. The goal of this thesis research is to find the optimal total free text query and the optimal free text query inside the Sub Heading for each subject from the taxonomy and to consult MD Info on which subjects could be changed or fused with other subjects because that particular content can be found without the subject in the taxonomy. So this research is not actually reducing the taxonomy, but showing which subjects could be replaced or changed or fused.

1.5 Research Question

The Research question of this thesis therefore is:
Of which subjects from the MD Info database, the article content can be satisfyingly retrieved, making use of total free text queries or free text queries inside the Sub Heading?

1.6 Methodology

To be to able to answer the research question of this thesis, certain steps are taken. A literature review is performed to find algorithms and methods to approach the problems that arise in this experiment. Also a detailed problem description is desirable to work from in the following experiment. Than the actual experiment starts. First off, the news article database is retrieved from the MD Info site. Then the data is prepared to find the most representative words for each subject. The data is filtered from reading signs and stop words, which are words which are filtered out prior to, or after, processing of natural language data. Then with these representative words, for each subject the optimal text query for inside the Sub Heading and the optimal free text query inside the complete article database are retrieved. After the optimal queries are found, the results are evaluated.

This is the overview:

1. Perform a Literature review
2. Define a detailed Problem Description
3. Retrieve news articles data set
4. Prepare data by filtering and applying stemming algorithms
5. Find most representative words for each subject
6. Finding queries for optimal search results
7. Show and evaluate the results

1.7 Chapter Outline

In chapter two, 'Related Work', the related work in the field shall be discussed. In chapter three, 'Problem Description', U can find an extensive problem description. Chapter four contains a theoretical study on the evolutionary approach to solve the query optimization problem. Chapter four contains the 'Analysis Method'. Chapter five describes the full experiment thoroughly step by step in the 'Experimental Design'. In chapter six, 'Results', the results are presented. Chapter seven, 'Conclusion', contains the conclusion.

Chapter 2

Related work

2.1 Introduction

There are several research fields that are relevant to the problem that we are addressing in this thesis. The first is that of Stemming. In the experiment we retrieve representative words for each subject. Words have multiple morphological variants for plurals or adjectives. These different variants have the same semantical meaning so we want them to be counted as one. Therefore Stemming literature is reviewed.

A measure is needed to define how relevant a word is to a subject and a measure is needed to for the performance of a query. To search for the optimal query an effective method is desired. The research fields related to these are treated in the following sections.

2.2 Work related to Stemming

In this research we shall use 'stemming' to enhance recall. Stemming is a method to bring morphological variants with the same semantical mean back to their stem, which is the same. This is vital to enhance recall in Information Retrieval, as the morphological variant are counted and recognized are one and the same word. In our experiment we are counting the occurrence of words in articles and words with the same semantical meaning should be counted as one.

Kraaij & Pohlman (1994) evaluate the Porter stemming algorithm for the Dutch language and apply some adaptations to make it fit for Dutch. The principle is to stem back back morphological variants back to their same base form(stem). We want this because we assume that these variants are semantically related. In their paper they mention two stemming techniques: Suffix Stripping and Stemming bases morphological analysis. The second technique involves dictionary lookup for each word that is being stemming and is slower than suffix stripping. It eliminates most errors and therefore can be useful for high complexity purposes. The porter algorithm uses Suffix Stripping, which uses small and efficient algorithms. The Dutch version of the Porter stemming algorithm reduces the vocabulary size with 57%.

Harman (1991) investigated the interaction of suffixing algorithms and ranking techniques in retrieval performance. Three algorithms were used, namely 'S' stemming algorithm, the Lovins (1968) algorithm, and the Porter (1980) algorithm. Although many other stemming already existed at the time, these three had widespread usage in the information retrieval community. An advantage is that stemming also saves some storage space, which is useful for online applications. Harman (1991) state that a the use of a stemming algorithm such as Porter's would be a realistic approach to improve information retrieval.

Mark Kantrowitz (2000) investigate the effect of several Stemmers on the Information Retrieval performance and accuracy. To analyze the effect they ran a number of experiments, running short and long queries on four different large data sets. The ranking of the retrieved documents was computed making use of TF-IDF. Both with long and short queries stemming had effect. They did seem to help.

2.3 Work related to Information retrieval: Word weighting

2.3.1 TF-IDF word weighting heuristic

In this project we are trying to find optimal search queries to find the articles that belong to each of the subjects. For that we need to define which words are most representative for a subject. There are several word weighting heuristics and TF-IDF is one of them. TF-IDF stands for Term Frequency Inverse Document Frequency.

The TF-IDF heurtic is described by Joachims (1996). In Joachims (1996) TF-IDF is used to retrieve the most relevant documents to a certain query word. The Term frequency TF(w,d) is the number of times that word w occurs in document d. A high TF is positive for the for the relevance of document d to word w. The Document Frequency DF(w) is the number of documents where word w appears in at least one time. A low DF is desired for the document to be exclusively relevant to the word. Therefore the inverse is used in the word weighting heuristic.

In the Joachims (1996) example, a document is weighted for a word. The measure can be used the other way around to weight a word for a document. In this project I want to weight words for a subject. In our case we can use the Turn Frequency as the number of times that a word appears in the subject and we can use the Document Frequency as the number of times that the word appears at least one time in all subjects.

Ramos (2000) examines the results of applying TFIDF to determine what words in a corpus of documents might be more favorable to use in a query. Words with high TF-IDF numbers imply a strong relationship with the document they appear in, suggesting that if that word were to appear in a query. Ramos (2000) shows that TF-IDF is a simple and efficient algorithm fir matching words in a query to document that are relevant to that query. The only downside of the algorithm is that it doesn't draw any relation to synonyms.

The TF-IDF word weighting heuristic is very useful to this project. After it is adapted to this project it doesn't only assign value to the appearance inside the subject but also to the uniqueness comparing to the other subjects. That makes TF-IDF a very useful measure to select the best representative words per subject, because later in the experiment they are used to form an optimal query with, which shall be measured by recall and precision, where recall can be seen as a frequency measure and precision as a uniqueness measure.

2.4 Work related to Information Retrieval: Performance measure

2.4.1 Quality measures

In the experiment we are finding the optimal query for each subject. Each query classifies articles as true or false, to be in the subject or not. To measure the quality of a certain query we need a measure do this with. Since we are measuring a classification problem we could be using 'accuracy' as a measure. We are classifying articles to be inside or outside a subject, so true or false. But when the data set is very unbalanced this is a poor metric to use as R.Musicant et al. (2003) state. For example, of a data set which has 99% of its points in class A, an accuracy maximizing classifier may draw the conclusion that all point are in class A. So for an unbalanced data set as all of the article sets -to be in a subject or not- are, an other measure is desired.

2.4.2 Precision and Recall

In Information Retrieval precision and recall are the most used measures. Rusinol & Llados (2009) provide the formula's for precision and recall in their paper. Precision is defined as the ratio between the number of relevant retrieved items and the number of retrieved items. Recall is defined as the number of relevant retrieved items as a ratio to the total number of relevant items in the collection. Precision measures the quality of including only relevant documents in the search and recall measures the effectiveness of retrieving as much relevant documents ass possible.

2.4.3 Combining Precision and Recall in the F-score

As precision and recall both are desired, a good information retrieval quality measure should combine both. As precision improves at the expense of recall and recall improves at the expense of precision (Gordon & Kochen (1989)), the decision maker must choose a compromise. A balanced measure between is used by many in information retrieval. The harmonic mean between precision and recall is the most used answer to this problem. This is called the F-score. Formula's are given by van Rijsbergen (1979). The β value is the weight of recall in the recall - precision ratio. With $\beta = 2$ recall is weighted twice as much as precision and with $\beta = 0.5$ precision is weighted twice as much as recall. The most popular value for β is 1 where precision and recall are weighted equally.

To measure the quality of a query, that can be seen as a hypothesis, to classify the entire article set to be be true or false to be in the subject or not, it is not desired to use accuracy since the dataset is unbalanced. There are more articles not in the subject that there are. So the combined measure of precision and recall is very fit to use in this project. Then is there the question of which β value to use. Since the value of 1.0 where both precision and recall are equally important is most popular and I don't see any particular reason to have a preference for one against the other, in this project the F-score with β=1.0 is used.

2.5 Work related to the Set Covering Problem

2.5.1 The Set Covering Problem

Beasley & Chu (1996) have applied several modifications to the basic genetic procedures. They use a genetic algorithm to solve a set covering problem, which is somewhat similar to our query optimization problem. The set covering problem is the problem of covering all the rows of a m-row, n-column, zero-one matrix (a_{ij}) by a subset of the columns at minimal cost. Here i stands for the row and j stands for the column. Defining $X_j = 1$ (with cost $c_j > 0$), if column j is in the solution and $X_j = 0$, if column j is not in the solution.

The Set Covering Problem is similar to our query optimization problem as we also try to to cover a row at minimal costs. The difference is that we try to maximize the row coverage where the 100% coverage can't be reached most times. Another difference is that the row that we try to cover (the subject) isn't all ones. In our problem the false coverage of zero's by one of the query solutions should also be punished somehow because it brings down the Information Retrieval score.

2.6 Work related to Evolutionary optimization

Wen-Chih Hung (1994) use the evolutionary approach in the form of Genetic Algorithms as well to solve two large test set covering problems, showing that the method is suitable for solving large problems just as the problem of this thesis. They state that 'survival of the fittest' provides the pressure for populations to develop increasingly fit individuals. Beasley & Chu (1996) also use the Genetic Algorithms to solve Set Covering and John et al. (2005) use Genetic Programming to a formula optimizing problem. We shall elaborate more on their work in the Evolutionary approach chapter.

Chapter 3

The Set Covering problem for news articles

3.1 MD Info, their news article database and the taxonomy

The company which owns the database is called MDInfo. They deliver business information and news to various customers in different branches. Their database contains summaries of articles from the past five years and they get these from over 700 sources, such as newspapers, magazines and multiple Internet sources. Each day there are about 80 articles added to their database. Each article is coded by one of the editing staff according to their own MD Info taxonomy. Their customers can access their database making use of so called 'profiles'. A profile is a search query which can be based on the taxonomy, a free text query or a combination of both. A user can make multiple profiles to satisfy his information need. MD Info often helps her customers with this process because the complexity and size of the taxonomy makes it complicated for the customer to define the profile settings on his own.

The 3 level taxonomy of consists of 60 Main Headings, 190 Sub Headings and 3194 Subjects. The article database contains 330721 articles, at moment that the database was downloaded from the MD Info site at the 28th of May.

3.2 The General Problem

MD Info faces the problem of having a too large taxonomy, containing 3194 subjects. This has some disadvantages for both MD Info and their customers. For MD Info it is very labor-intensive to classify articles into over three thousand different Subjects. For the customer it is difficult to choose which subjects they want to receive news from. It is also difficult, say at the moment not possible, to link their taxonomy to international standard taxonomies.

Nowadays with information coming available on the Web, the number of articles that are produced every day grows exponentially. With the taxonomy at its current state, these processing of these articles remains very expensive. If the taxonomy could be less specific without to much loss of relevant information reaching the customer, MD Info increases its chances to make the classification automatically. This allows the firm to increase their daily number of processed articles. So they want to make this taxonomy smaller, but have no idea where start and in which parts of the taxonomy to cut Subjects or Headings loose.

To consults the company for this problem, in this project the article sets that correspond with each subject are retrieved by optimal free text queries. Let's imagine a set of articles that correspond to Subject X and let's say query Q is used to retrieve that set of articles. In figure 3.1 the articles that belong to the subject X are visualized by the grey square. In figure 3.2 the articles that belong to query are visualized by the light blue square. In case of figure 3.2 the query

Q retrieves a large part of the original data set without retrieving too much irrelevant articles. In case of figure 3.3 not so much articles of the original data set are retrieved by query Q and much irrelevant article are retrieved. If Q is the best query for subject X, in case of figure 3.2, the subject is retrievable and could be replaced or changed or fused. In case of figure 3.3, the subject is poorly retrievable.

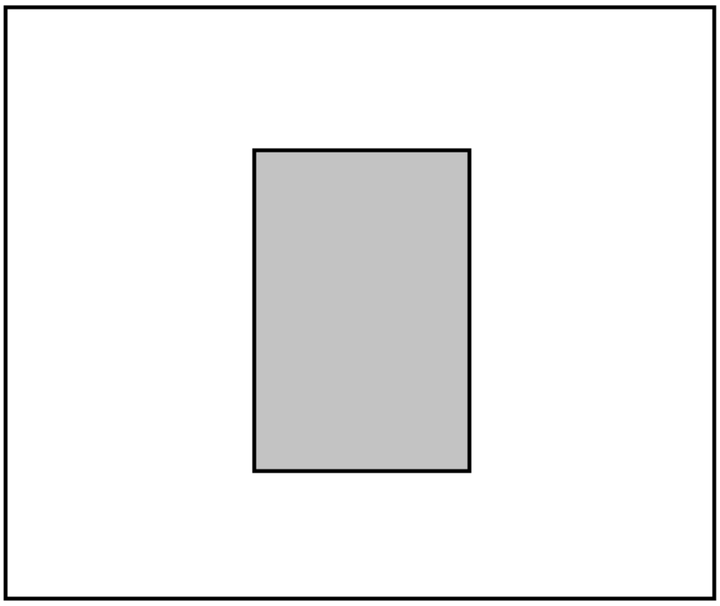
Figure 3.1: Subject X article set

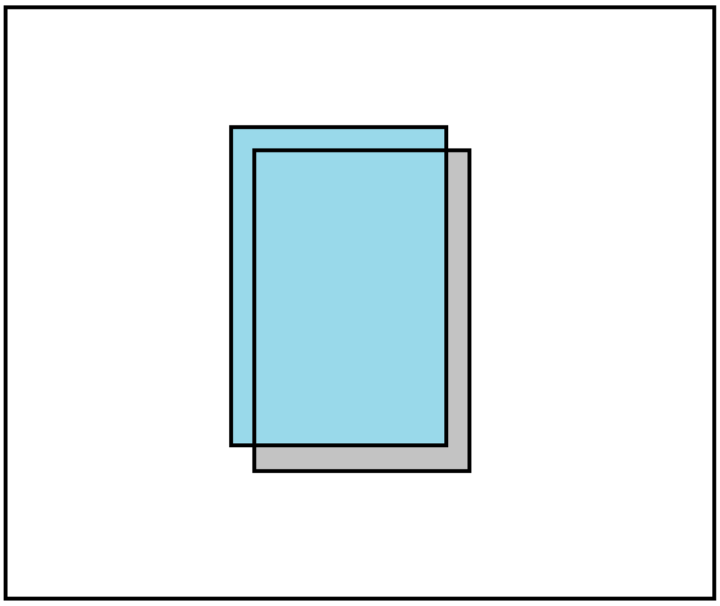
Figure 3.2: Subject X article set, well covered by query Q article set

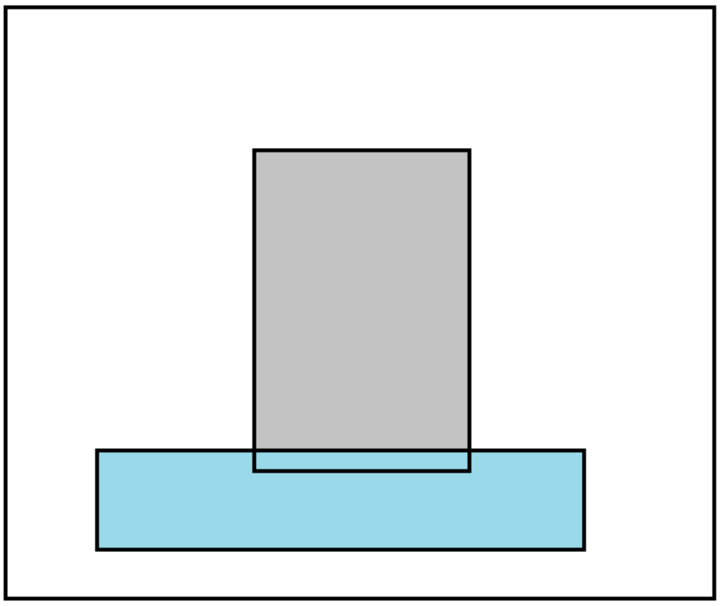

Figure 3.3: Subject X article set, poorly covered by query Q article set

Chapter 4

Content Stemming and Information Retrieval Measures

4.1 Porter's Stemming algorithm for Dutch

Before optimal queries for each subject in the taxonomy can be found, other steps need to be performed. First off, the articles must be retrieved and the content of the articles must be filtered and stemmed to bring morphological variants back to their same stem. With the Dutch Stemming algorithm package the content is to be stemmed. Porter's algorithm is based on a series of steps that each remove a certain type of suffix by way of substitution rules. The original Porter's algorithms has only suffix rules. Because most Dutch past participles contain the pre- or infix 'ge', the algorithm is extended by pre- and infix rules. Most of the rules are based on the measure of a word. This measure is the number of vowel-consonant sequences, where double vowels or consonants are counted as one. In formulating the affix rules for the algorithm several criteria were taken into account. These are the most important three:

1. Inflectional morphology should be covered as fully as possible. Inflectional affixes (-en, for plural or -e for an adjective) do not influence the basic meaning of the underlying stem, so they can always be removed.
2. Only those derivational affixes which do not substantially affect the information conveyed by the should be removed. Affixes like -heid can be remove but an affix like on- can't be removed.
3. The most frequent affixes should be removed. This algorithm only concentrates on removing the most frequent affixes.

For the Dutch Porter stemmer six clusters of rules were created. The rules inside a cluster are mutually exclusive, meaning that when the first matching rule in the cluster is applied no others are tried. The affix clusters are defined by the level at which the affix occurs. Inflectional affixes are removed before derivational suffixes. So the affixes are removed in a consecutive order. In addition to these clusters some conditions were designed to cover some specific Dutch language phenomena.

In their paper Kraaij & Pohlman (1994) describe the 'ENDSWITH V/C' condition, which means that each stem should end with a vowel consonant combination. An other condition is the 'DUP V' condition. In many cases in the Dutch language the stem has a double vowel while in the morphological variant there is only one as is in the case of 'lopen' and 'loop'. The rules which remove these affixes are marked for the 'DUP V' procedure where the vowel is doubled.

Using the porters stemming algorithm three types of errors can occur:

1. Linguistic Incorrect stems. Some stems that are incorrect can be generated. If this incorrect is unique there is semantically still no problem. But if the incorrect stem is identical to another stem, this will result in retrieving errors.

2. Homographs. There are word which are spelled identical but do not have the same semantical meaning.

3. Irregular verbs. Irregular verbs cannot be stemmed into the same stem with the algorithms.

It is desirable to count morphological variants of the same word as one word as they have the same semantical meaning. Porters Stemming Algorithm for Dutch, created by Kraaij & Pohlman (1994), is the best available algorithm for the Dutch language. They have also made a application to perform the stemming with. This is very useful to this project and therefore is used in the experiment.

4.2 Measuring the representativeness of words with TF-ISF

Having prepared and stemmed the articles database, the most representative words for each subject can be identified. For each subject the most representative words are selected to create the optimal queries with later on. The representativeness is measured with the TF-IDF measure. TF-IDF stands for Term Frequency - Inverse Document Frequency. This measure is originally used to measure the representativeness of a document to word, as relevant documents to a search word are retrieved. The measure can also be used the other way around to measure the representativeness of a word to a document. The Term Frequency is retrieved by counting the appearance of a word inside a document. But only the term frequency of a word in a document is not sufficient to measure the representativeness of a word. If a certain words appears a few times in a document, but also appears in every other document, it isn't very unique to that certain document and therefore wouldn't be very representative. The frequency of the word inside the other documents is the document frequency(DF). In the DF is calculated by counting how many times the word appears at least one time in each document from the entire document database. For a word to be representative for a document a high term frequency and a low document frequency are desired. Combining a positive performance measure for the term frequency and a negative performance measure for the document frequency gives us the Term Frequency - Inverse Document Frequency (TF-IDF) measure.

Since in this thesis I am interested in the representativeness of a word to a subject, I use the term frequency(how many times the word appears in the articles corresponding to the subject) inside a subject and a document frequency inside the other subjects(how many times the word appears at least one time in each of the other subjects).

4.2.1 Formula's

The original TF-IDF formulas as stated in the literature review are 4.1a and 4.1b. There the IDF is calculated by the inverse of the fraction between the total number of documents in the database by the document frequency as can be seen in 4.1a. The weight of document i to word w is then calculated by 4.1b.

$$IDF(w) = log\left(\frac{|D|}{DF(w)}\right) \tag{4.1a}$$

$$d^{(i)} = TF(w,d) * IDF(w_i) \tag{4.1b}$$

In this thesis I calculate the TF-IDF value for each word that appears at least in a subject, so for all word-subject combinations. Let's bring back the Subject-Word sparse matrix back to our attention(table 6.1). The Term Frequency is the value of $Subject_m$, $Word_n$ in the matrix. The Document Frequency is in this case called the Subject Frequency (SF) and is the number of values that are above zero in column $Word_n$. The $|D|$ is in our case the cardinality of all the subjects which is always 3194 in this project, denoted by $|Subjects|$. The Inverse Subject Frequency is calculated with formula 4.2a. From now on I call the measure the TF-ISF measure. I am calculating the weight of a $Word_n$ to a $Subject_m$ with formula 4.2b.

$$ISF_{n,m} = log\left(\frac{|Subjects|}{SF(n)}\right) \tag{4.2a}$$

$$TF - ISF_{n,m} = TF_{n,m} * ISF_{n,m} \tag{4.2b}$$

4.3 Measuring the performance of queries with the F-score

Having the most representative words for each query, the finding of optimal queries can start. The performance of a query is measured with a measure to combine precision and recall. The harmonic mean of both, with β as recall preference weight, is the $F - score_\beta$. As stated in the related work section I have no grounded arguments to apply a preference to either recall or precision so I use the $\beta = 1.0$ F-score variant where both are weighted equally.

4.3.1 General F-score formula's and formula's used in this project

In the general formula's for precision, recall and the $F - score_1$, the relevant data set is called 'rel' for relevant. The data set that is retrieved by a search is called 'ret' for retrieved. This bring us to formula's 4.3.

$$Precision = \frac{|ret \cap rel|}{ret} \tag{4.3a}$$

$$Recall = \frac{|ret \cap rel|}{rel} \tag{4.3b}$$

$$Fscore_\beta = (1 + \beta^2) * \frac{Precision * Recall}{\beta^2 * Precision + Recall} \tag{4.3c}$$

$$Fscore_{1.0} = \frac{2 * Precision * Recall}{Precision + Recall}, \beta = 1.0 \tag{4.3d}$$

Now let's image an article set S that corresponds to subject S and a article set Q that corresponds to query Q. That gives us two sets, set S and set Q. Article set Q is supposed to retrieve as much articles as possible from the article set S. So article set S is the relevant set 'rel' and the article set Q is the retrieved article set 'ret'. The F-score to measure the performance of query Q to retrieve the articles from S with is then calculated with the formula's from 4.4.

$$Precision = \frac{|Q \cap S|}{Q} \tag{4.4a}$$

$$Recall = \frac{|Q \cap S|}{S} \tag{4.4b}$$

$$Fscore_{1.0} = \frac{2 * Precision * Recall}{Precision + Recall}, \beta = 1.0 \tag{4.4c}$$

Chapter 5

The Evolutionary Approach

5.1 Introduction

To solve the article set covering I use the evolutionary approach in the form of Genetic Programming. Beasley & Chu (1996) state that the evolutionary approach can be understood as an 'intelligent' probabilistic search algorithm which can be applied to a variety of combinatorial optimization problems. It is based on the evolutionary process of biological organisms in nature. In evolution the fittest individuals are more likely to survive as they adapt to their environment. The weakest individuals are more likely to die. As generations evolve the best genes will increasingly be passed on to the next generations, creating even more fit offspring.

There is to choose between a Genetic Algorithm and Genetic programming. The main difference is that in GA the solution size is fixed and in GP it is not. An initial population is randomly generated with individuals having a chromosome with the possible solution. The fittest individuals, the best possible solutions, are more likely to reproduce, generating the next population. Fit individuals reproduce using crossover, where from two parents characteristics are passed on to their children creating the next generation. This process repeats until a satisfactory solution is found or an end criteria is met.

```
Algorithm 5.1.1: Basic Genetic Algorithm($GA$)

$GenerateAninitialPopulation$
$EvaluatefitnessOfindividualsinthePopulation$
repeat
  $SelectParentsfromthePopulation$
  $Recombine(mate)parentstoproducechildren$
  $EvaluatefitnessOfthechildren$
  $Replacesomeorallofthepopulationbythechildren$
until $aSatisfactorySolutionhasbeenfound$
```

John, et al. (2005) use the genetic programming method to solve a formula finding problem. These programs can vary in their length which was the problematic limitation of the Genetic Algorithm to our problem.

5.2 The tree like program representation

John, et al. (2005) represent the possible solutions as trees. A tree consist out of functions and terminals. Figure 5.1 is the tree program for a formula representation. The terminals are independent variables or constants (the x, y and 3 in the figure). These are the leaf nodes. The

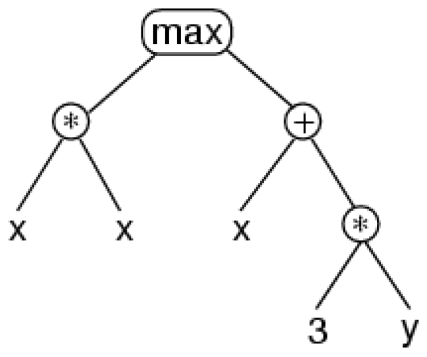

Figure 5.1: Genetic Programming program tree representation

functions are here the mathematical operators(*, +, max). These are the internal nodes. In the tree representation a sub-tree is called a branch and the tree structure is called the architecture.

In our query optimization problem the list of most representative words for each subject would be the terminals and the AND / OR operators would be the functions. This is visualized by a sample query tree of the Subject 'Europese Unie' (European Union) in figure5.2.

The query represented by the tree is: (europees OR ((eu AND brussel) OR lidstaat)).

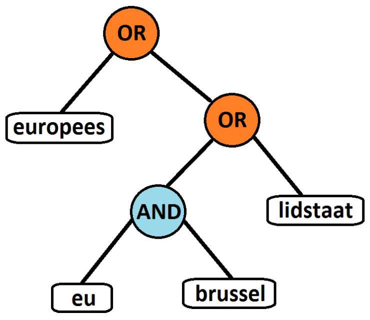

Figure 5.2: Genetic Programming program tree representation, Query

Interpreting a program tree means executing the nodes in an order that guarantees that nodes are not executed before all arguments in their their statement are known. This is usually done by traveling the tree down from the root and postponing the executing until the values of the child nodes of a node are known. An example of this system is represented in figure 5.3.

5.3 Preparatory steps

The preparatory steps of Genetic Programming specify what information the user must provide in advance to the genetic programming system. Once the run has launched, the system runs in the same order for each problem. So Genetic Programming is problem independent. It is important that the preparatory steps are taken with care and precision. These are the five steps that need to specified.

1. the set of terminals
2. the set of functions

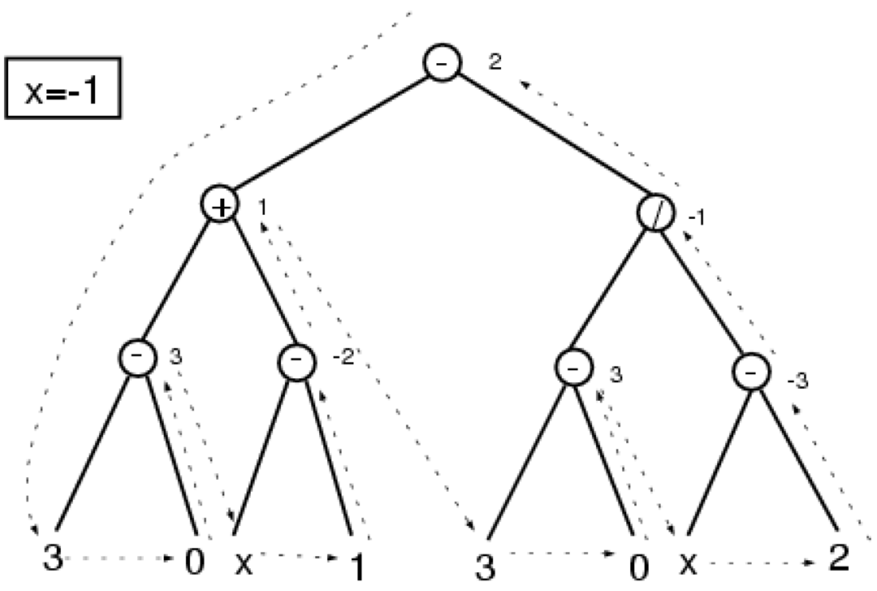

Figure 5.3: Genetic Programming Program tree interpretation

3. the fitness measurement
4. certain parameters for controlling the run
5. the termination criterion

Steps one and two are the word and two operators, AND and OR. The fitness functions is the F-score measure. The fourth step is to set certain parameters to control the run. These are parameters such as initial population size, probabilities for genetic operators (such as mutation, crossover and parent selecting), the maximum size for programs and others. The termination criterion may be a reached by a certain fitness measure or a number of generations or some other criterion depending on the problem. In the paragraphs below I elaborate on some parameter that need to be set to create a successful Genetic Programming environment.

Population replacement There are two options in population replacement. The first option is to replace the entire population by the offspring. This is called the generational approach. The second option is to replace the less fit individuals of the population by offspring. This is called the steady-state approach.

Parent selection techniques A popular parent selection technique is Tournament selection. Here two pools with size T are created with individuals form the population. From each pool, the fittest individual is chosen for crossover. This repeats many times depending on how many offspring is desired. With a small T for pool size, weak individuals have higher probability to reproduce as their chances reduce with large pool sizes.

Crossover operators The crossover operator is responsible for creating children from two parents. Often one or two crossover point operators are used. In one point crossover one point is chosen and the child gets the first half of the genes from parent one and the second half from parent two. Two point crossover works the same, but then with two point and two children as result. In this heuristic they use a generalized fitness-based operator called the *fusionoperator*, which takes the parent fitness into account. This operator decides for each gene individually which of the parents to copy based on probability based on the parent fitness. As we don't use this operator in the experiment we shall not go into it.

Variable mutation rate In Genetic Algorithms mutation is seen as a background operator to preserve variety and to prevent loss of information. After crossover mutation is applied to each child. Each bit changes with a small probability. In the heuristic of their paper Beasley & Chu (1996) use variable mutation rate. They use a lower bound rate of 1/n where n is the size of the population. As the Genetic algorithm converges the mutation rate goes up to keep variety.

Choosing the population size Considering a random generated Set Covering problem, generated by randomly selecting one column for each row and afterwards removing the not needed columns. Beasley & Chu (1996) provide a formula to calculate the initial population size the Genetic Algorithm should use.

First they consider the density which is notated by ϕ, which is the fraction of ones in the zero-one matrix (a_{ij}) (see table **??**). The average number of columns in each Solution S_p is $1/\phi$ columns. An initial population of N therefore has a an average of N/ϕ columns. The average number of appearances of a column in the solution in denoted as μ. μ must be at least one or greater depending on the Algorithm users preferences. To provides sufficient coverage the number of columns in the initial solution must be equal or greater than then this average number of appearances times the number of columns in the Set Covering problem(denoted by n). See formula's 5.1a and 5.1b.

$$N/\phi \geq \mu n \tag{5.1a}$$

$$N = \mu\phi n \tag{5.1b}$$

5.4 Running the Genetic Programming

As stated Genetic Programming is problem independent. This means that the execution runs following the same Genetic Programming flowchart for each problem. See the flowchart of genetic programming in figure 5.6. In this flowchart, N is the number of desired runs that the Genetic Programming should execute. M is the size of the population. To explain the flowchart we have divided it into 4 stages A, B, C and D, as you can see in the figure. In stage A the first run and first generation are initialized. In stage B, firstly is checked wether the Termination criterion is met. If not, the fitness measure is applied to each individual in the population. In stage C, one of the four Genetic Operations are applied until a new generation of size M is generated. These are the operators:

- Reproduction(P_r): An individual from the previous population is reproduced into the next generation
- Crossover(P_c) : New offspring programs are created from two parents, selected based on fitness (for example with the tournament method) (see figure 5.4)
- Mutation(P_m) : One child is created by randomly mutating a randomly chosen individual based on fitness (example figure 5.5)
- Architecture altering(P_a): One individuals architecture is changed and put into the next generation

When stage C is finished creating a new generation, stage B is entered again. When the termination criterion is met in stage B, stage D is entered. When the desired number of runs is performed the Flowchart is finished.

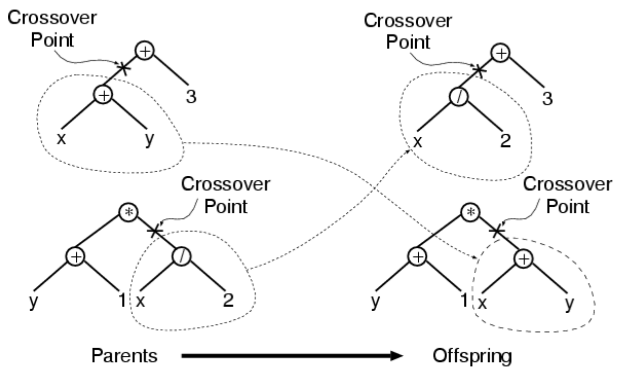

Figure 5.4: Genetic Programming crossover operator

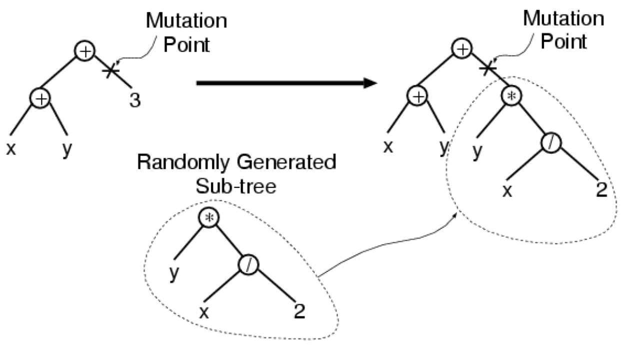

Figure 5.5: Genetic Programming mutation operator

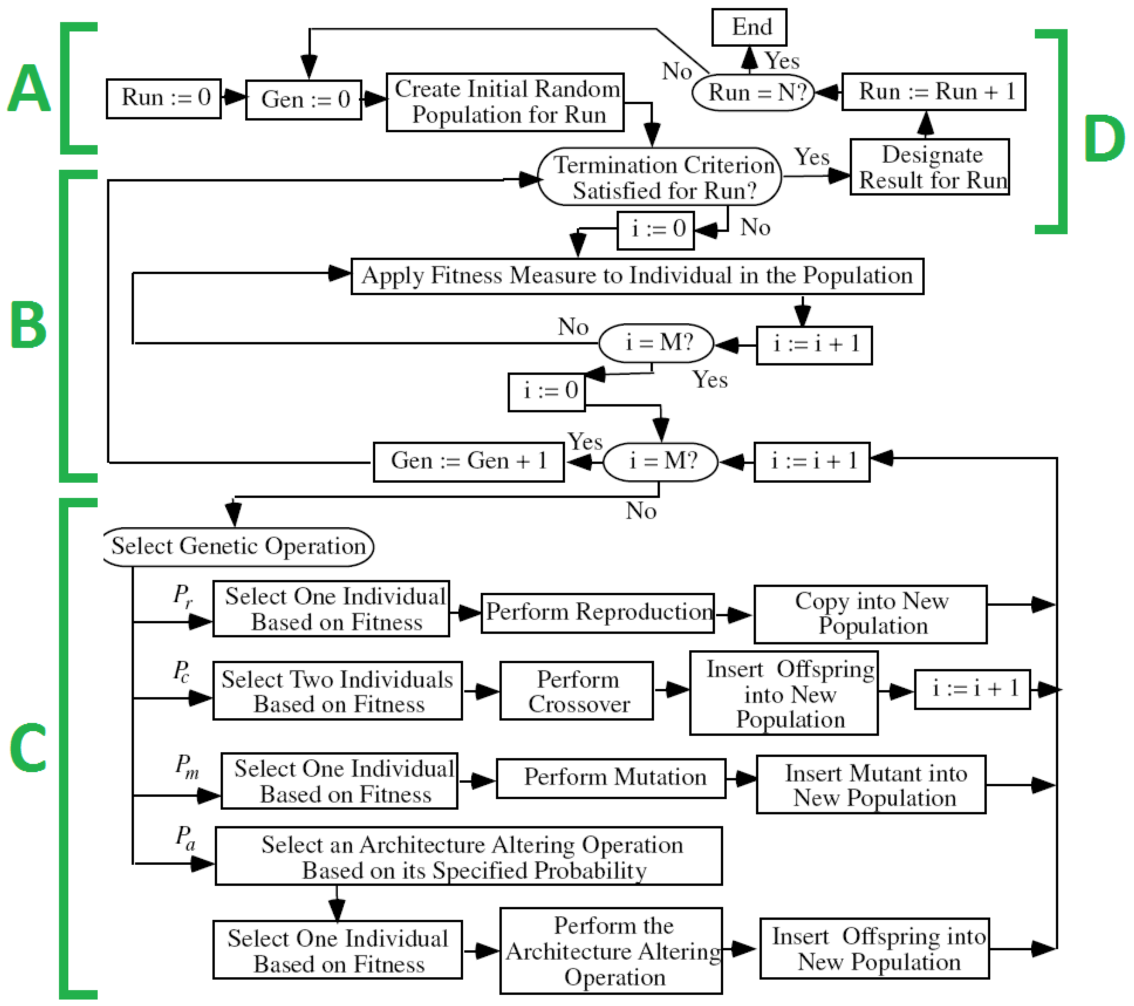
A
B
C
D
Run := 0
Gen := 0
Create Initial Random Population for Run
End
Yes
No
Run = N?
Run := Run + 1
Termination Criterion Satisfied for Run?
Yes
Designate Result for Run
i := 0
No
Apply Fitness Measure to Individual in the Population
No
i = M?
i := i + 1
Yes
i := 0
Yes
Gen := Gen + 1
i = M?
i := i + 1
No
Select Genetic Operation
P_r
Select One Individual Based on Fitness
Perform Reproduction
Copy into New Population
P_c
Select Two Individuals Based on Fitness
Perform Crossover
Insert Offspring into New Population
i := i + 1
P_m
Select One Individual Based on Fitness
Perform Mutation
Insert Mutant into New Population
P_a
Select an Architecture Altering Operation Based on its Specified Probability
Select One Individual Based on Fitness
Perform the Architecture Altering Operation
Insert Offspring into New Population

Figure 5.6: Genetic Programming basic flowchart

Chapter 6

Finding Optimal Queries

6.1 Overview

6.1.1 Steps in the Experimental Design

The experiment of this project is performed step by step. Important is to note that the data that is produced during every step is stored separately so that in the process earlier steps can be performed again if some settings need to be changed. The process of performing the steps sequentially and repeating them when desired is visualized in the flowchart in figure 6.1.The experiment consist out of the following steps.

1. Retrieve articles and taxonomy from the MD Info database
2. Prepare, filter and apply stemming to articles
3. Generate Subject-Word, Subject-Article and Word-Article sparse matrices
4. Find representative words by calculating TF-ISF values
5. Find optimal queries using Genetic Programming
6. Show Results

6.1.2 Some technical project information

The application that performs the steps described in this experimental design is programmed with Java. The Stemming package that is used to stem the article content is programmed with ANSI C by Kraaij & Pohlman (1994). The articles are saved into XML format. The treated content of the articles is also saved into xml format. The sparse matrices are saved into txt files.

6.2 Retrieve articles and taxonomy from the MD Info database

The first step of this experiment is to retrieve the entire article database and the taxonomy from the MD Info site. The taxonomy is stored inside the Java code for quick access. The article database was downloaded on the 28^{th} of May 2010. The Java application was used to log into the MD Info site, with the login account that was provided to me by the company. At the site I manually created a profile(query) which searched for all subjects in all news sources. The application was used to download all articles inside that profile and store them into XML files. In total 330721 articles were downloaded. I chose to save the articles per thousand into 331 separate XML files.

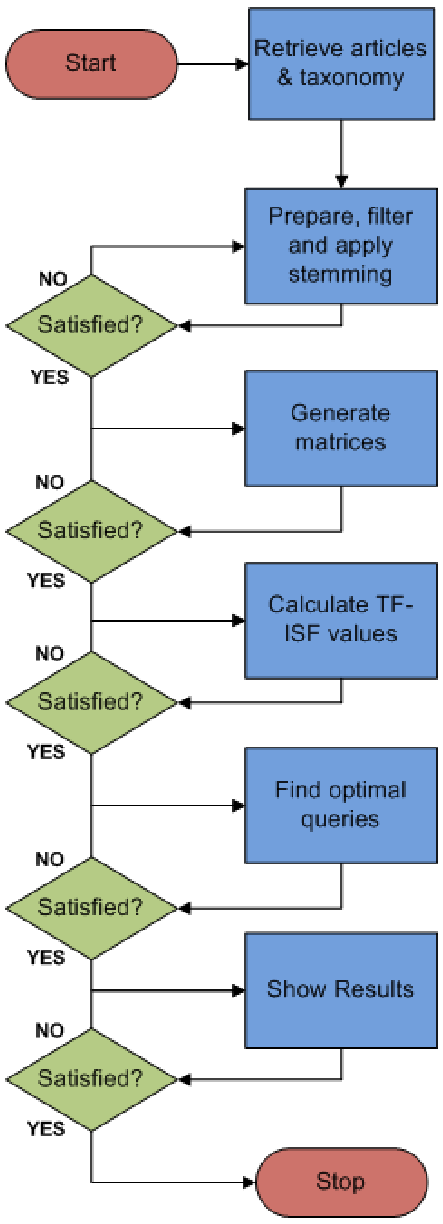

Figure 6.1: Subject X article set

6.3 Prepare, filter and apply stemming to articles

```
Algorithm 6.3.1: FILTER ARTICLE CONTENT($filter$)

repeat
  $Selectarticle$
  for each $(content)line$
    $transformlineintolowercase$
    $removeallreadingsigns$
    $removeallnumbers$
    $fixallerrorletters$
    for each $word$
      $lookatstopwordslist$
      if $inlist$
        $removeword$
until $Allarticleshavebeenfiltered$
```

After downloading the article database some content preparing need to be done. The database is checked for double articles and any double articles are removed from the database. Saving the content into XML format with Java brought some errors with letters with an accent. These errors are fixed. All the reading signs and numbers are deleted. All content is transformed into lower case. Having done this, the content is ready to be stemmed.

6.3.1 Article storage in XML format

Since we are working with a large article database, the storage methods are important for the success of the entire project. The articles are saved in XML format. This is because XML is very user friendly, human understandable and good to work with, with the programming language Java, in which the experiment is build. See how an example article is saved into XML.

Listing 6.1: sample: "articles1.xml"

```
<Articles>
    <Article>
        <Aid>935280</Aid>
        <Title>Amerikanen gebruiken smartphone vaak voor
        internettoegang</Title>
        <Content>
            **article content**
        </Content>
        <Subject>
            <SubjectName>Internet</SubjectName>
            <SubjectOid>1704</SubjectOid>
        </Subject>
        <Subject>
            <SubjectName>Mediagedrag, mediaconsumptie</SubjectName>
            <SubjectOid>528</SubjectOid>
        </Subject>
        <Subject>
            <SubjectName>Mobiele communicatie</SubjectName>
            <SubjectOid>1699</SubjectOid>
        </Subject>
        <Subject>
```

```
            <SubjectName>Verenigde Staten</SubjectName>
            <SubjectOid>5</SubjectOid>
        </Subject>
    </Article>
<Article>
```

6.3.2 Stemming the article content

Building a Dutch Stemming application myself would be too much work and would be way outside the scope of this thesis. Luckily Kraaij & Pohlman (1994) have already build a Dutch Stemming algorithm. One downside is that the stemmer is written in the programming language ANSI C, which is quite unfamiliar to me. The original stemming package is able to read a test input file and to print all separate words stemmed onto the screen. I have rewritten this code with a loop to read in a folder with input txt files and to output a folder with txt files with the corresponding stemmed content.

The Java application creates suitable input files, with one word per line, with start and stop codes to separate the articles. The Stemming package transforms those files into stemmed output. Then the Java application is used to transform the stemmed content back into the XML format. This process is visualized by figure 6.2

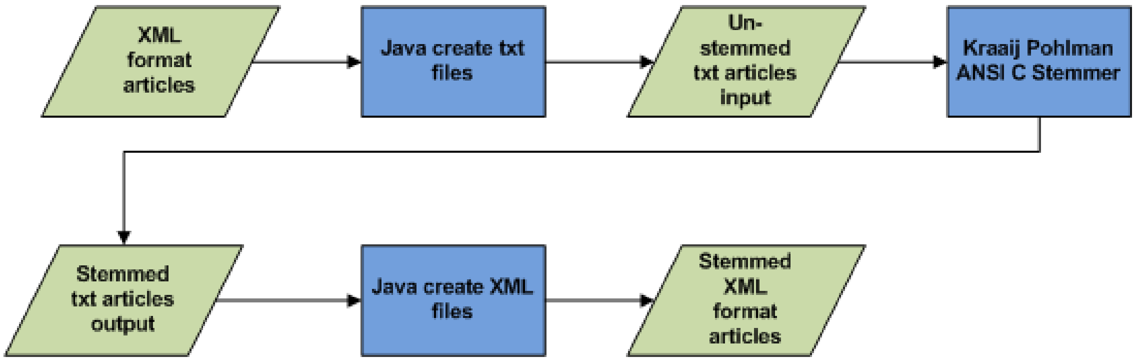
XML format articles
Java create txt files
Un-stemmed txt articles input
Kraaij Pohlman ANSI C Stemmer
Stemmed txt articles output
Java create XML files
Stemmed XML format articles

Figure 6.2: Experiment flowchart

6.4 Generate Subject-Word, Subject-Article and Word-Article sparse matrices

6.4.1 Sparse matrix storage

After the previous step is done, the entire article database is ready for analysis. To calculate the TF-ISF values I need to know how many times each word appears in a subject. Therefore the Subject-Word sparse matrix is created. To evaluate query performance during the Genetic Programming I need to know in which articles belong to each subject and in which articles each word appears. Therefore the Subject-Article and Word-Article sparse matrices are needed. The desired Subject-Word matrix is visualized in table 6.1.

The Subjects are on the Y-axis and the Words are on the X-axis. Both all the subjects(3194) and all the words that have appeared in the entire database(488658) are saved as a list in a txt file. The list of Subjects and the list of Words that correspond with the sparse matrix axes are represented in tables 6.2 and 6.3. If a word appears one or more times in a subject this is saved into the sparse matrix in the following format: "subjectindex, wordindex, value", this is visualized in table 6.4. For each value in that sparse matrix, the TF-IDF value is calculated. These values are saved in a similar format, double sparse matrix, since multiple decimal values are calculated.

	Words(n)					
Subjects(m)	0	1	2	3	..	488658
0	-	-	2	-	-	-
1	-	-	-	-	-	1
2	-	1	-	-	-	-
..	-	-	-	-	-	-
3194	-	-	-	-	53	-

Table 6.1: Subject-Word integer sparse matrix

Index	**Subject**
0	Adviserende organisaties/commissies van de rijksoverheid (WRR, SCP, CPB)
1	Belastingdienst
2	Brandweer
..	..
3194	Voedsel en Waren Autoriteit (VWA)

Table 6.2: Y-axis Subject indices(in Subject-Word and Subject-article matrices)

Index	**Word**
0	amerikaan
1	bruik
2	smartphone
..	..
488658	sleuteltekst

Table 6.3: X-axis Word indices(in Subject-Word sparse matrix), Y-axis Word indices(in Word-Article sparse matrix)

0,465,4
0,6458,1
0,6459,1
0,6460,2
0,6461,3
...
1649,953,11
1649,1858,10
1649,1003,41
1649,7308,1
1649,10590,102
...

Table 6.4: Subject-Word integer Sparse matrix storage format

	Articles(n)					
Subjects(m)	0	1	2	3	..	330721
0	-	-	true	-	-	-
1	-	-	-	-	-	true
2	-	true	-	-	-	-
..	-	-	-	-	-	-
3194	-	-	-	-	true	-

Table 6.5: Subject-Article boolean sparse matrix

	Articles(n)					
Words(m)	0	1	2	3	..	330721
0	-	-	true	-	-	-
1	-	-	-	-	-	true
2	-	true	-	-	-	-
..	-	-	-	-	-	-
488658	-	-	-	-	true	-

Table 6.6: Word-Article boolean sparse matrix

The articles appearing in subjects are stored in a Subject-Article sparse matrix and the word appearance in articles is stored in a Word-Article sparse matrix. These two matrices are true/false boolean matrices. The desired sparse matrices are visualized in tables 6.5 and 6.6, the X and Y axes of the sparse matrices are visualizes in tables 6.2, 6.3 and 6.7 and both matrices are stored as in table 6.8. Since all the saved values are 'true' values only the coordinates need to be saved in the following formats: "subjectindex, article index" and "wordindex, articleindex".

Index	**Article Aid**
0	935280
1	935329
2	935331
..	..
330721	206405

Table 6.7: Y-axis Article indices (in Subject-Article and Word-Article matrices)

0,1465
0,3458
0,4459
0,7460
0,68461
...
1649,93
1649,18
1649,103
1649,7308
1649,30090
...

Table 6.8: Subject-Article boolean Sparse matrix storage format, the Word-Article sparse matrix is saved in the same format

6.4.2 Sparse matrix creating process

A this point we already have the Subject list. The Word list and Article list still need to be generated at this point. The words and articles are written to the Word list and Article list in order of appearance inside the database. The Java application runs once through all the articles of the entire articles database, creating the word and article lists on the go, and saving all Subject-Word, Subject-Article and Word-Article combinations into sparse matrices. See below the process in which the matrices are created.

```
Algorithm 6.4.1: CREATE SPARSE MATRICES($sparsematrices$)

repeat
 $SelectArticle$
   $AddArticletoArticlelist$
   $returnArticleindex$
   for each $Word$
       if $WordisnotinWordlist$
       $addWordtoWordlist$
       $returnWordindex$
         else $returnWordindex$
     $addWord, Articlecombinationtosparsematrix$
   for each $Subject$
     $addSubject, Articlecombinationtosparsematrix$
     for each $Word$
     $add + 1toSubject, Wordcombinationofthesparsematrix$
until $AllArticlesareanalyzed$
```

6.5 Find representative words by calculating TF-ISF values

In the previous step the Subject(m)-Word(n) sparse matrix is retrieved. All the non zero values, the TF-ISF value can be calculated using formula 4.2b from the analysis method. The TF-ISF values are saved in a double sparse matrix. For each subject, the highest TF-ISF scoring words are selected from that matrix as the most representative words for the Genetic Programming procedure. The subjects, and the corresponding representative words are saved into a XML file.

6.6 Find optimal queries using Genetic Programming

6.6.1 Query optimization problem

Now that the most representative words for each subject are retrieved, the finding of the optimal queries can start. Before I go into detail with the Genetic Programming I want to visualize the query optimization problem. The articles that correspond to each subject are saved in the Subject-Article sparse matrix. The articles that correspond to each word are saved in the Word-Article sparse matrix.

Each subject has a corresponding article set. Let us call the article set that corresponds to the subject article set S. For each of the 40 words there is a corresponding article set of articles where that word appears in. Let us call the article set that corresponds to the $Word_n$ articles set W_n. Now there is set S which must be covered as accurate as possible with the optimal AND / OR combinations of sets W_1, W_2 until W_{40}. In figure 6.3 the articles set S is visualized. In figure 6.4, articles sets $W_{1,2,..,40}$ are visualized as well. The optimal combination of the W sets must be found to cover as much of the articles set S as possible and at the same time cover as less as possible outside the set S area.

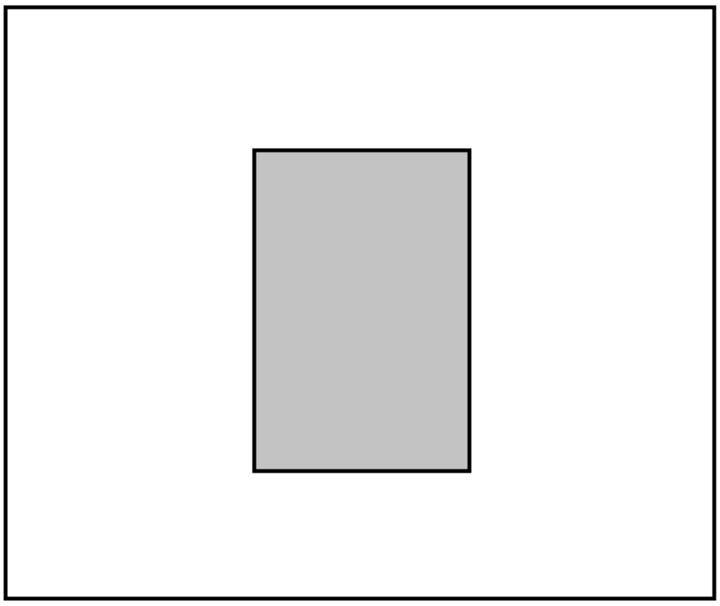
Figure 6.3: Articles set S, articles corresponding to the subject

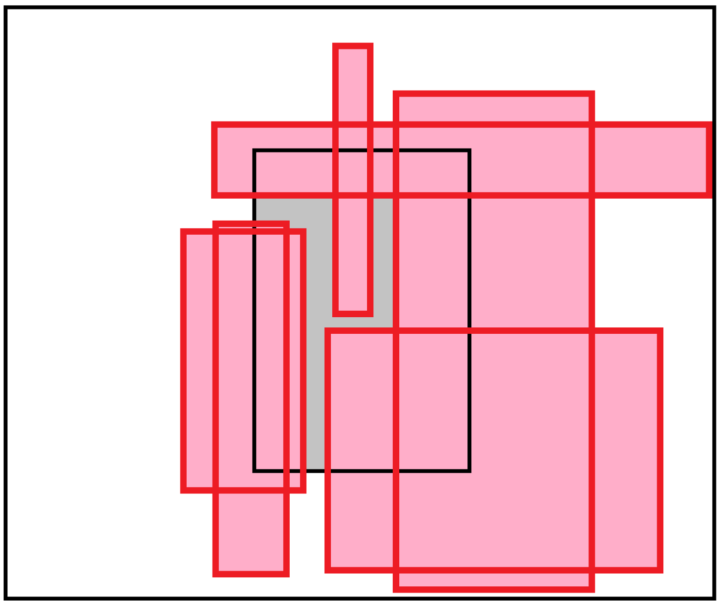
Figure 6.4: Set S, covered by articles sets $W_{1,2,..,40}$

6.6.2 Genetic Programming

The Genetic Programming was set with the Java Programming language. For each non-empty subject Genetic Programming is performed to find the optimal query. The setup work following the flowchart in the Evolutionary approach chapter. The Programming is done with the 20 most representative word for the Subject. The first population is set up with 400 individuals, namely the 20 individuals, all 190 combinations of two words with the OR operator and all 190 combinations of two words with the AND operator. The fitness of each tree is represented by the f-score of the corresponding query.

After the initial population a population size of 100 is chosen. Each tree representation of a query, after the first crossover is of size 2 or larger. This means on average each word appears 10 times or more in a population. This should be sufficient. Then three operators are used. The elitism operator simple copies ten individuals with the highest unique fitness into the new Population.

The crossover operator is used to combine the two parent and to create one child. Tournament selection is used to chose two parents. A pool size of 10 is chosen, which is a tenth of the population. The child is created by copying the tree of the first parent and replacing a random chosen node(except for the root node) with a randomly chose node from the second parent(here the root node is an option). The child is put into the next generation.

The third operator is the mutation operator. The mutation rate is chosen as a variable. As the generation count get higher the mutation count also gets higher with an upper limit of ten.

The fourth operator wasn't used.

Each generation of hundred was created by ten times elitism, zero to ten times mutation and the rest by crossover. A run is ended by the discovery of a Query of fitness 1.0 or by a same individual for at least 5 generations after 8 generations.

This was repeated for all the Subjects.

Chapter 7

Results

After the experiment is finished for each subject, except for the subjects with no articles, the optimal query and the corresponding f-score is known. We have performed the Genetic Programming to find the optimal query to retrieve the subjects inside their Sub Heading, inside their Main Heading and inside the entire database. In this chapter we shall show and explain the results from several perspectives.

7.1 F-score thresholds

To show the relation between some variables and the f-score we have set f-score thresholds from 0.0 to 1.0. The relation between the f-score and the number of subjects, the average number of articles and the average query length is discussed in the subsections below.

7.1.1 Relation between Subjects and F-score

The first relation is that between the number of Subjects and the F-score. Table 7.1 and figure 7.1 show the number of Subjects that correspond to each F-score threshold. In the figure we see the three lines, one for the optimal query inside the total database, one for the optimal query inside the main heading and one for the optimal queries inside the sub heading. The figures for Sub and Main Heading decrease slowly at the start as they decrease steeper later on. For 2604 subject, the optimal query was found by the Genetic Programming Experiment. 142 Subjects were fully covered inside the total database, 187 inside their main heading and 207 inside their sub heading, with an f-score of 1.0. Setting a f-score of 0.7 gives 272 subjects inside the total database, 868 subjects inside their main heading and 1104 subjects inside their sub heading, that are satisfying retrieved by the text query.

7.1.2 Relation between the number of articles and F-score

Another interesting relation to pay attention to is that between the F-score and the number of Articles inside each Subject that passes that threshold. This is interesting to know because we Subjects to have a high f-score because subject content is typical and not because for example the nr of articles is very low. So a closer look to the relation between the fscore and the number of articles inside each is very useful. The relation between the f-score and the average number of articles in each passing subject is show in table 7.2 and graph 7.2. Looking at the graph one can see a clear turning point for all three of the figures. The turning point for the total database figure is earlier at an fscore threshold of 0.2 where the two others have their turning point between the 0.6 and 0.7 threshold. This graph shows that the fscore starts to have a really strong influence on the number of articles inside the subjects from a fscore of 0.6 for the retrieving inside the main- and subheading. Inside the total database the fscore starts having influence on the number of articles from of fscore of 0.2.

F-score	Subs total	Subs MH	Subs SH
0.0	2604	2604	2604
0.1	2497	2582	2595
0.2	2111	2444	2544
0.3	1617	2191	2405
0.4	1168	1922	2203
0.5	785	1570	1903
0.6	479	1166	1498
0.7	272	868	1104
0.8	209	617	736
0.9	145	311	352
1.0	142	187	207

Table 7.1: The number of Subjects for F-score thresholds in the total database, inside the main heading and inside the sub heading

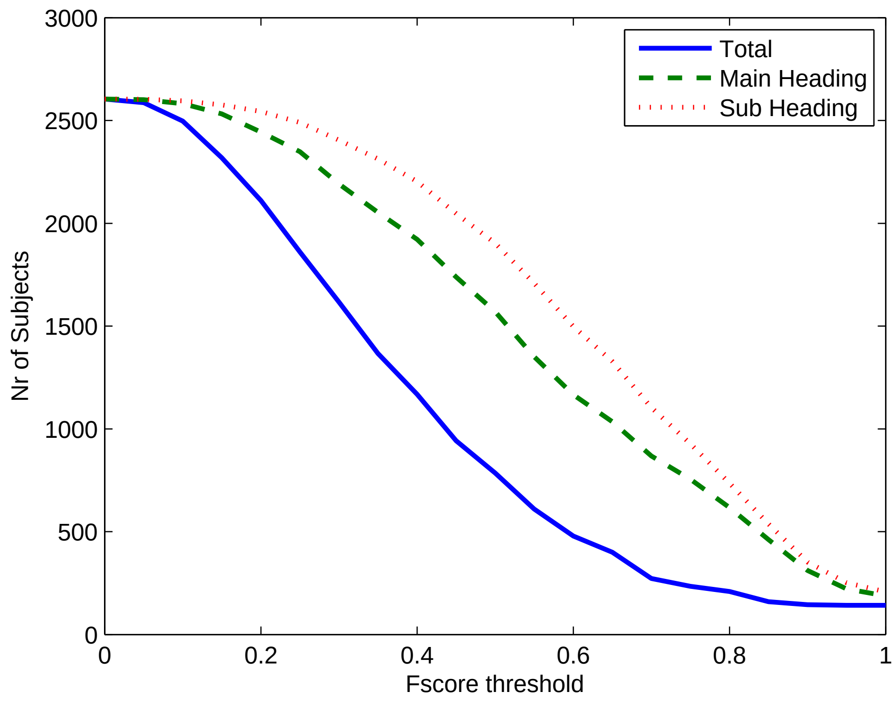

Figure 7.1: The number of Subjects for F-score thresholds

7.1.3 Relation between the query length and the F-score

The length of the optimal queries can vary starting at a size of one word to any higher number. It is insightful to study the relation between the F-score and the query length. The relation between the f-score and the average query length of the passing subjects is show by table 7.3 and figure 7.3. The average query length of Subjects is around 3.30 for the 2604 subjects. This decreases to between 2.59 and 2.01. The number seems quite small. The largest text query consists out of 12 words. And looking at the query sizes, this means that there are a lot of optimal query of size 1 and 2 pulling down the average.

F-score	Av.arts total	Av.arts MH	Av.arts SH
0.0	326	326	326
0.1	327	327	326
0.2	319	333	329
0.3	280	332	333
0.4	237	323	333
0.5	193	324	337
0.6	118	310	336
0.7	90	258	303
0.8	10	172	223
0.9	1	43	100
1.0	1	2	2

Table 7.2: The average number of Articles inside the Subject for F-score thresholds in the total database, inside the main heading and inside the sub heading

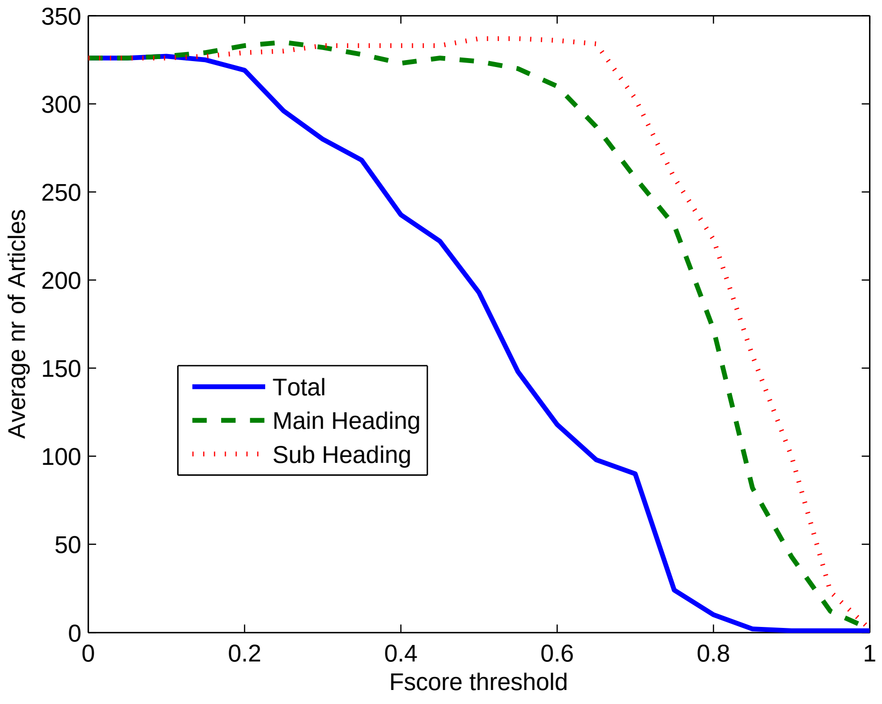

Figure 7.2: The number of Articles inside a Subject for F-score thresholds

7.2 Dutch and Foreign companies

Inside the taxonomy their are two Main Headings that stand out in retrievability. These are the headings 'Nederlandse bedrijven'(ducth companies) and 'Buitenlandse bedrijven'(foreign companies). Inside the main heading the average fscore of the subjects of the dutch companies is 0.8 and inside the total database it is 0.64. Inside the main heading the average f-score of the subjects of the foreign companies is 0.8 and inside the total database it is 0.54. Because these subjects contain all news about one company it is logic that these subjects are better retrievable than the rest.

F-score threshold	Av.Ql total	Av.Ql MH	Av.Ql SH
0.0	3.27	3.20	3.30
0.1	3.26	3.20	3.30
0.2	3.28	3.18	3.29
0.3	3.26	3.12	3.24
0.4	3.20	3.04	3.20
0.5	3.15	2.93	3.11
0.6	3.12	2.78	2.98
0.7	3.00	2.63	2.76
0.8	2.83	2.44	2.50
0.9	2.62	2.25	2.17
1.0	2.59	2.26	2.01

Table 7.3: The average Query length of a Subject for F-score thresholds in the total database, inside the main heading and inside the sub heading

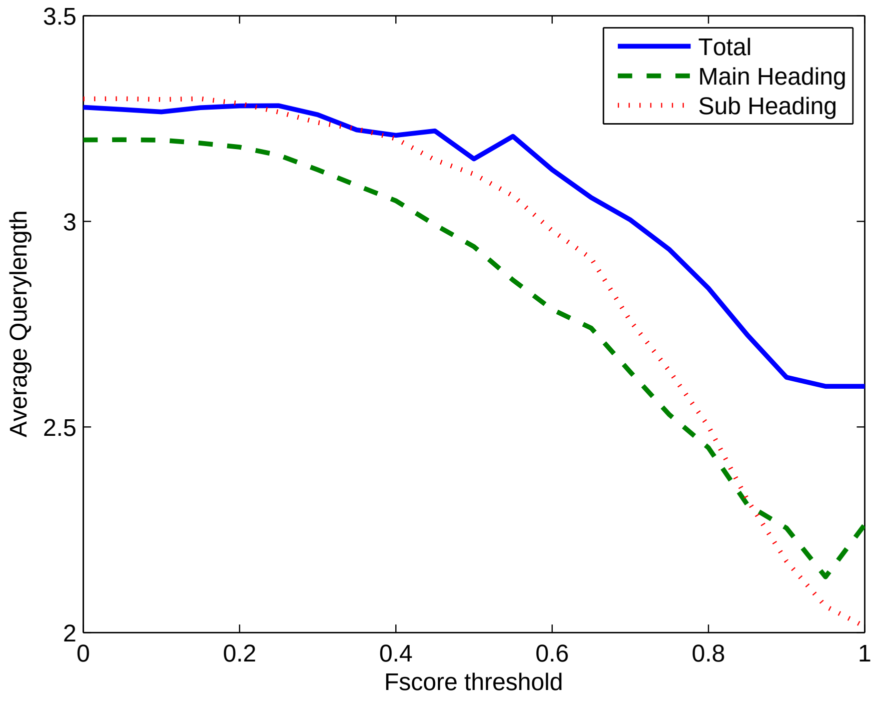

Figure 7.3: The number of Articles inside a Subject for F-score thresholds

7.3 Subjects of Interest

In the previous section we have seen the relationship between the fscore and the nr of subjects, the average number of articles and the average querylength for retrieval inside the entire database, inside their main heading and inside their sub heading. We have seen a clear relation the f-score and the nr articles inside a subject. Want we don't want to do is mark a subject as retrievable because there are only one or two articles inside the subject. We also want the separate the dutch and foreign company headings from the rest as it is logic that these subjects are well retrievable. So from here on we are interested in subjects that not inside the dutch or foreign company heading and in subjects that contain at least 20 articles. When a subject then also has an f-score of at least 0.7 we can call it a subject of interest.

Category	Nr of Subjects
Empty Subjects	590
Dutch or Foreign companies	859
With less than 20 articles	262
Remaing subjects	1483

Table 7.4: Subject categories table

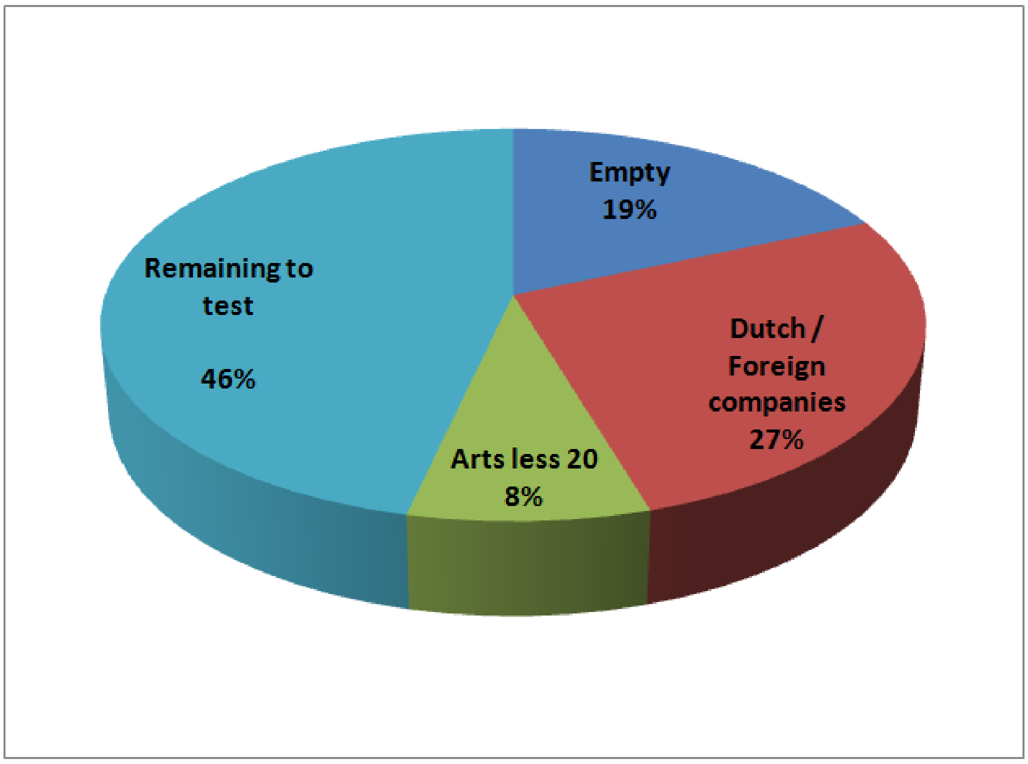

Figure 7.4: Pie chart of subject categories

To provide a clear view we have categorized the subjects. This can be seen in table 7.4 and pie chart 7.4. There are in total 3194 subjects inside the taxonomy. 590 of them have no articles, 859 of them are inside the dutch and foreign company headings and 262 of them have less than 20 articles. This leaves us 1483 subjects to put to the test.

7.4 Retrieval Results

The retrieval inside the total database wasn't all too successful. Out of the 1483 remaining subjects only 9 subjects were retrieved successfully which is 0.6 %. For retrieval inside the main headings 133 subjects were retrieved successfully which is 8.9% and for the retrieval inside the sub headings 315 subjects were satisfyingly retrieved which is 21.2%.

Chapter 8

Conclusions

8.1 Conclusions drawn from my Results

In this thesis we have searched for the optimal query for 2604 subjects of the MDInfo Taxonomy. The subjects are assigned a f-score corresponding to their optimal query. My conclusion is that a significant part of the subject can be seen as satisfying retrievable. in table 7.1 I've shown the relation between the F-score and the number of Subjects passing that f-score threshold. If one would take a 0.7 threshold 1104 subjects would pass.

Examining the results two other important things of influence came to attention. The first one is the relation between the number of articles inside the subjects and the f-score. We a subject to score high because the content is representable by a query, not because there is just one or articles in it. As show by figure 7.2, from a f-score threshold of 0.65 the number articles suddenly start decreasing very steep as the f-score rises. This show there is a clear relation between the number of articles and the f-score. Judging a subject on retrievability this must be taken into account.

The second thing is the occurrence two outstanding Sub-Headings namely 'Nederlandse bedrijven'(Dutch companies) and 'Buitenlands bedrijven'(Foreign companies). These headings store company related articles, making it no wonder the scores are so high.

Taking these two important thing into account, by leaving the two Sub Headings out and by leaving all Subjects with less that 20 articles out of the equation we had 1483 remaining subjects. By setting an 0.7 f-score threshold, we retrieved 0.6% inside the entire database, 8.9% inside the main headings and 21.2% inside the sub headings.

Looking at the results of this research MD Info could fuse or change some subjects inside the sub headings. We think that a retrievability percentage of 21.2 with an f-score of 0.7 isn't bad. The percentage inside the entire database and inside the main headings are too low to make any difference. This study also tells the decision maker how representative and unique the content is for each the subjects, which is helpful in deciding to change, leave out or fuse one subject with another or several subjects. Concluding, the retrieval results inside the sub headings can be used to bring down the number of subjects somewhat, but other methods should be used if they want to bring down the number more drastic.

8.2 Future work

I think having the TF-IDF, or TF-ISF measure as I call it in this thesis, of all the words to each subjects it appears in very valuable. I think that information could also be used for other purposes such as automatic classification, relating subjects to each other or searching subjects from customer side.

Bibliography

J. Beasley & P. Chu (1996). 'A genetic algorithm for the set covering problem'. *European Journal of Operation Research* pp. 392–404.

M. Gordon & M. Kochen (1989). 'Recall-Precision Trade-Off: A Derivation'. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* **40(3)**:145–151.

D. Harman (1991). 'How Effective Is Suffixing?'. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* pp. 8–15.

T. Joachims (1996). 'A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization'. *School of Computer Science Carnegie Mellon University* pp. 5–7.

G. P. John, et al. (2005). 'Chapter 5 GENETIC PROGRAMMING'. *Stanford University, Stanford, CA, USA and Department of Computer Science, University of Essex, UK* .

W. Kraaij & R. Pohlman (1994). 'Porter's stemming algorithm for Dutch'. *Wetenschappelijke bijdragen aan de derde STINFON Conferentie* pp. 167–180.

V. M. Mark Kantrowitz, Behrang Mohit (2000). 'Stemming and its effects on TFIDF Ranking'. *School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213 U.S.A.* pp. 357–359.

J. Ramos (2000). 'Using TF-IDF to Determine Word Relevance in Document Queries'. *Department of Computer Science, Rutgers University, 23515 BPO Way, Piscataway, NJ, 08855* pp. 1–4.

D. R.Musicant, et al. (2003). 'Optimzing F-Measure with Support Vector Machines'. *American Association for Artificial Intelligence(www.aaai.org)* pp. 356–360.

M. Rusinol & J. Llados (2009). 'A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices'. *Dept. Ciencies de la Computacio, Computer Vision Center, Edifici O, Universitat Autonoma de Barcelona,* pp. 84–96.

van Rijsbergen (1979). 'Information Retrieval'. *London; Boston. Butterworth, 2nd Edition 1979. ISBN 0-408-70929-4* .

J.-T. H. Wen-Chih Hung, Cheng-Yan Kao (1994). 'A Genetic Algorithm Approach for Set Covering Problems'. *Department of Computer Science and Information Engineering National Taiwan University, Taipei, Taiwan* pp. 569–574.

Glossary

F

f-score	The harmonic mean between precision and recall.

P

precision	In the field of information retrieval, precision is the fraction of retrieved documents that are relevant to the search.

Q

quey	A query can be defined as a request for information from a database.

R

recall	Recall in Information Retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved.
retrievable	A subject is retrievable when the article set that corresponds with the subject can be retrieved satisfyingly using a free text query and not the taxonomy subject.

S

stemming	The mapping of different morphological variants to their base form(stem).
stopword	words which are filtered out prior to, or after, processing of natural language data.

T

taxonomy	A taxonomy is a collection of entities organized into a hierarchical structure. Each entity in a taxonomy is in one or more parent/child (broader/narrower) relationships to other entities in the taxonomy.