DEVELOPMENT OF SEARCH ENGINE BASED ON VECTOR SPACE MODEL

SAADOUNE BILLAL

Development of search engine based on vector space model

2019 – 2020

DEDICATION

I dedicate this modest work:
To all members of my family for their continued support and I wish them good health and long life
To all my friends, especially to my friend Hadji Faiçal who has been my companion for the last years
To all my teachers who have done their best to give us as much information as possible about our study
Finally, to all those who have contributed in any way to the accomplishment of this work.

ACKNOWLEDGEMENTS

Above all, we take our thanks to our Almighty God for giving us the faith, strength and courage to complete this modest work

Many thanks to my supervisor Dr.LOUNNAS Bilal for his support, and advise.

I would like to thank all the people who have helped me directly or indirectly to the realization of this brief.

Thank you so much

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

GENERAL INTRODUCTION

This part contains the general introduction of the project and we talk about overview, motivation and problematic issues

1 GENERAL INTRODUCTION

1.1 OVERVIEW

Search engines are services that allow the user to search for content on the Internet and be in the form of a website, images, videos or other data. These engines are known as information retrieval systems.

Information retrieval systems are natural language processing fields that are defined as the process of obtaining information or resources from a collection of resources. There are also several models of information retrieval systems according to the method of information retrieval. For example, there are Boolean models in a exact retrieval method. There is also a vector space model in a retrieval method according to the best.

The vector space model is one of the most used models in the search engines so that the retrieval depends on the similarity or convergence between the documents and the user query.

1.2 MOTIVATION

Currently. Due to the huge increase in Internet users, developers have to improve the performance of applications available on the Internet. Among these improvements is improving the performance of search engines. In this context we would like to develop an information retrieval system that meets all the requirements of the user in good performance.

1.3 RESEARCH PROBLEMS AND CONTRIBUTIONS

The importance of information retrieval systems is the accuracy of results. Sometimes the results are different from what the user requested. For example, there may be a failure to identify the words close in meaning or there is ambiguity between the vocabulary or there is a difference in the spelling of terms in the same language from one country to another, we will respond to this problem, and we will develop information retrieval system depends on the model vector space to give us the best possible results.

1.4 ORGANIZATION OF MEMORY

Our memory is divided into four parts. The first contains the general introduction of the project and we talk about overview, motivation and problematic issues. The second part contains the litterateur topics to help the reader to understand the contribution of this project. This part contains two chapters: information retrieval and vector space model.

In the third part is the application part and contains one chapter in which the design and development of our system to retrieval information. The last parts is the general conclusion and future work

Part II

LITERATEUR PART

This part contains two chapters: information retrieval and vector space model

2 INFORMATION RETRIEVAL

2.1 INTRODUCTION

Efforts are being made to build a robust information retrieval system using models suitable for specific languages. It allows the management, storage, processing and retrieval of documents, informations and websites.

In this Chapter we identify the concept of information retrieval and its history. We also know the components of the retrieval system and how it is the process of information retrieval. With mention of the types of models used in this system.

2.2 INFORMATION RETRIEVAL

In this section we will learn about the definition of information retrieval and the history of this notion, and we will understand the process of retrieval of information and different models of retrieval.

2.2.1 *What is the IR*

Although just getting a credit card from your wallet so you can type out the card number it is considered a form of information retrieval. However, the retrieval of information can be very broad and can be used in the academic field of study.

We can also define information retrieval as follows:

Def1 ::

Information Retrieval is understood as a fully automatic process that responds to a user query by examining a collection of documents and returning a sorted document list that should be relevant to the user requirements as expressed in the query.

Def2::

The activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text indexing.

Def3::

The scientific discipline that deals with the representation, organization, storage and maintenance of information objects and in particular textual objects. The representation and organization of the information items should provide the user with easy access to the relevant information and satisfy the user's various information needs.

We have already seen three definitions, because the area of information retrieval is very large. Also we can find more definition of the field in the waste, but all refer to the same meaning is the process of retrieval of documents according to the needs of the user and the data must be represented with the index of all texts for easy access.

2.2.2 *The History of Information Retrieval Research*

The long history of information retrieval does not begin with the internet or with the invention of computers, it goes back to 9000 BCE represented by a simplest way of getting information, it began when people start to agriculture.

The following figure 1 will summarize IR history timeline:

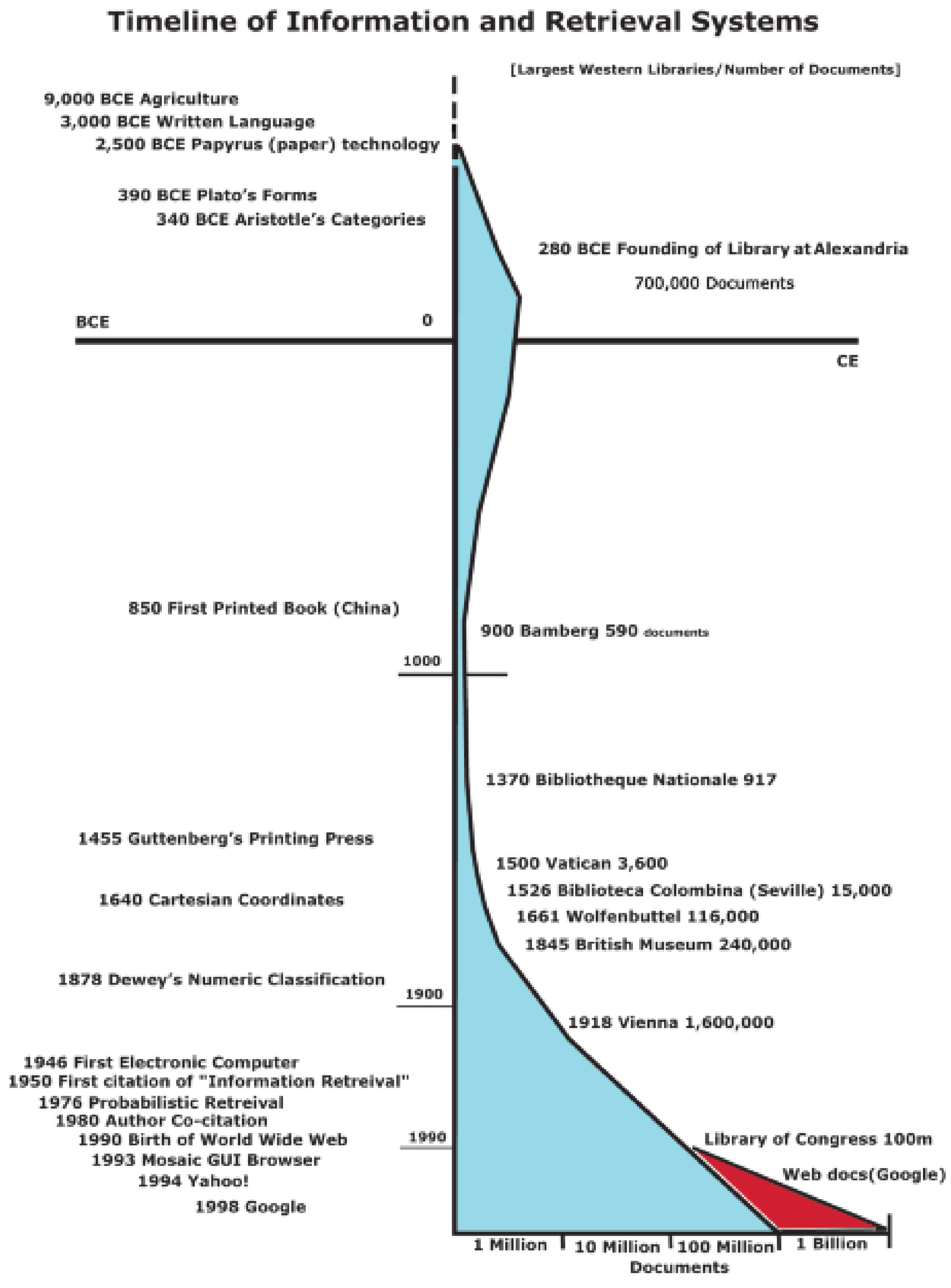


Figure 1: Information retrieval timeline history

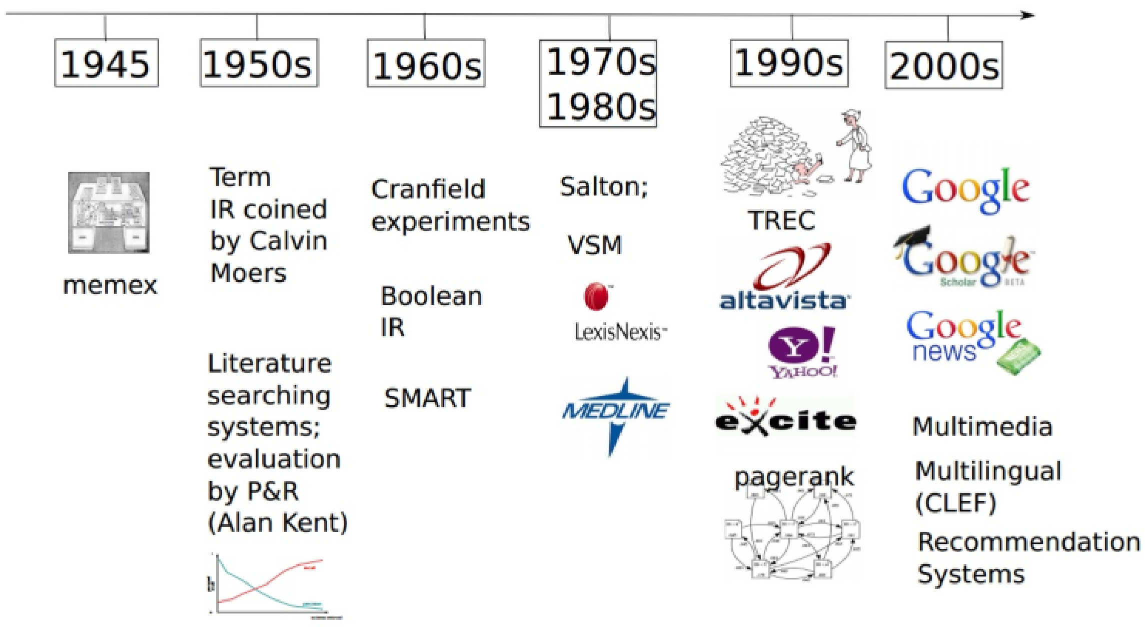


Figure 2: Information retrieval timeline history in the last decade.

- ïEarly days (late 1950s to 1960s): foundation of the field
  - Luhn's work on automatic indexing.
  - everdon's Cranfield evaluation methodology and index experiments.
  - Salton's early work on SMART system and experiments.
- 1970s-1980s: a large number of retrieval models
  - Vector space model
  - Probabilistic models
- 1990s: further development of retrieval models and new tasks
  - Language models
  - TREC evaluation
  - Web search
- 2000s-present: more applications, especially Web search and interactions with other fields â Learning to rank
  - Scalability (e.g., MapReduce)
  - Real-time search

2.2.3 *An information retrieval system*

The diagram in figure 3 shows the form of a three-component information retrieval system: input, processor, and output:

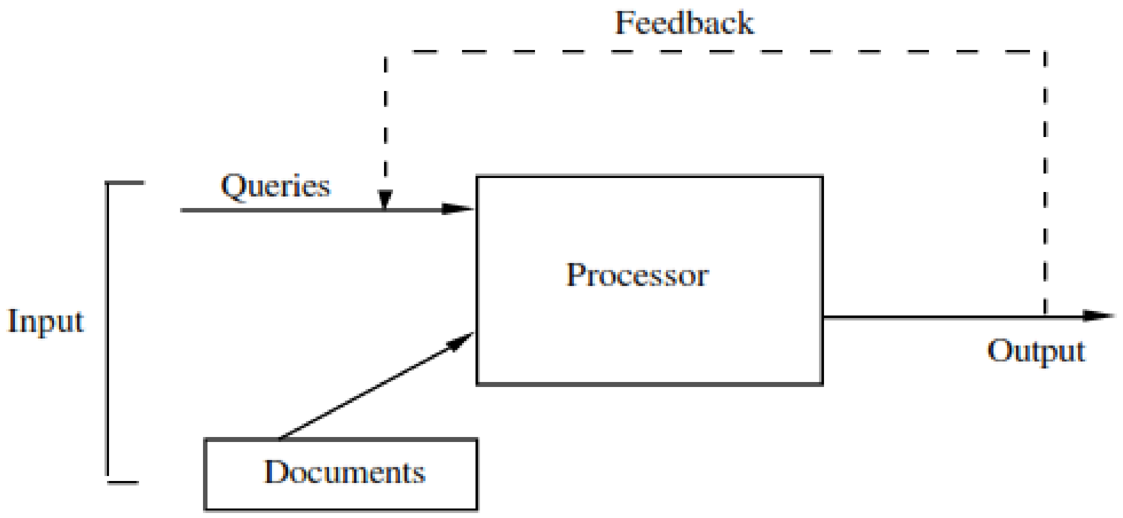


Figure 3: A Typical IR System.

First, the input, is to find a representation of all the documents, as well as the query.

Second, the processor, which are structured information in an appropriate way with how to perform the actual retrieval function.

Finally, we come to the output, which is usually a set of ordered documents.

2.2.4 *Information versus Data Retrieval*

The operating system represents data retrieval systems, for example Windows searches for files by specifying the name accurately. For information retrieval systems such as Bing or Google, it produces output in the document set to be identical to the query by processing natural text.

In information retrieval systems, the performance of the system is increased by feedback on the output displayed so that the documents related to the query are examined.

2.3 THE RETRIEVAL PROCESS

The retrieval process defines a database of texts by the database administrator. After finishing the definition of documents, we build an index of text that is unstructured as the critical data, the most popular is the inverted file as shown in Figure 4. The operation retrieval can be initiated. We analyze and convert the user's need by the processes applied to the text. Then, the query operations are applied before the actual query is created, which provides a system representation of the user's need. The query is then processed to obtain the recovered documents. Before sending documents to the user, they are ranked according to the probability of relevance. The documents are then examined in order to search for useful information.

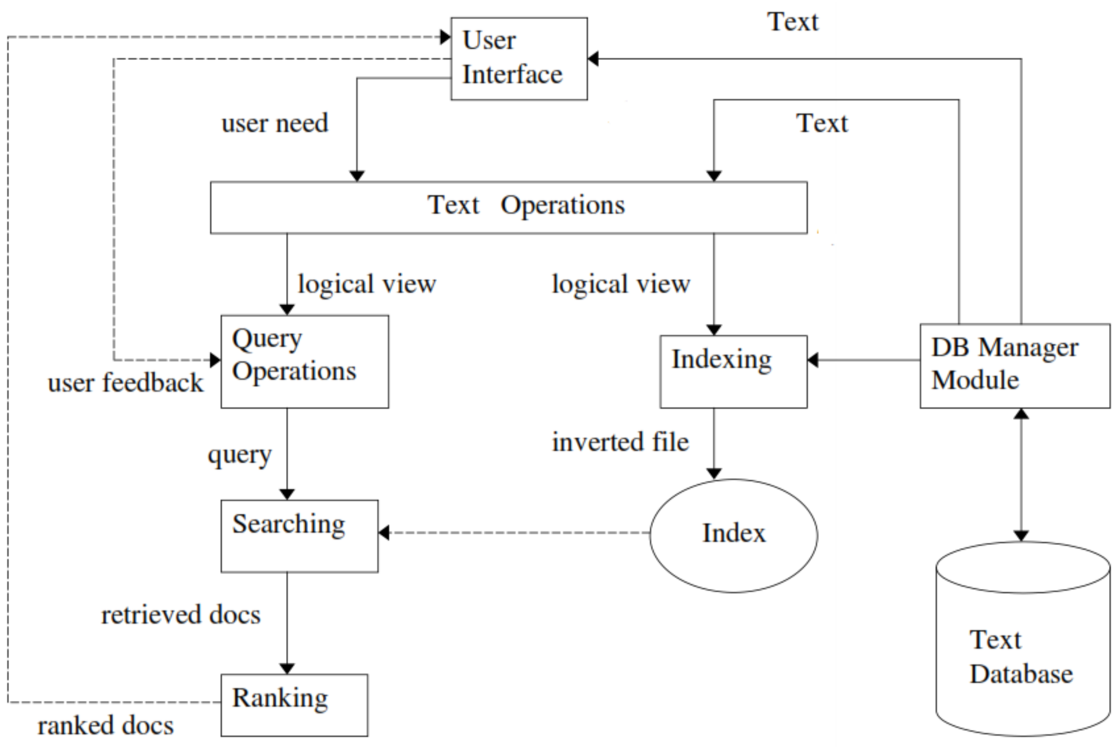


Figure 4: The process of retrieving information

2.3.1 *The term vocabulary and postings lists*

In this section, we mention how the basic unit of a document is defined with the sequence of characters that it consists of (Section 2.1.1). Then some substantive linguistic issues with the vocabulary of terms used by the System (Section 2.1.2).

2.3.1.1 *Document delineation and character sequence decoding*

Digital documents are a set of bytes found in a file or on a Web server. Documents are processed so that they are converted into a linear sequence of characters. For the English text in ASCII encoding, this is trivial. But often things become much more complicated. Character sequences may be encoded by one of different single byte or multi-byte encoding systems, such as Unicode UTF-8. We need to specify the correct encoding by selecting the user or by heuristic methods .We may save the encoding option to help us identify the language used to write.

You may have to decode characters from some binary representation, such as Microsoft Word DOC files and / or compressed format, such as zip files.

2.3.1.2 *Determining the vocabulary of terms*

In this section, define the terms used in the retrieval system, in addition to solve some linguistic issues problems, for example stemming and lemmatization, tokenization, stop word and normalization.

2.3.1.2.1 TOKENIZATION Tokenization is a task that cut text into pieces such as words, keywords, phrases, symbols and other elements that we call tokens. In the Tokenization process, some characters such as punctuation are ignored.

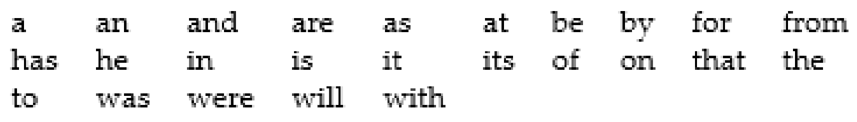
Here is an example of tokenization:

Input: Friends, Romans, Countrymen, lend me your ears;

Output: $\boxed{\text{Friends}}\ \boxed{\text{Romans}}\ \boxed{\text{Countrymen}}\ \boxed{\text{lend}}\ \boxed{\text{me}}\ \boxed{\text{your}}\ \boxed{\text{ears}}$

2.3.1.2.2 DROPPING COMMON TERMS: STOP WORDS Stop words are commonly used words (such as "The, are, than") are discarded from the search engine. When we create the index, certain words are removed from the index entry and are called a stop list.

Stop words are inappropriate search purposes because they appear frequently in the language, so they are dropped in order to save time and space.



a	an	and	are	as	at	be	by	for	from
has	he	in	is	it	its	of	on	that	the
to	was	were	will	with					

Figure 5: A stop list of 25 words

Some search engine allow the using of stop word, because when using term frequency and inverse document frequency we are allowed to lose or eliminate high weight or large frequency of stop word.

2.3.1.2.3 NORMALIZATION (EQUIVALENCE CLASSING OF TERMS)

Token normalization is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences (E.g. USA vs. U.S.A).The most standard way of normalizing is to create equivalence classes, which are named after one member of the set.

Might cause unexpected results (E.g. C.A.T. cat).

2.3.1.2.4 STEMMING AND LEMMATIZATION

The main objective of the stemming and lemmatization is to extract the root or base form and reduce wordâs form. Where the process of stemming is chopping at the ends of the words in order to achieve a base form. Lemmatization is a process in which the basic form (dictionary form) is achieved by removing word endings using vocabulary and morphological analysis. And known as the lemma.

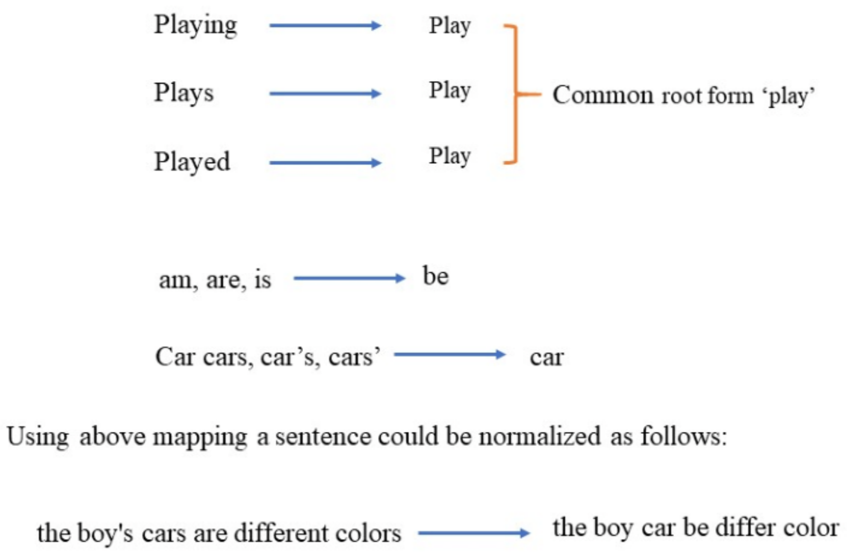

Playing
Play
Plays
Play
Played
Play
Common root form 'play'
am, are, is
be
Car cars, car's, cars'
car
Using above mapping a sentence could be normalized as follows:
the boy's cars are different colors
the boy car be differ color

Figure 6: Example of Stemming and lemmatization

2.3.2 *Dictionaries and tolerant retrieval*

In Section 2.3.2 we developed the ideas underlying inverted indexes for handling Boolean and proximity queries. Here, we develop techniques that are robust to typographical errors in the query, as well as alternative spellings. In Section 2.3.2.1 we develop data structures that help the search for terms in the vocabulary in an inverted index. In Section 2.3.2.2 we study the idea of a wildcard query.

We then turn to other forms of imprecisely posed queries, focusing on spelling errors in Section 2.3.2.3. Finally, in Section 2.3.2.4 we study a method for seeking vocabulary terms that are phonetically close to the query term(s).

2.3.2.1 *Search structures for dictionaries*

Given the query and inverted index, our task is to determine the existence of the query term in the vocabulary.

The vocabulary in the search process uses a classical data structure called the dictionary and has two solutions: tree search and hashing.

Hashing is used to search the dictionary by hashing each term into an integer over a space large enough to avoid hash collisions. Search trees solve many problems by allowing them to enumerate all the vocabulary terms that start with an automat. The binary tree is the most used. Searching for a term that starts from the root to each internal node and based on the result, the search progresses to one of the lower sub-trees.

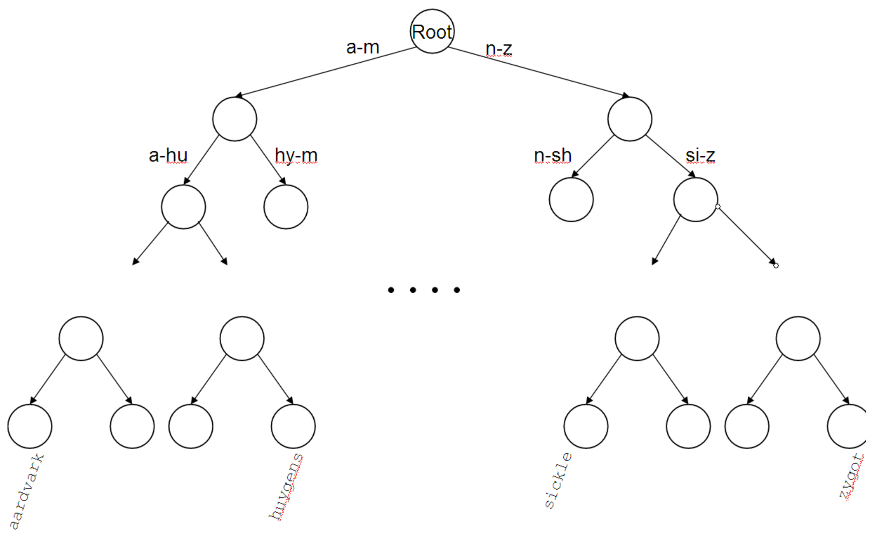


Figure 7: Example of Binary tree

2.3.2.2 *Wildcard queries*

Wildcard queries are used in the following cases: (1) Documents containing any variants (such as color vs. colour) are searched for by the user's knowledge of the various variables of the spelling terms. (2) Not sure the query term pronunciation for used (eg. awl vs. all, leading to the a*l query). (3)The user is not sure of the correct writing of the word or foreign words (for example, the query state * FlÃchenlÃnder. (4) The documents contain terms that can be retrieved using stemming ; but the retrieval is unsure (e.g., judicial vs. judiciary, leading to the wildcard query judicia*).

A query such as mon* is known as a trailing wildcard query, So that find all docs containing any word beginning with "mon" and in range: $nom \leqslant w < non$, and same case *mon find all words ending in "mon" but not Easy, we use the b-tree or binary tree for retrieve all words.

2.3.2.3 *Spelling correction*

We next look at the problem of correcting spelling errors in queries. For instance, we may wish to retrieve documents containing the term carrot when the user types the query carot. We look at three steps to solving this problem: the first based on edit distance and the second based on k-gram overlap, the third Related Context sensitive.

2.3.2.3.1 EDIT DISTANCE

When you see two strings S1 and S2, is the number of operations to convert one to the other provided they are at a minimum. Most commonly, the permitted editing operations for this purpose are: insert, delete, replace character with another character.

E.g., the edit distance from cat to dog is 3 and from cat to act is 2

2.3.2.3.2 K-GRAM INDEXES FOR SPELLING CORRECTION

A k-gram Is a sequence of k characters .Thus fam, amo and mou are all 3-grams occurring in the term famous. The $ character indicates the beginning and end of the term. For example, the term famous contains: $fa, fam, amo, mou, ous, us$, of 3-grams.

The dictionary contains all the k-grams of a term in the vocabulary and the postings list from k-gram refers to all the terms that contain k-grams, the k-grams index is used to retrieve the terms that match the k-grams query.

The 2-gram (bigram) index in Figure 8 shows (a portion of) the postings for the three bigrams in the query bord.

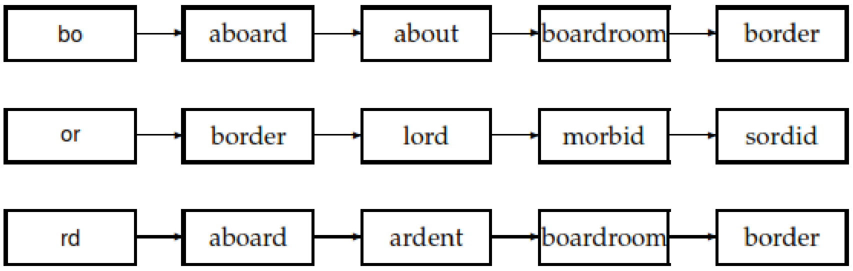


Figure 8: Matching at least two of the three 2-grams in the query bord.

2.3.2.3.3 CONTEXT SENSITIVE SPELLING CORRECTION

In literature, the problem of correction of context-sensitive spelling is included as a task of ambiguity so that the ambiguity of words is formulated through confusion in sets.

Spell errors are corrected by Context-sensitive spelling correction. For example, the word peace is a true word in isolation but it is wrong in this sentence in terms of context (peace from the cake) and it should be a piece instead of peace. The spelling correction systems are sure to check the words each time but do not correct these errors, although they represent 25% to 50% of the errors found (in English data) and must be processed.

2.3.2.4 *Phonetic correction*

A tolerant retrieval technique is associated with phonetic correction as the spelling errors that arise on the query sounds like the target term. The idea lies in the generation of each term, "phonetic hash" where the hash of terms is similar in sounds to the same value.

The idea of phonetic correction was used to identify the names of criminals in the International Police Departments in the early 20th century so that the names of criminals could be corrected.

2.4 MODELS OF INFORMATION RETRIEVAL

Retrieving the result on the query relates to the Retrieval models. The difference is not only in the expression or structure of the sentence for the query language but also in the form of the documents. An appropriate representation must therefore be found for effective retrieval.

There are several representations, in this section, we will address two models of information retrieval that provide exact matching and best matching.

2.4.1 *Exact match models*

Exact matching is documents are either retrieved or not, but the retrieved documents are not ranked.

2.4.1.1 *The Boolean Model*

In the Boolean retrieval model, a query is created in a logical structure of terms so that the operands are the index terms that are linked to the operators AND, OR, and NOT.

Through the set operators, the documents belonging to subsets that correspond to the terms of the query index are returned. For example: Find all documents include "Teacher And Student Or Not Employee".

2.4.2 *Best match models*

The best-match query is to find an answer for the query in the documents according to the relative similarity (in some cases dis-similarity).

2.4.2.1 *The Vector Space Model*

In the vector model, each document is a vector of index words with weights relevant to the importance of the term in the document. The index term weights are evaluated on the basis of a frequency of the index terms in the document, query, or collection. The documents are then returned by the system and are ranked according to the best similarity.

The vector space model contains four techniques used:

- Cosine similarity
- Inner Product
- Jaccard Similarity
- Dice Similarity

2.4.2.2 *Probabilistic Models*

The probabilistic model is searched by calculating the probability of relevant information from unrelated information. And to estimate how closely the document matches the query. The probability estimation is also defined as the probability of a term occurring in a collection of texts.

2.4.3 *Advantages and drawbacks of IR models*

The purpose of the Boolean retrieval model is to formulate complex expressions for users and it is difficult for the normal user to create Boolean queries so that all the terms are equally important and the set of retrieved documents is subject to the criterion of conformity. Therefore, the model used is more useful for retrieving data rather than information.

In order to reduce the size of the data, each word is represented in the dictionary without a suffix and is considered in the document and does not need to be added to it. However, the accuracy of the retrieval can be reduced with these improvements.

In the VSM, we can apply to each term a value from the corpus so that there is no difference between the long and short document, because of the use of Inverse document frequency, and we can calculate the importance of the term differently. But this model holds a flaw is that the long document can contain the terms specified in the query only in the title and abstract, although it is relevant to the query so that it is of low importance compared to a short document that contains in the footer the same terms.

There is also a drawback in VSM that there is no preference in the proximity of terms in other words the missing order so that the terms are separated in the document.

The Probabilistic retrieval Model are based on explicit assumptions but are not always commensurate with reality, so it requires that the terms be independent while ignoring frequency and weights.

2.5 IR APPLICATIONS

One of the models of NLP is the retrieval of information in the presence of many applications and techniques that solve the problems of life. There are some fields that use information retrieval:

1. Search engines: Most web systems that use the information retrieval system.
2. Digital libraries: Offices are also users of information retrieval system where they help to find information quickly from books.
3. It also uses information retrieval system in media such as searching for images, video, news and others.

2.6 EVALUATION IN INFORMATION RETRIEVAL

Measurement requires between the query and the set of documents to see how closely they are related. In the information system there are two techniques for the more common measures: recall and precision. When you search for the subject of what the database is divided to four sections are shown in Figure 9. However, there is the possibility of retrieving or not retrieving these items.

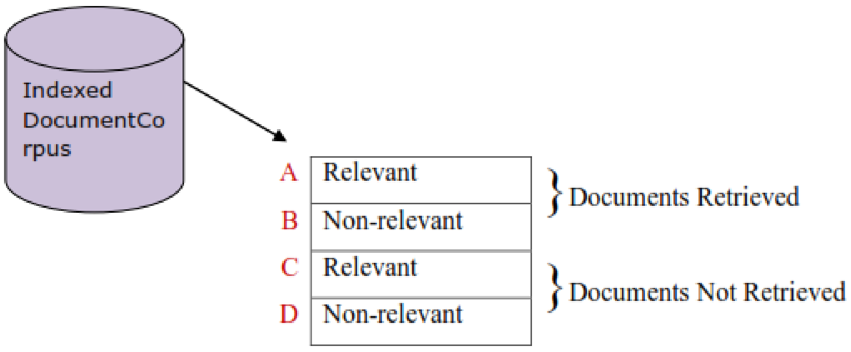


Figure 9: Effects of Search on Total Document Space.

2.6.1 *Precision*

Precision (b) is a relevant documents from the retrieved documents. It is called the positive predictive value in binary classification.

$$\text{Precision} = P = \frac{\text{Number\_Relevant\_Retrieved}}{\text{Number\_Total\_Retrieved}} = \frac{A}{A+B}$$

Where,

Number_Relevant_Retrieved: The number of documents retrieved and related to the query.

Number_Total_Retrieved: the total number of documents retrieved from the corpus.

2.6.2 *Recall*

Recall (R) is documents relevant to the query from the total documents in corpus that are relevant to the user's search.

$$\text{Recall} = R = \frac{\text{Number\_Relevant\_Retrieved}}{\text{Number\_Relevant}} = \frac{A}{A+C}$$

Where,

Number_Relevant_Retrieved: The number of documents retrieved and related to the query.

Number_Relevant: The number of documents in the corpus that are relevant to the query.

2.7 CONCLUSION

in this chapter, we studied the notion of information retrieval and different tools and techniques of IR models. To retrieve relevant documents according to user's need.

3 VECTOR SPACE MODEL

3.1 INTRODUCTION

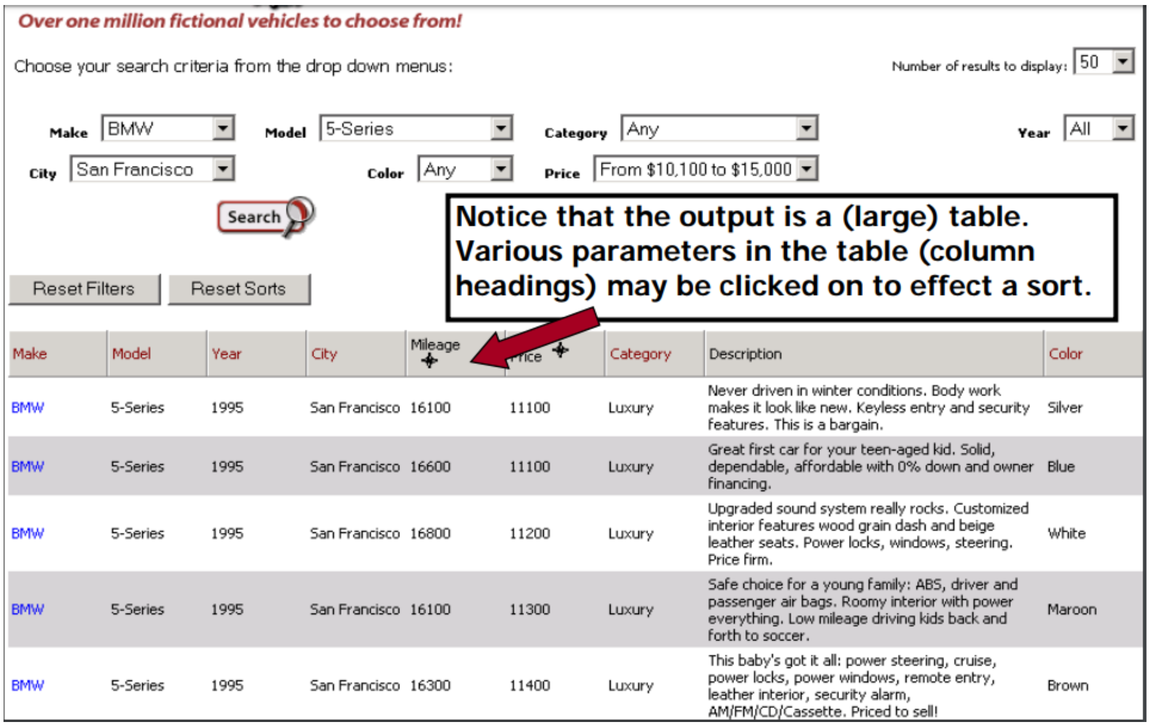
The Boolean retrieval can match or cannot match a user query. This makes it difficult to examine a large number of retrieved documents which match a user query. The documents produced should be ranked so that a certain degree is taken to match the user's input. In the other hand vector space retrieval can rank documents for a user query using many techniques. This Chapter is divided into four axes:

1. We work on the parametric and zone Index, which allows us to record documents in order according to the query and thus easy to retrieve.
2. In Section 2, we give each term a weight of importance in the document or in the corpus based on some statistics.
3. In the third section we calculate the degree between the document and the query through the weights and are called points of space vector.
4. In Section 4 we focus on the many forms of term-weighting found in the vector space model.

3.2 PARAMETRIC AND ZONE INDEXES

We always believe that documents are only a sequence of terms, but in fact certain structures may be related to each document and the machine can identify them as descriptive data and an example of the reality of the title and author of the document, the date of publication. These metadata may carry limited values such as the set of dates of authorship.

Fields may similar Zones but content is free to be arbitrary for the zone. For example, titles and abstracts are considered to be zones. They are indexed with a separate inverted index. See Figure 11 and Figure 12.



Over one million fictional vehicles to choose from!

Choose your search criteria from the drop down menus:

Number of results to display: 50

Make: BMW
Model: 5-Series
Category: Any
Year: All
City: San Francisco
Color: Any
Price: From $10,100 to $15,000

Notice that the output is a (large) table. Various parameters in the table (column headings) may be clicked on to effect a sort.

Make	Model	Year	City	Mileage	Price	Category	Description	Color
BMW	5-Series	1995	San Francisco	16100	11100	Luxury	Never driven in winter conditions. Body work makes it look like new. Keyless entry and security features. This is a bargain.	Silver
BMW	5-Series	1995	San Francisco	16600	11100	Luxury	Great first car for your teen-aged kid. Solid, dependable, affordable with 0% down and owner financing.	Blue
BMW	5-Series	1995	San Francisco	16800	11200	Luxury	Upgraded sound system really rocks. Customized interior features wood grain dash and beige leather seats. Power locks, windows, steering. Price firm.	White
BMW	5-Series	1995	San Francisco	16100	11300	Luxury	Safe choice for a young family: ABS, driver and passenger air bags. Roomy interior with power everything. Low mileage driving kids back and forth to soccer.	Maroon
BMW	5-Series	1995	San Francisco	16300	11400	Luxury	This baby's got it all: power steering, cruise, power locks, power windows, remote entry, leather interior, security alarm, AM/FM/CD/Cassette. Priced to sell!	Brown

Figure 10: Parametric search. In this example, we have a collection that contains fields that allow us to select products according to fields such as the type and price of the vehicle.

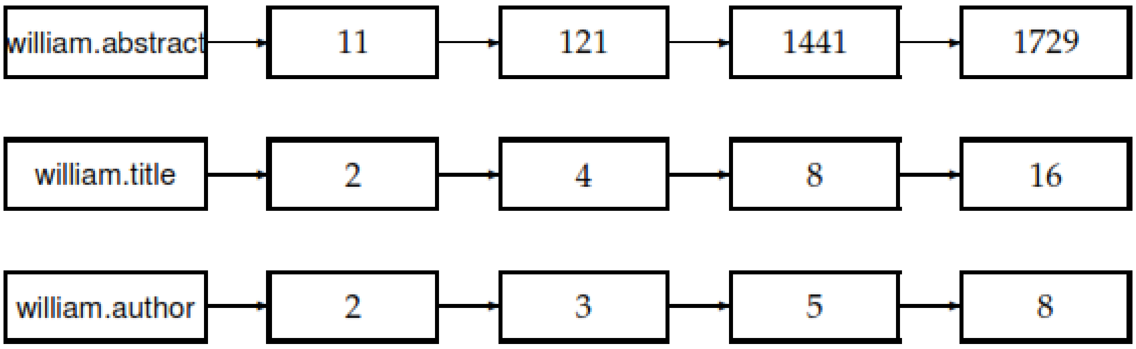


Figure 11: Basic zone index; zones are encoded as extensions of dictionary entries.

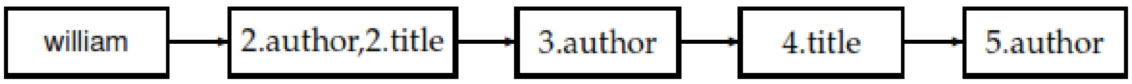
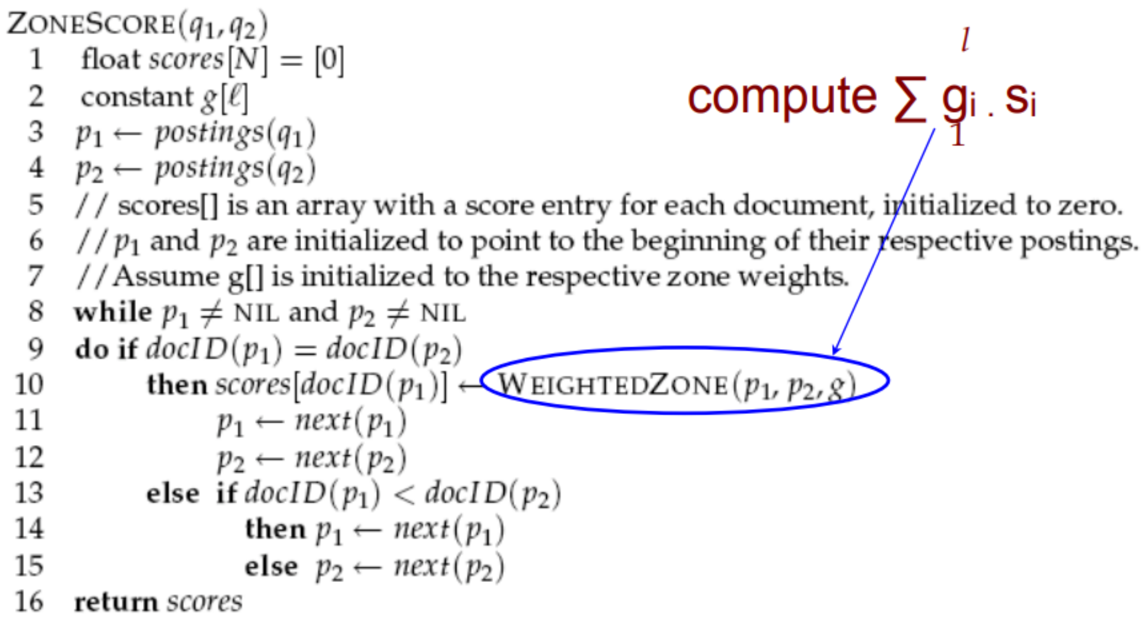


Figure 12: Zone index in which the zone is encoded in the postings rather than the dictionary.

3.2.1 *Weighted zone scoring*

Give a separate weight for each zones exists in a document, for example there are three zones (or fields): body, title, and author. But not all zone are equally important. e.g.author (Z_1), $\mathtt{title}(Z_2)$, $\mathtt{body}(Z_3) \rightarrow Z_1 = 0.3, Z_2 = 0.15, Z_3 = 0.55$

(so that they add up to 1) Score for a zone = 1 if the query term occurs in that zone, 0 otherwise (Boolean logic). For Example: Query term appears in author and body only. Document score: (0.3 * 1) + (0.55 * 1) = 0.85


compute $\sum_{1}^{l} g_i . s_i$

```
ZONESCORE($q_1, q_2$)
  float $scores[N] = [0]$
  constant $g[\ell]$
  $p_1 \leftarrow postings(q_1)$
  $p_2 \leftarrow postings(q_2)$
  // scores[] is an array with a score entry for each document, initialized to zero.
  //$p_1$ and $p_2$ are initialized to point to the beginning of their respective postings.
  //Assume g[] is initialized to the respective zone weights.
  while $p_1 \neq$ NIL and $p_2 \neq$ NIL
  do if $docID(p_1) = docID(p_2)$
        then $scores[docID(p_1)] \leftarrow$ WEIGHTEDZONE($p_1, p_2, g$)
             $p_1 \leftarrow next(p_1)$
             $p_2 \leftarrow next(p_2)$
        else if $docID(p_1) < docID(p_2)$
               then $p_1 \leftarrow next(p_1)$
               else $p_2 \leftarrow next(p_2)$
  return $scores$
```

Figure 13: Algorithm for computing the weighted zone score from two postings lists.

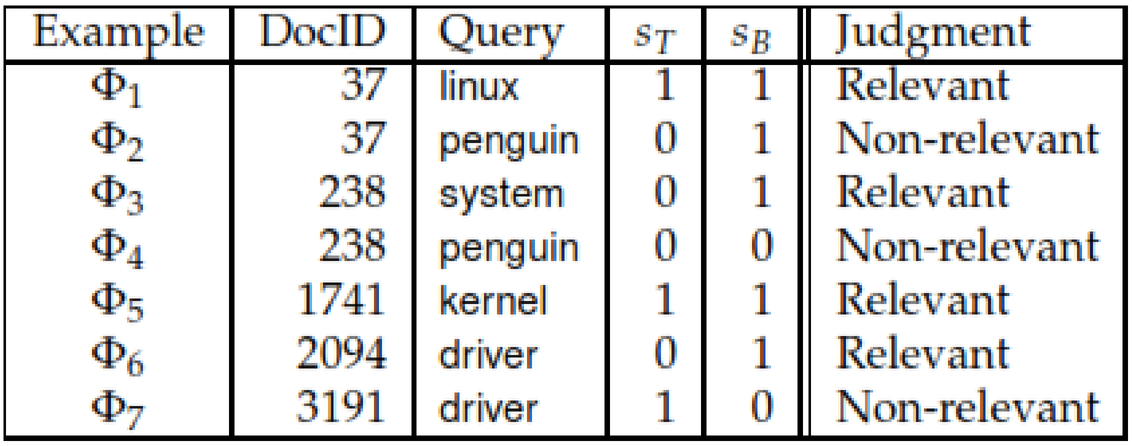
3.2.2 *Learning weights*

Each document contains body and title zone. Learning weights is the simple case of weighted zone scoring. We use the Boolean function to match. For calculate

$s_T(d, q)$ and $s_B(d, q)$. In order to determine the match of the body/title zone between the document and the query, we calculate:

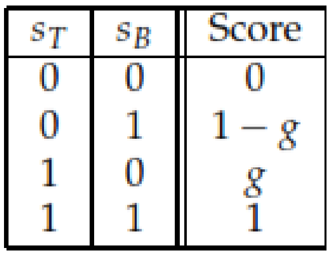
$$\text{score}(d, q) = g \cdot s_T(d, q) + (1 - g)\, s_B(d, q)$$

g is the parameter of zone weight, using a set of training examples .each of which is a triple of the form $\Phi_j = (d_j\ , q_j, r(d_j, q_j))$ where $r(d_j, q_j)$ is the editorial relevance judgment. See Figure 14 and Figure 15 to explain the simple machine learning



Example	DocID	Query	s_T	s_B	Judgment
Φ_1	37	linux	1	1	Relevant
Φ_2	37	penguin	0	1	Non-relevant
Φ_3	238	system	0	1	Relevant
Φ_4	238	penguin	0	0	Non-relevant
Φ_5	1741	kernel	1	1	Relevant
Φ_6	2094	driver	0	1	Relevant
Φ_7	3191	driver	1	0	Non-relevant

Figure 14: illustration of training examples.



s_T	s_B	Score
0	0	0
0	1	$1 - g$
1	0	g
1	1	1

Figure 15: four possible combinations of s_T and s_B

3.3 TERM FREQUENCY AND WEIGHTING

In information retrieval. Term frequency and inverse document frequency is a digital statistical tool that determines the importance of words in a collection or within a document. It is also considered an influential factor in the retrieval of information and text mining and the most widely used and popular schemes in the search systems and constitutes 83% of the digital libraries search system. To calculate the weight of each term we multiply the two compounds term frequency and Inverse document frequency:

$$\text{tf-idf}_{t,d,D} = \text{tf}_{t,d} \times \text{idf}_{t,D}$$

3.3.1 *Inverse document frequency*

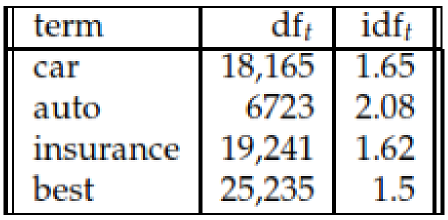
In information retrieval, The Inverse document frequency (IDF) is part of the Tf-Idf. It's a measure of similarity of terms. The inverse document frequency (idf) of the word t is defined as follows:

$$\text{idf}_t = \log(N/\text{df}_t)$$

The document frequency Df: Number of documents contain a term t .

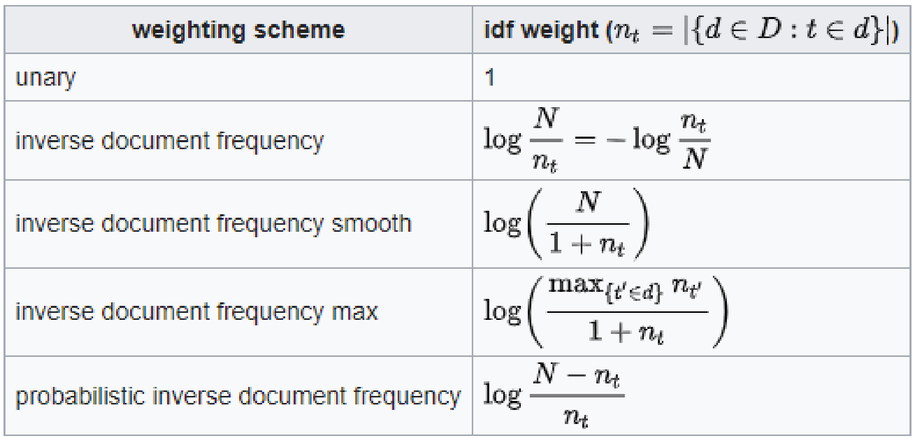
The number : the total number of documents in a collection.

In Figure 16, Example of idf's in the Reuters collection of 806,791 documents



term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

Figure 16: Example of Idf.



weighting scheme	idf weight ($n_t = \lvert\{d \in D : t \in d\}\rvert$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log \left(\frac{N}{1+n_t}\right)$
inverse document frequency max	$\log \left(\frac{\max_{\{t' \in d\}} n_{t'}}{1+n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N-n_t}{n_t}$

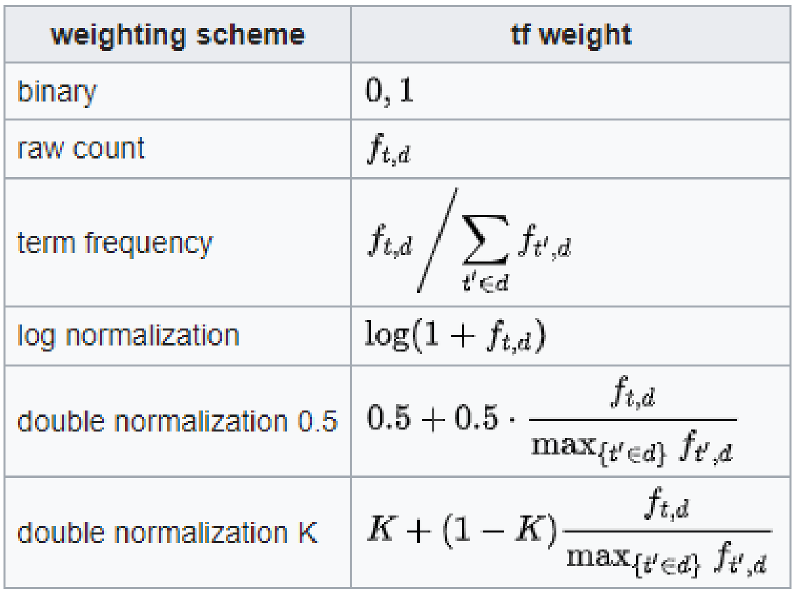
Figure 17: Variants of inverse document frequency (Idf) weight

3.3.2 *Term frequency weighting*

The term frequency weighting is a numeric value assigned to terms that give importance to terms in the document. Tf of the term t is defined as follows:

$$tf_{t,d} = \begin{cases} \log_{10}(1 + f_{t,d}), & \text{if } f_{t,d} > 0 \\ 1 & ,\text{otherwise} \end{cases}$$

The raw count by $f_{t,d}$: Number of word in document d



weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Figure 18: Variants of term frequency ( ) weight

3.4 THE VECTOR SPACE MODEL FOR SCORING

In this section we will talk about the document vector, which is a tool for determining the importance of term and is included in the process of retrieval and classification

3.4.1 *Dot products*

We have two vectors: $\vec{a}$=($a_1, a_1, a_3, ...$) and $\vec{b}$ =($b_1, b_1, b_3, ...$) where a_n and b_n are the components of the vector and in our study this is the TF-ID weight of each word in the document. n is the vector dimension. The product dot is defined as follows:

$$\mathrm{sim}(a, b) = \vec{v}(a) . \vec{v}(b) / \mid \vec{v}(a) \mid \mid \vec{v}(b) \mid = \cos(a,b)$$

The Cosine Similarity $(\cos\theta)$ will generate a metric that says how related or match are two documents by looking at the angle instead of magnitude, Such as in the example follow:

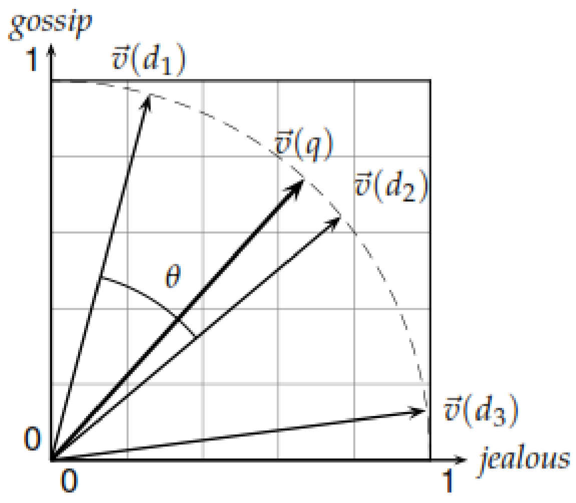
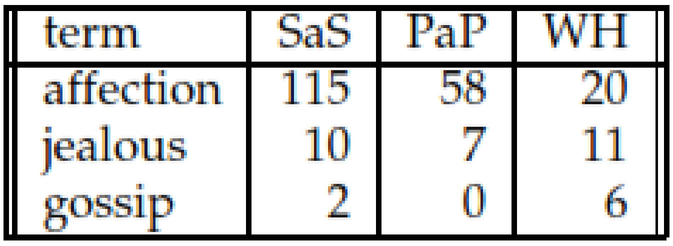


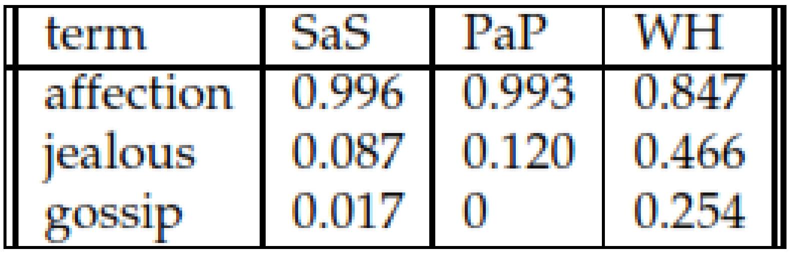
Figure 19: Cosine similarity illustrated. sim(d1 , d2) = cosθ.

In this example, there are three novels: Austenâs Sense and Sensibility, Pride and Prejudice and Emily Brontë's Wuthering Heights. We calculate the frequency and the normalized of three words affection, jealous and gossip. See Figure 20 and Figure 21:



term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

Figure 20: Term frequencies in three novels



term	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

Figure 21: Term vectors for the three novels of Figure 20.

3.4.2 *Queries as vectors*

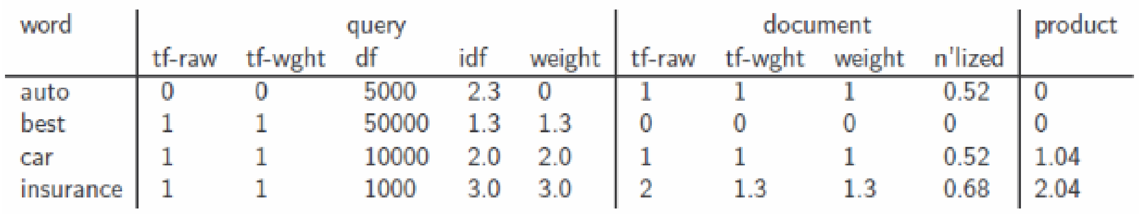
Queries are represented by vectors, such as each document. Then the score is determined equal to point product of each document:

$$\vec{v}(q).\vec{v}(d)$$

We can select the documents most closely related to the query through the result of score. As in the following:

$$\mathrm{score}(q,d) = \vec{v}(q).\vec{v}(d) / \mid \vec{v}(q) \mid \mid \vec{v}(d) \mid = \sum_{i=1}^{|v|}(qi,di) / (\sum_{i=1}^{|v|}(qi^2).\sum_{i=1}^{|v|}(di^2))$$

In this example(figure 22), We now consider the query best car insurance on a fictitious collection with N = 1,000,000 documents where the document frequencies of auto, best, car and insurance are respectively 5000, 50000, 10000 and 1000.



word	query					document				product
	tf-raw	tf-wght	df	idf	weight	tf-raw	tf-wght	weight	n'lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

Figure 22: Example calculate the score between Query and document

Key to columns:

- tf-raw: Raw term frequency.
- tf-wght: Logarithmically weighted.
- df: Document frequency.

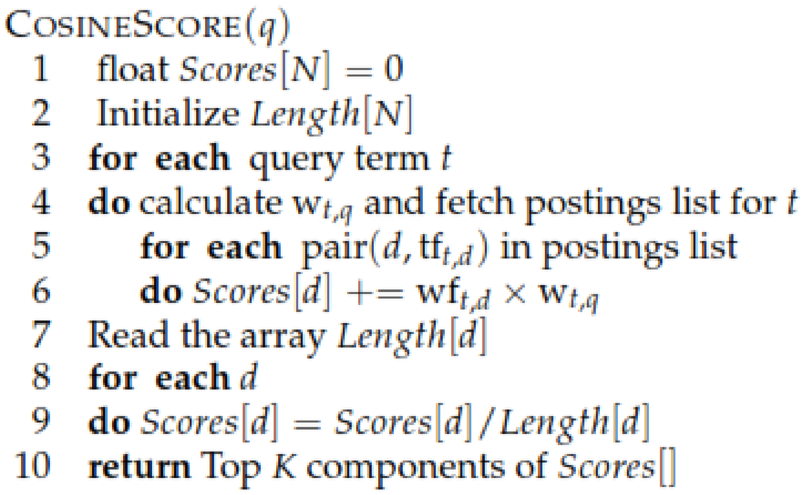
- idf: inverse document frequency.
- weight: The final weight of the term in the query or document.
- n′lized: Document weights after cosine normalization.
- product: The product of final query weight and final document weight.

The final similarity score between query and document:

$$\sum_j (w_{qi}.w_{di}) = 0 + 0 + 1.04 + 2.04 = 3.08$$

3.4.3 *Computing vector scores*

We have both the documents collection and the query is a vector. We want to select documents that take the highest score of correlation with the query and then are arranged according to the best. In Figure 23 we provide an algorithm for calculating the highest scores K documents in N collection.



```
COSINESCORE($q$)
  float $Scores[N] = 0$
  Initialize $Length[N]$
  for each query term $t$
  do calculate $w_{t,q}$ and fetch postings list for $t$
      for each pair($d, tf_{t,d}$) in postings list
      do $Scores[d] += wf_{t,d} \times w_{t,q}$
  Read the array $Length[d]$
  for each $d$
  do $Scores[d] = Scores[d] / Length[d]$
  return Top $K$ components of $Scores[]$
```

Figure 23: The basic algorithm for computing vector space scores.

3.5 VARIANT TF-IDF FUNCTIONS

TF-IDF weighting has many variants. We discuss some of the principal ones in this Section and summarize these alternatives in Section 3.5.3.

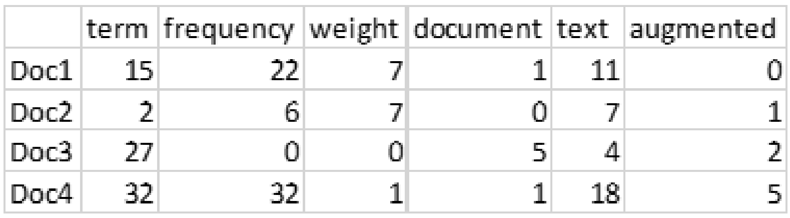
3.5.1 *Sub-linear tf scaling*

For example, repeating 50 times a term in a document compared to repeating it once is important for retrieving documents. There have been several researches in this variable. And the variable used frequently is the logarithm of the weighted term frequency so that it is considered a normalized term frequency on the collection, which assigns a weight given by

$$wf_{t,d} = \begin{cases} 1 + \log tf_{t,d}, & \text{if } tf_{t,d} > 0 \\ 0, & otherwise \end{cases} \tag{1}$$

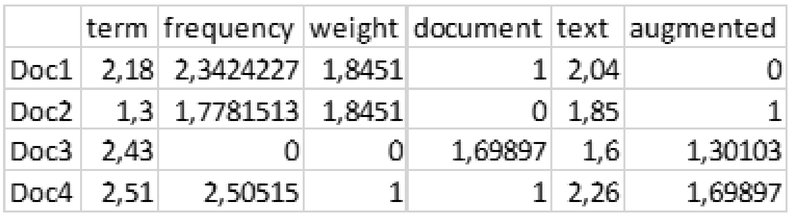
In this form, we may replace by some other function as in (1), to obtain:

$$\text{tf-idf}_{t,d,D} = tf_{t,d} \times idf_{t,D}$$



	term	frequency	weight	document	text	augmented
Doc1	15	22	7	1	11	0
Doc2	2	6	7	0	7	1
Doc3	27	0	0	5	4	2
Doc4	32	32	1	1	18	5

Figure 24: Example Sub-linear tf.



	term	frequency	weight	document	text	augmented
Doc1	2,18	2,3424227	1,8451	1	2,04	0
Doc2	1,3	1,7781513	1,8451	0	1,85	1
Doc3	2,43	0	0	1,69897	1,6	1,30103
Doc4	2,51	2,50515	1	1	2,26	1,69897

Figure 25: Example of Sub-linear tf after normalize the tf (Figure 24).

3.5.2 *Maximum tf normalization*

Normalization of weights is one of the well studied methods that are attributed to the terms and may take the maximum value of them $tf_{max}(d) = max_{\tau\epsilon d} tf_{t,d}$, We can calculate the normalized term frequency of each term in the document by

$$\mathrm{ntf}_{t,d} = a + (1-a)\mathrm{tf}_{t,d}/\mathrm{tf}_{max}(d)'$$

The a=0.5 is one of the most used so that the normalization of weights writes in the form

$$\mathrm{ntf}_{t,d]} = 0.5 + 0.5\mathrm{tf}_{t,d}/\mathrm{tf}_{max}(d)'$$

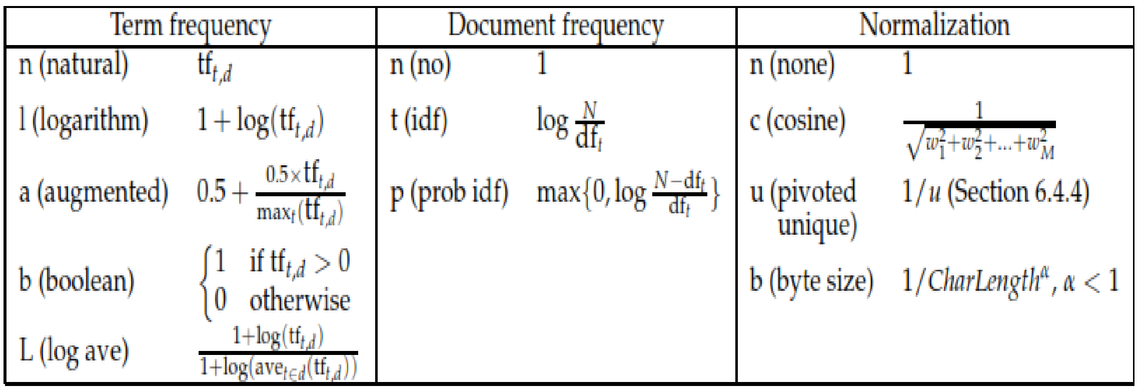
There are some problems in normalization weights we remember them:

1. Difficulty adjusting due to the stability of style.

2. The large frequency of a term in a document is contrary to the content of the document

3.5.3 *Document and query weighting schemes*

The method of scoring vector space is the selection of vector weights $\vec{v}(d)$ and $\vec{v}(q)$, The equation (d) is the basis of the retrieval system, which is used in any form of scoring in the vector space. There are many weighting schemes that different in the form of weights shown in Figure 26 this system is called SMART notation, It is represented the ddd.qqq form so that the three ddd characters represent the weight of the document, and the other three characters qqq are for the query, while the second triplet gives the weighting in the query vector. The first letter in each triplet specifies the term frequency component of the weighting (tf-weight), the second the document frequency component (Idf), and the third the form of normalization used ((tf-idf). For example, a very standard weighting scheme is lnc.ltc.



Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log\frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Figure 26: SMART notation for tf-idf variants.

3.5.4 *Pivoted normalized document length*

The reduction of the normalization factor for the document is an increase in the enhancement of retrieval of documents so that there is an inverse relationship. The Figure 27 shows the Pivote normalization using the cosine or the byte size. The set of documents is retrieved by pivote and slope. so that the pivote is the point of intersection of the two curves. The slope is the tangent of the angle made by old normalization with the new normalization as shown in figure 27.

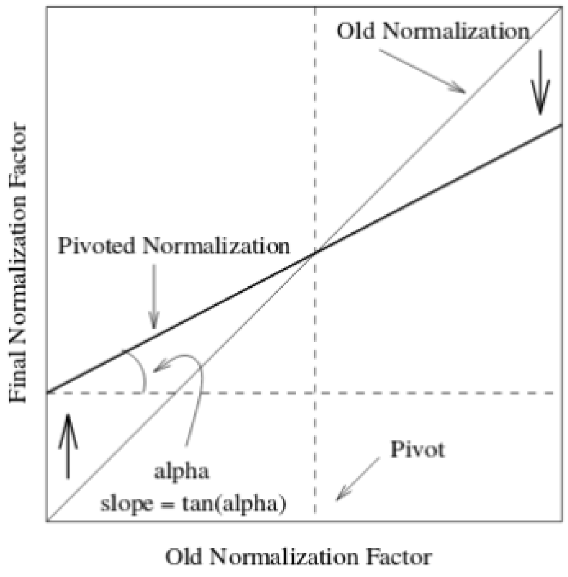


Figure 27: Pivoted normalization.

The pivoted normalization factor is represented by the equation for a line of gradient

$$\text{Pivoted\_Normalization} = (1 - \text{Slope}) \times \text{Pivot} + \text{Slope} \times \text{Old\_Normalization}$$

3.6 CONCLUSION

A vector space model is one of the most important models used in information retrieval systems and it carries a set of features such as the order of results as close to similarity and the representation of both the document and the query in a vector so as to contribute to the measurement of similarity.

Part III

APPLICATION PART

This part contains one chapter which are the design and development of our system to retrieval information

4 DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

In this Chapter we will present the conceptions of our retrieval system and the general structure of the environment in which we have developed the website. In the end we will explain the website some basic snapshots of the system.

4.2 OBJECTIVE

We aim to develop an application that have a set of basic tools:

- Upload English documents to the repository of the web site.
- Search for relative documents.
- Allow user to choose the SMART notation.
- Browse the text collection.

4.3 GENERAL STRUCTURE OF THE ENVIRONMENT

We will talk in this section about the environment in which we develop our program with the mention of programming language used in development and some of the libraries and packages imported.

4.3.1 *Programming language*

4.3.1.1 *ASP .NET*

ASP.NET is a framework for generating on-demand web pages, launched by Microsoft in July 2000, and used to implement web applications. This is a major evolution of Active Server Pages (ASP, aka Classic ASP), whereby this technique was incorporated into the Microsoft .NET platform. The ASP.NET engine is a filter plugged into the Internet Information Services (IIS) web server. It is distributed with the .NET framework. ASP.NET can be used with any programming language for the .NET platform (Visual Basic .NET, C , JScript ...).

4.3.1.2 *C sharp (C#)*

C (C sharp in British English) is an object-oriented programming language, marketed by Microsoft since 2002 and intended to develop on the Microsoft .NET platform. It is derived from C ++ and very close to Java which it takes the general syntax and concepts, adding concepts such as operator overload, indexers and delegates. It is used in particular to develop web applications on the ASP.NET platform.

4.3.1.3 *Python*

Python is an interpreted programming language, multi-paradigm and multiplatform. It promotes structured, functional and object-oriented imperative programming. It has strong dynamic typing, automatic garbage collection management, and an exception management system.

4.3.2 *Development environment*

We will talk about Microsoft development environment and data storage using Microsoft SQL Server.

4.3.2.1 *Microsoft Visual Studio*

Microsoft Visual Studio is a suite of development software for Windows and mac OS designed by Microsoft. The latest version is called Visual Studio 2019. Visual Studio is a complete set of development tools for generating ASP.NET web applications, XML web services, desktop applications, and mobile applications. Visual Basic, Visual C ++, and Visual C all use the same integrated development environment (IDE), which allows them to share tools and makes it easy to create solutions that use multiple languages.

In addition, these languages help to leverage the functionality of the .NET framework, which provides access to key technologies that simplify the development of ASP web applications and XML web services with Visual Web Developer.

4.3.2.2 *Microsoft SQL Server*

Microsoft SQL Server is a relational database management system (RDBMS) that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. Microsoft SQL Server is one of the three market-leading database technologies, along with Oracle Database and IBM's DB2.

4.3.3 *Framework and Library*

4.3.3.1 *Natural Language Toolkit (NLTK)*

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

4.3.3.2 *IronPython*

IronPython is an open-source implementation of the Python programming language which is tightly integrated with the .NET Framework. IronPython can use the .NET Framework and Python libraries, and other .NET languages can use Python code just as easily.

4.3.3.3 *Entity Framework*

Entity framework is the framework Object-Relational mapping (ORM) that Microsoft makes available as part of the .NET development. Its purpose is to abstract the ties to a relational database, in such a way that the developer can relate to the database entity as to a set of objects and then to classes in addition to their properties.

4.4 DESIGN

In this section we will present the conceptual aspect of our work using the Unified Modeling Language (UML) modeling language.

4.4.1 *Uses Case Diagram :System Inforamtion retreival*

Uses Case Diagram is precise and specify the main functional objectives of the application as well as the relation between these objectives and the users. Largely, the whole design is based on this diagram.

In this diagram(figure 28), we will show two actor User and Administrator dealing with the Information retrieval system, for example Login, Registration, Search Information, management system.

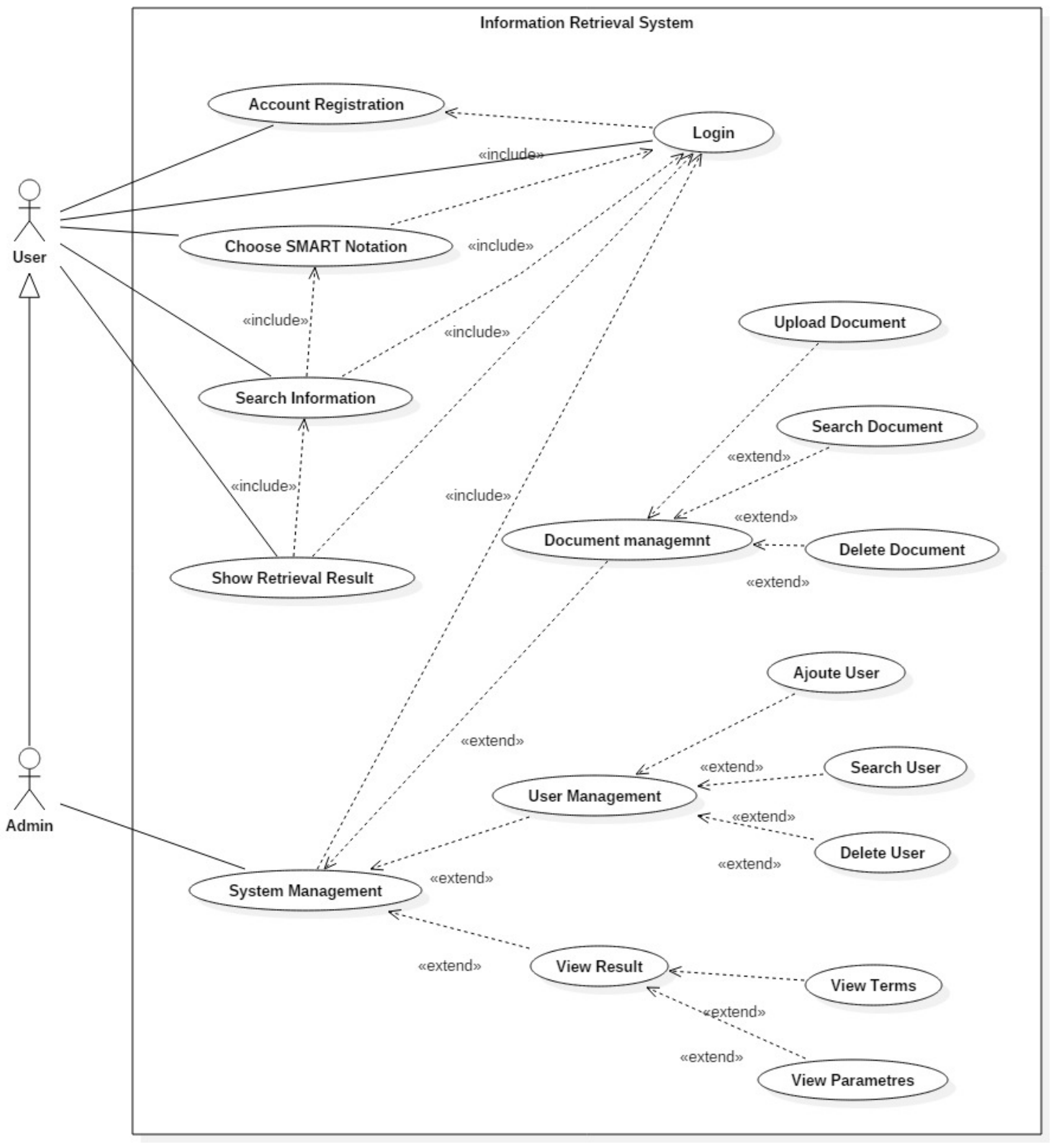


Figure 28: Uses Case Diagram of system information retrieval

4.4.2 *Activity Diagram*

One of the important UML Diagrams is the activity diagram, which is defined as a flowchart that designs flow from one activity to another, describing the dynamic aspects of the system.

In this part, we will show some activity diagrams:

- Login and Registration Activity.

- Search information in document activity.
- Upload document activity.

4.4.2.1 *Login and Registration Activity*

Activity diagram for Login and registration user and contains:

- Verification of information entered.
- Registration new user.

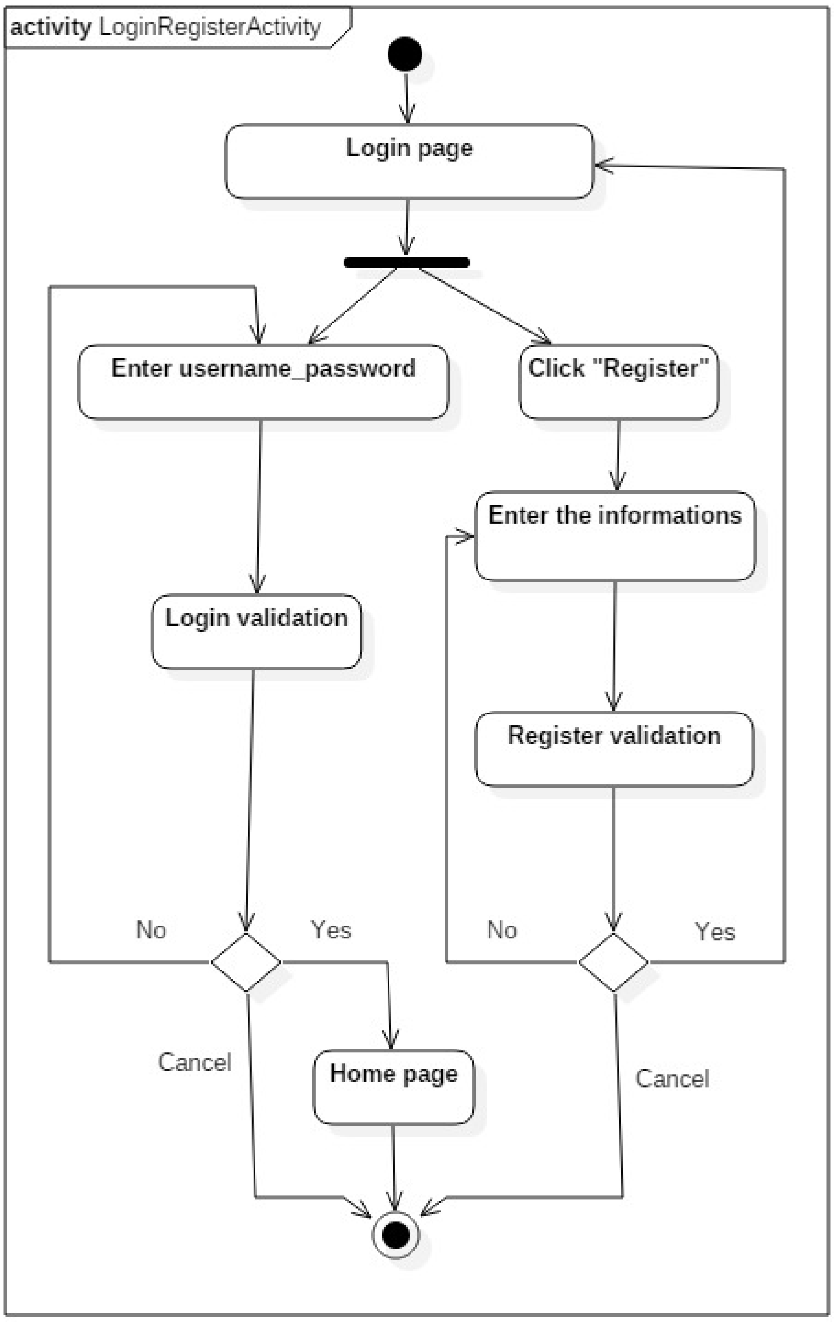


Figure 29: Diagram Login and Registration Activity

4.4.2.2 *Search Activity*

Activity diagram for a task search for input information and contains:

- Verification of the document entered.

- Word list process.
- Calculation parameters.
- calculation the similarity Document_Query and show the result.

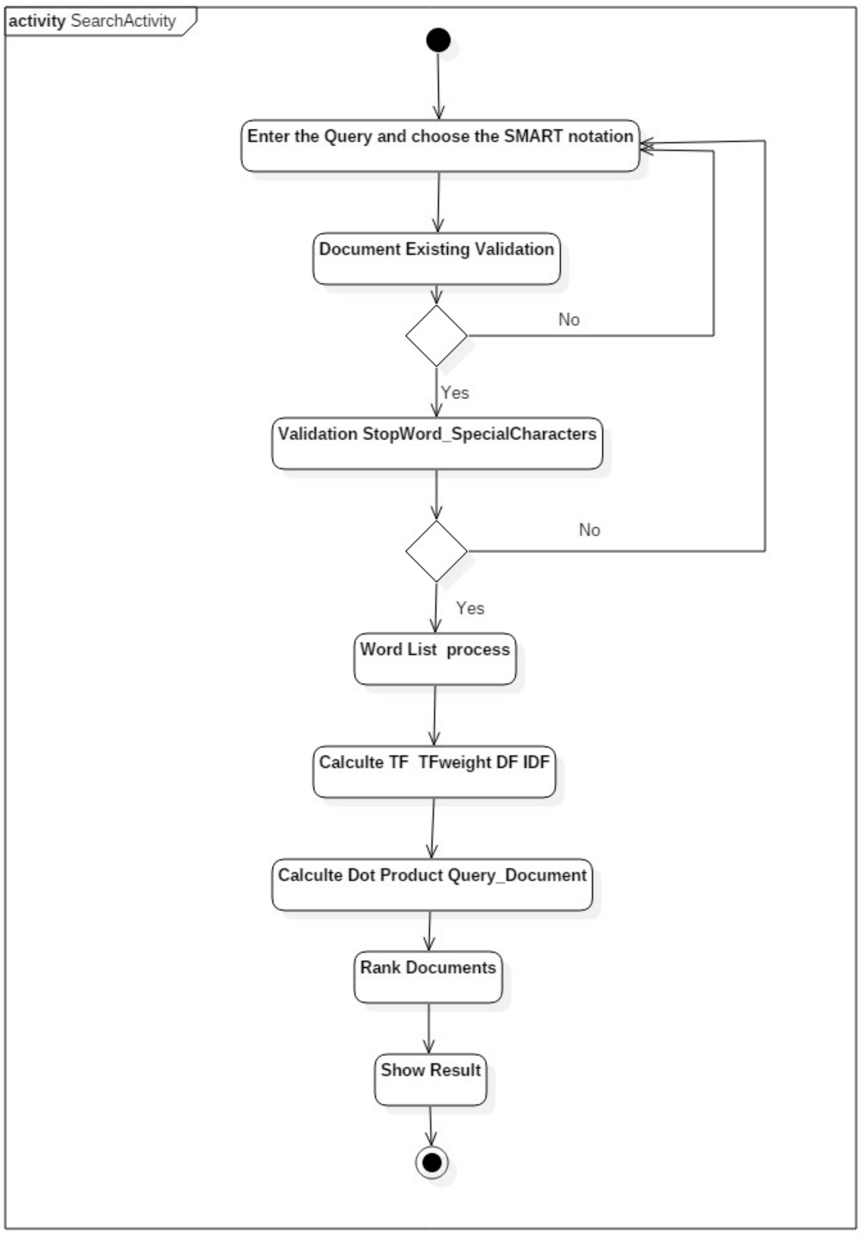
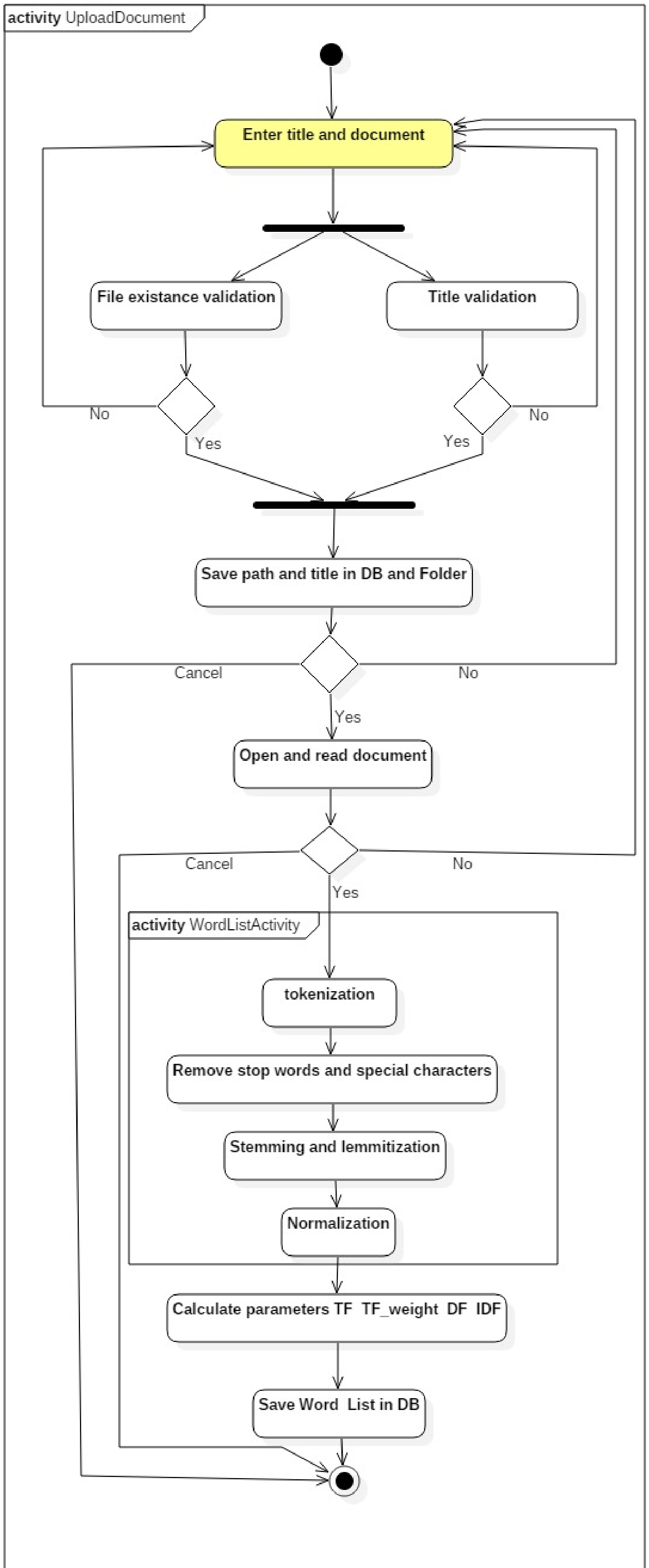

activity SearchActivity
Enter the Query and choose the SMART notation
Document Existing Validation
No
Yes
Validation StopWord_SpecialCharacters
No
Yes
Word List process
Calculte TF TFweight DF IDF
Calculte Dot Product Query_Document
Rank Documents
Show Result

Figure 30: Diagram Search Activity

4.4.2.3 *Upload Activity*

The following activity diagram is show the process of Upload document and contains:

- Validation of input title and document.
- Process of save and read document.
- The word list preparation process.
- calculation Parameter and save word list in DB.


activity UploadDocument
Enter title and document
File existance validation
Title validation
No
Yes
Yes
No
Save path and title in DB and Folder
Cancel
No
Yes
Open and read document
Cancel
No
Yes
activity WordListActivity
tokenization
Remove stop words and special characters
Stemming and lemmitization
Normalization
Calculate parameters TF TF_weight DF IDF
Save Word List in DB

4.4.3 *Sequence Diagram*

The UML sequence diagram shows how the operations are performed, shows the order of interactions and focuses on time.

This section shows us the most important Sequence Diagrams in our system such as searching for information or uploading a document with the registration process and login.

4.4.3.1 *Login Diagram*

In the sequence diagram of user logins, we show the sequence of events. The login begins by entering the user's information, after which the information is verified and in the end it is determined whether he is allowed to enter or not.

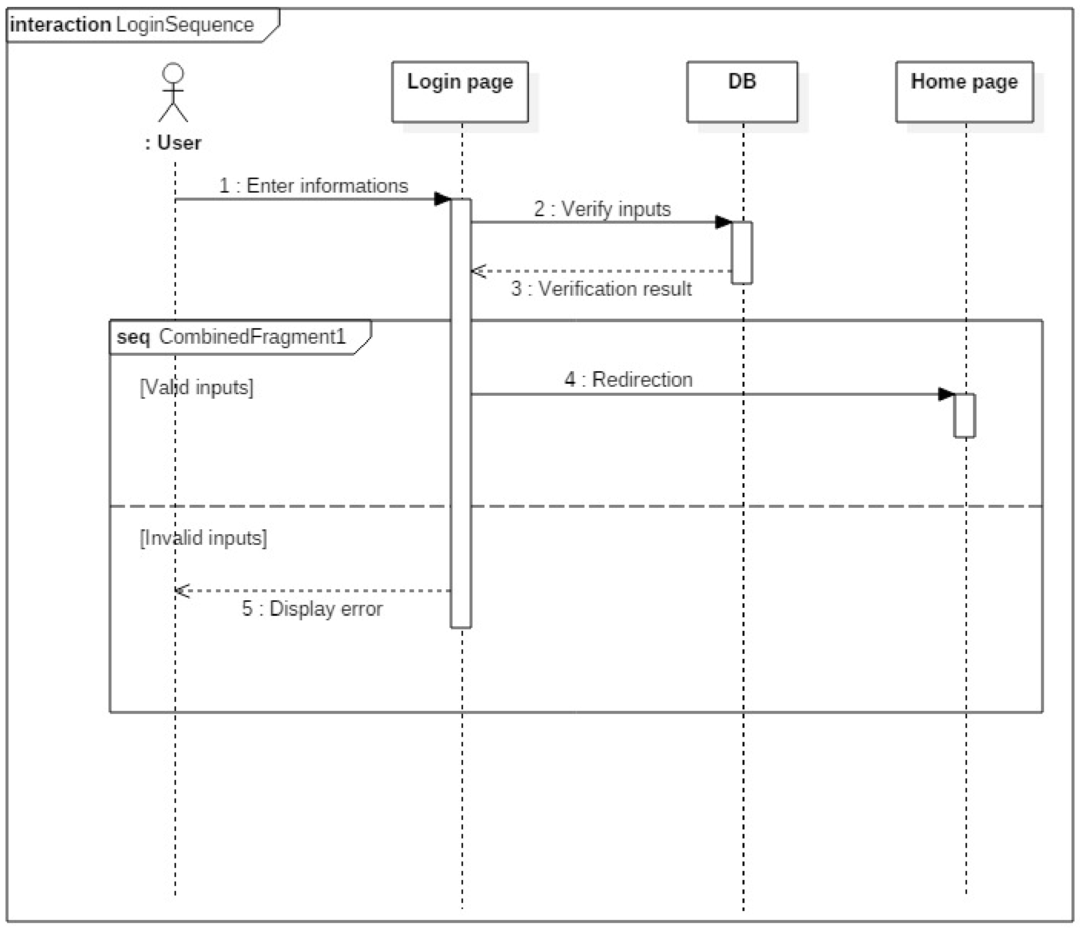


Figure 32: Diagram Sequence of Login

4.4.3.2 *Registration Diagram*

In the sequence diagram of the user registration, we show the sequence of events starting with the input of the new user information, after which the information is verified and in the end it is determined whether it will be stored in the database or not.

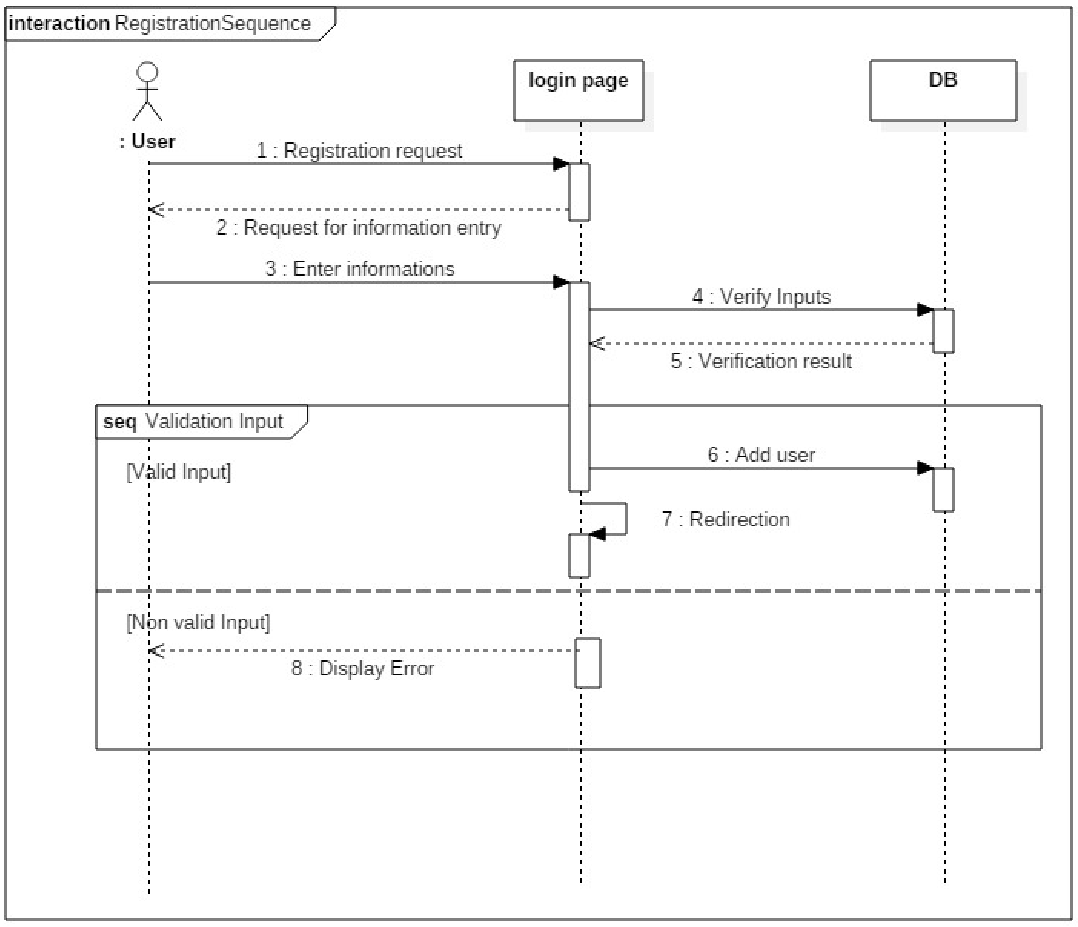


Figure 33: Diagram Sequence of Registration

4.4.3.3 *Search Diagram*

In the sequence diagram for searching query information, we check the nature of the words entered by the user, then we prepare the list of words for the query and calculate the weight TfâIdf for each word of the query and in the end we measure the similarity and show the result.

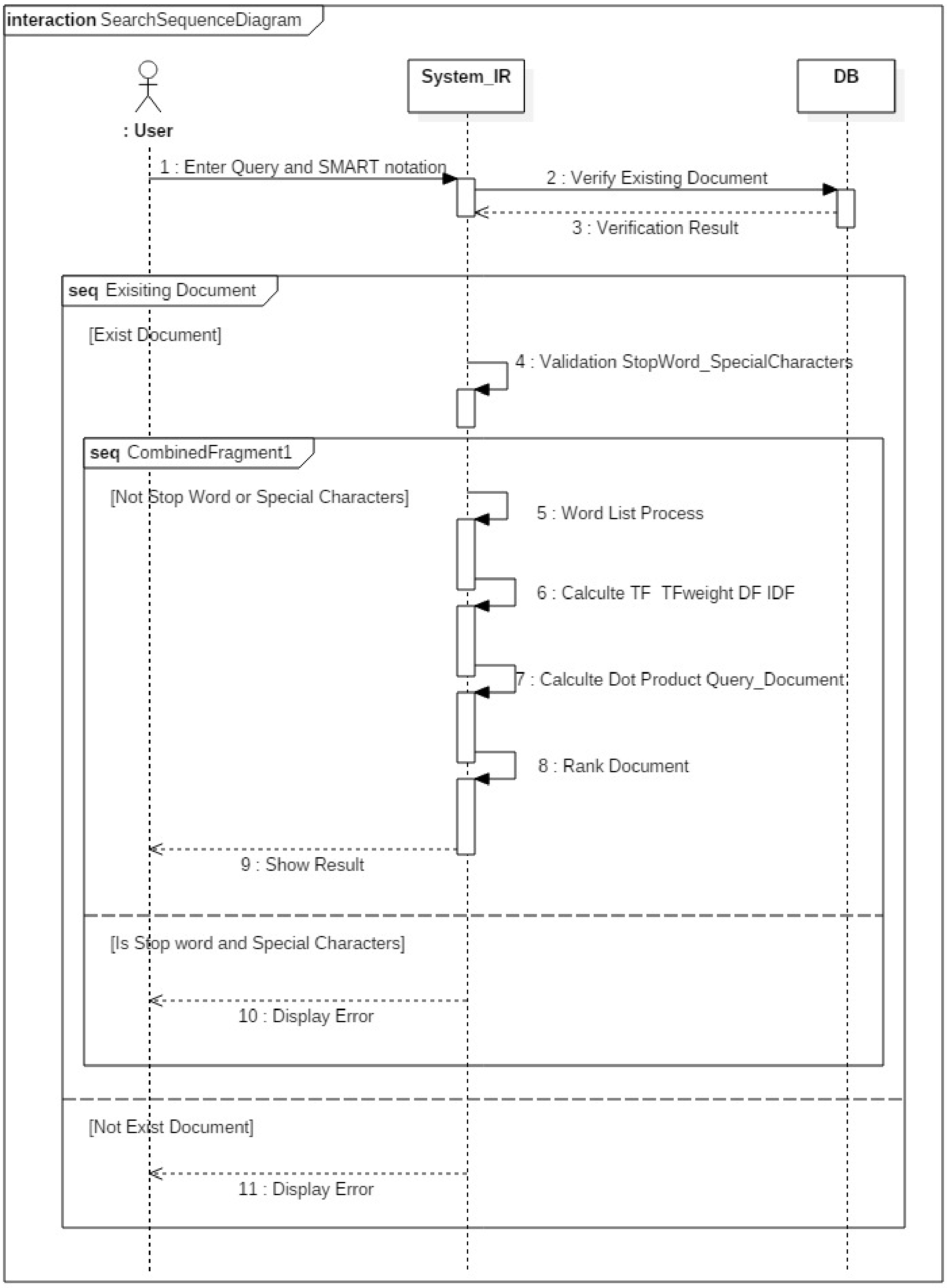

interaction SearchSequenceDiagram
: User
System_IR
DB
1 : Enter Query and SMART notation
2 : Verify Existing Document
3 : Verification Result
seq Exisiting Document
[Exist Document]
4 : Validation StopWord_SpecialCharacters
seq CombinedFragment1
[Not Stop Word or Special Characters]
5 : Word List Process
6 : Calculte TF TFweight DF IDF
7 : Calculte Dot Product Query_Document
8 : Rank Document
9 : Show Result
[Is Stop word and Special Characters]
10 : Display Error
[Not Exist Document]
11 : Display Error

Figure 34: Diagram Sequence of Search Information

4.4.3.4 *Upload Diagram*

In this diagram, we upload the documents in the server database so that they can be checked before loading the existence of the document or not. Then the procedures for saving, opening and reading the document. The words inside the documents are processed to create the word list, which in turn calculates the weight TfâIdf of each word, and in the end the list of the words is stored in the database.

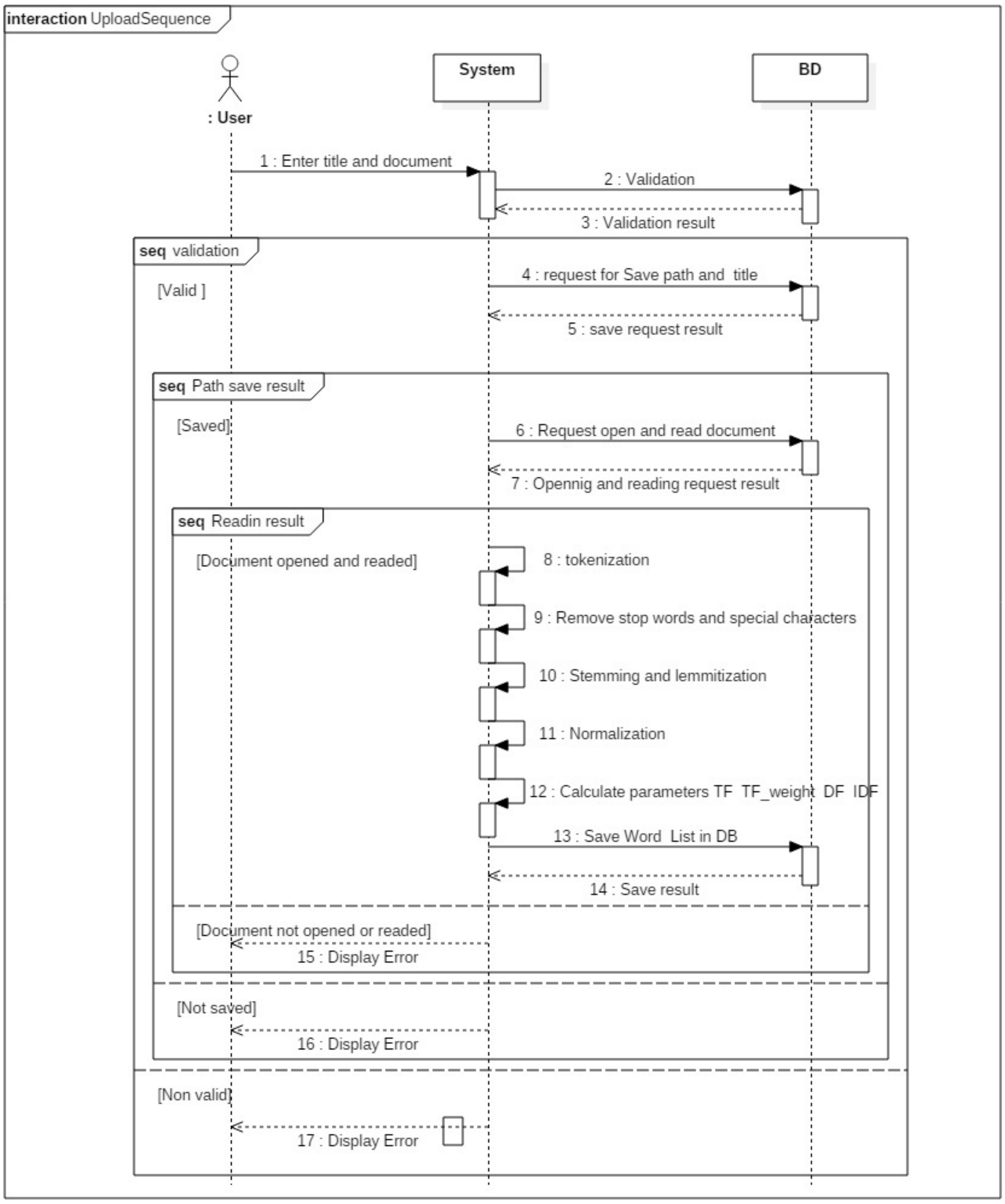

interaction UploadSequence
: User
System
BD
1 : Enter title and document
2 : Validation
3 : Validation result
seq validation
[Valid]
4 : request for Save path and title
5 : save request result
seq Path save result
[Saved]
6 : Request open and read document
7 : Opennig and reading request result
seq Readin result
[Document opened and readed]
8 : tokenization
9 : Remove stop words and special characters
10 : Stemming and lemmitization
11 : Normalization
12 : Calculate parameters TF TF_weight DF IDF
13 : Save Word List in DB
14 : Save result
[Document not opened or readed]
15 : Display Error
[Not saved]
16 : Display Error
[Non valid]
17 : Display Error

Figure 35: Diagram Sequence of Upload Document

4.4.4 *Class Diagram*

A class diagram is an illustration of the relationships and dependencies of source code between classes in UML. The class defines the methods and variables in an object, which is a specific entity in a program or unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP).

We have used a certain number of objects in our system such as document class, user class, and word class ... etc.

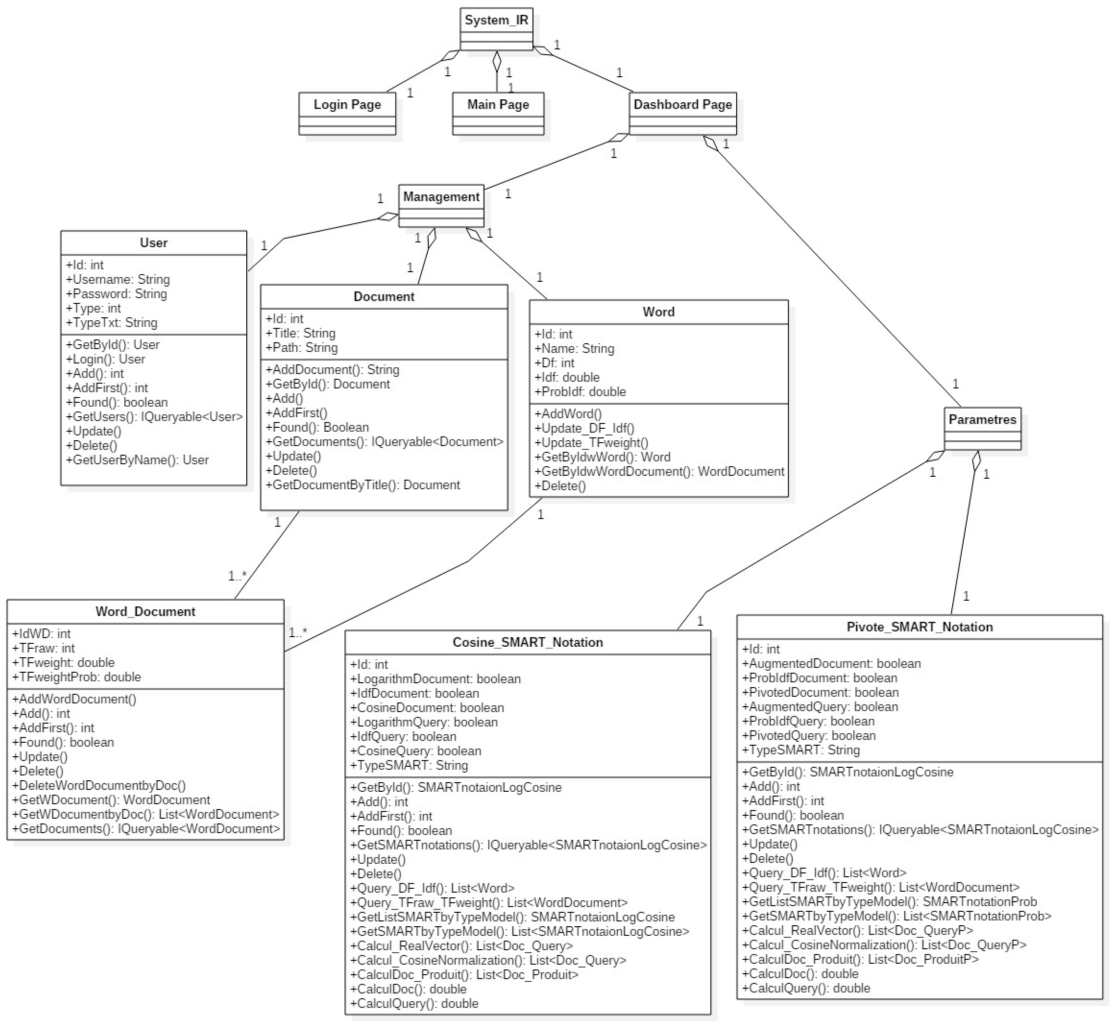


Figure 36: Diagram Class of System Information Retrieval

4.5 IMPLEMENTATION

We will show in this section snapshots of the main parts of our website. And it consists of three sections:

- The login page contains registration and login.

- The home page contains the search, about website, and the contact.
- The dashboard page that contains system maintenance (user, documents, parameters search).

Figure 37 shows the home page of our web site:

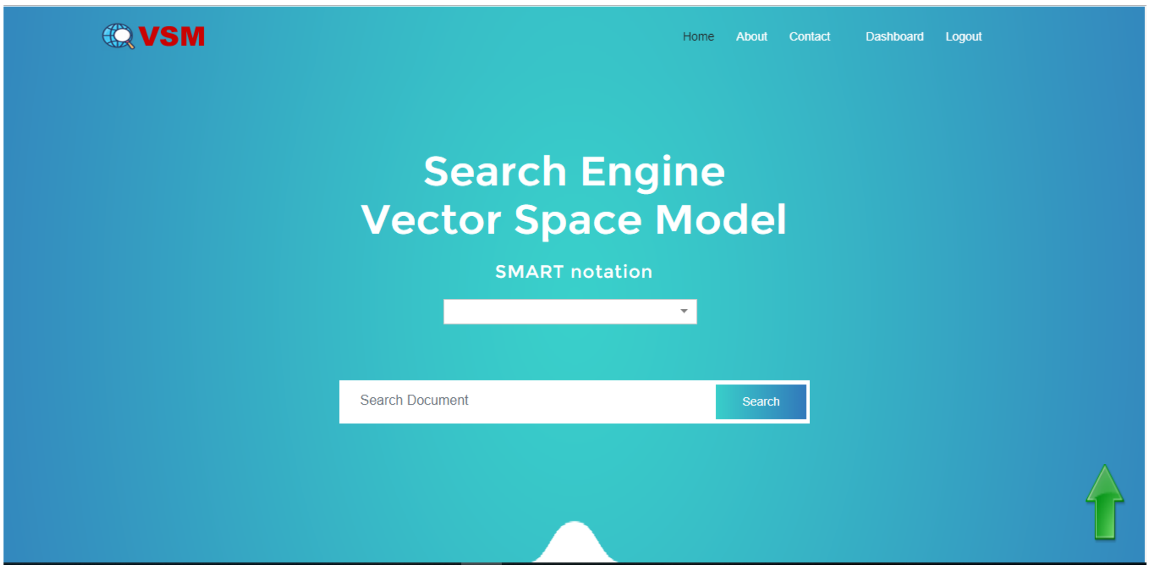
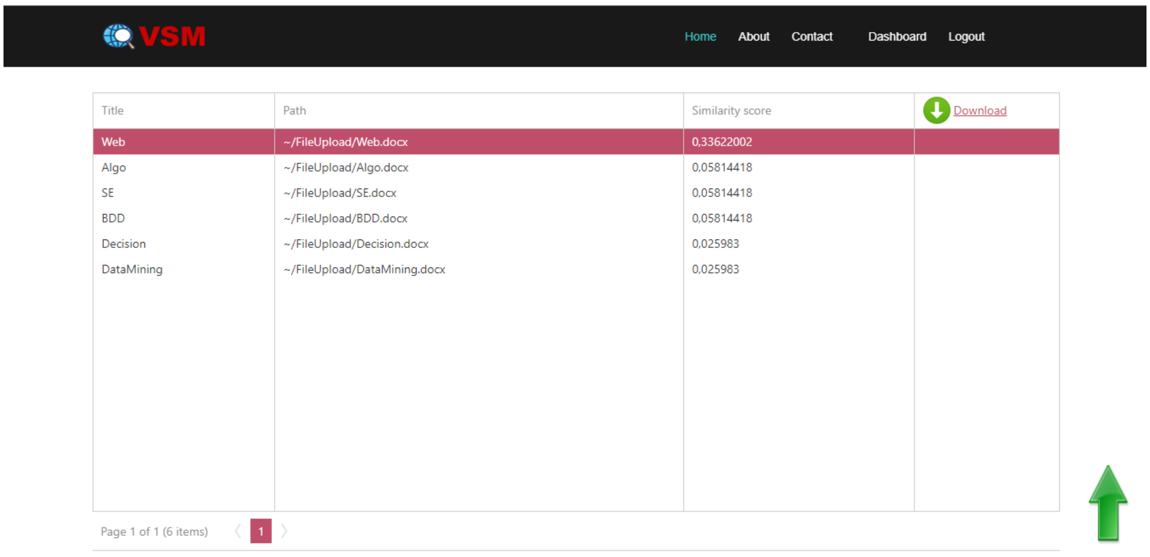


Figure 37: Search Home Page

Figure 38 shows example the result of search in home page.



Title	Path	Similarity score
Web	~/FileUpload/Web.docx	0.33622002
Algo	~/FileUpload/Algo.docx	0.05814418
SE	~/FileUpload/SE.docx	0.05814418
BDD	~/FileUpload/BDD.docx	0.05814418
Decision	~/FileUpload/Decision.docx	0.025983
DataMining	~/FileUpload/DataMining.docx	0.025983

Figure 38: Search Result Page

Figure 39 shows the page login and registration in our web site.

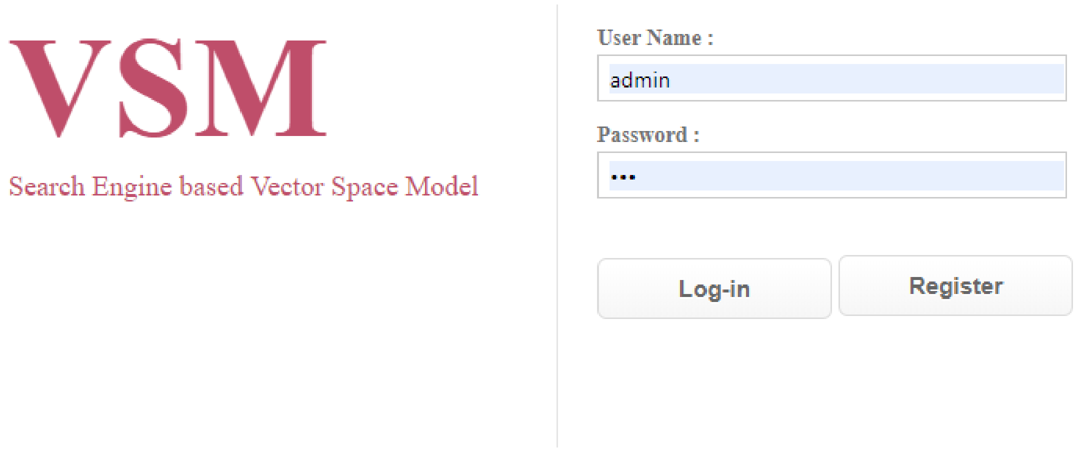
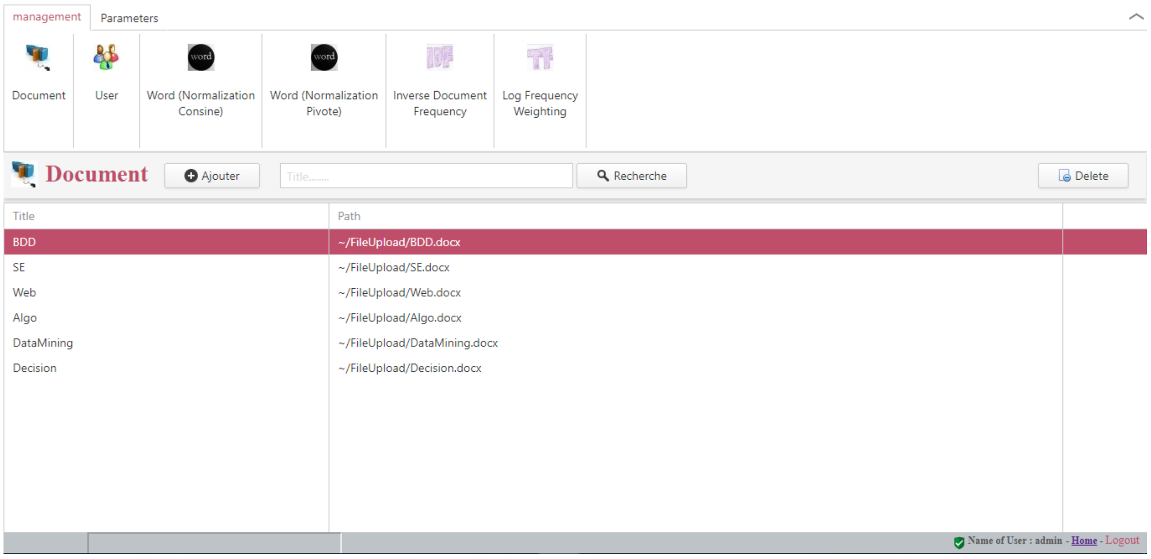


Figure 39: Login and Registration User Page

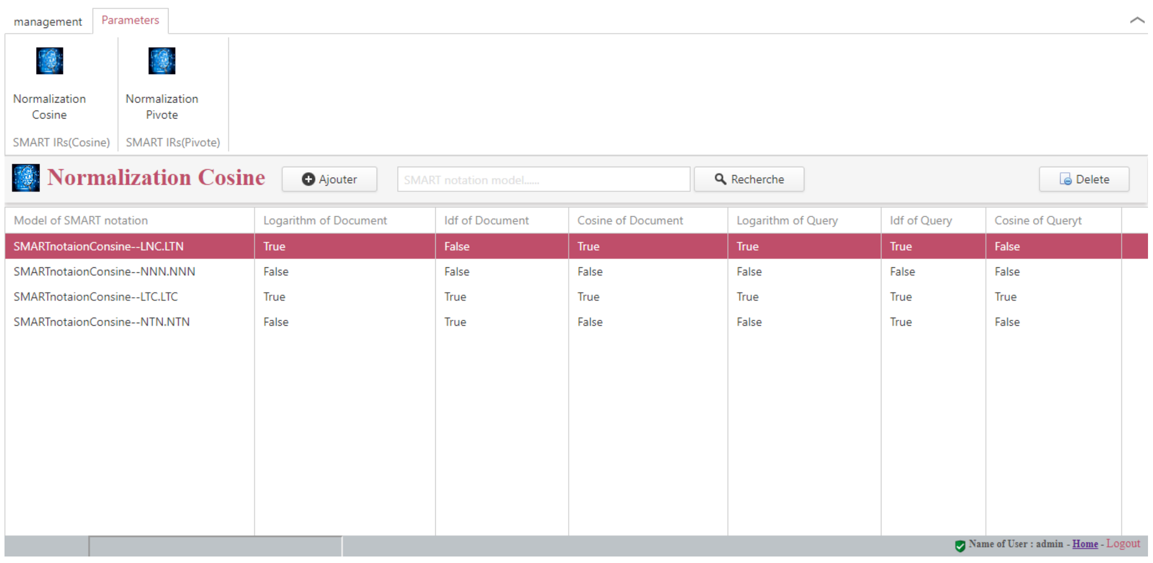
Figure 40 shows the management system (document, user) and view the all terms in corpus with the result of calculation parameters (TF, TF weight, DF, IDF, and TF-IDF).


management
Parameters
Document
User
Word (Normalization Consine)
Word (Normalization Pivote)
Inverse Document Frequency
Log Frequency Weighting
Document
Ajouter
Title.......
Recherche
Delete
Name of User : admin - Home - Logout

Title	Path
BDD	~/FileUpload/BDD.docx
SE	~/FileUpload/SE.docx
Web	~/FileUpload/Web.docx
Algo	~/FileUpload/Algo.docx
DataMining	~/FileUpload/DataMining.docx
Decision	~/FileUpload/Decision.docx

Figure 40: Management System Page

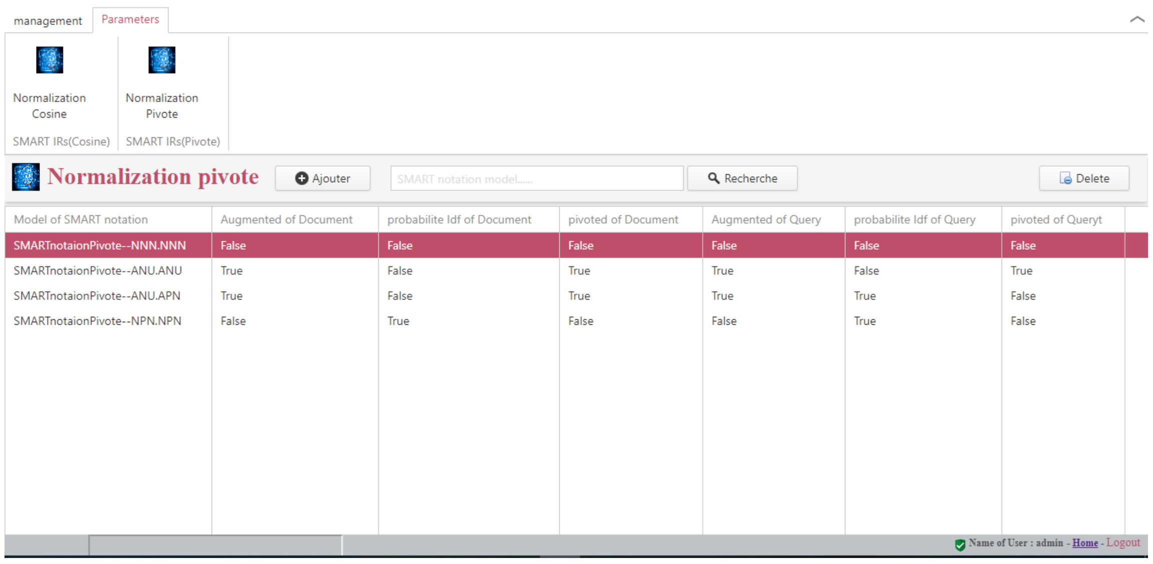
Figure 41 and figure 42 shows the management SMART notation used in the Search, there are two notation: normalization cosine and normalization pivote.



Model of SMART notation	Logarithm of Document	Idf of Document	Cosine of Document	Logarithm of Query	Idf of Query	Cosine of Queryt
SMARTnotaionConsine--LNC.LTN	True	False	True	True	True	False
SMARTnotaionConsine--NNN.NNN	False	False	False	False	False	False
SMARTnotaionConsine--LTC.LTC	True	True	True	True	True	True
SMARTnotaionConsine--NTN.NTN	False	True	False	False	True	False

Figure 41: Parameter Normalization Cosine Page

Figure 42 show the page of the management pivote SMART notation.



Model of SMART notation	Augmented of Document	probabilite Idf of Document	pivoted of Document	Augmented of Query	probabilite Idf of Query	pivoted of Queryt
SMARTnotaionPivote--NNN.NNN	False	False	False	False	False	False
SMARTnotaionPivote--ANU.ANU	True	False	True	True	False	True
SMARTnotaionPivote--ANU.APN	True	False	True	True	True	False
SMARTnotaionPivote--NPN.NPN	False	True	False	False	True	False

Figure 42: Parameter Normalization Pivote Page

4.6 CONCLUSION

We have shown in this section the application of the retrieval system. The scientific field is motivating and important for working in it. Dealing with documents written in a specific language is considered one of the many researches that attempt to develop information retrieval systems.

Part IV

GENERAL CONCLUSION

The last parts is the general conclusion and future work

5 GENERAL CONCLUSION

The work on information retrieval systems is one of the motivational and important areas. In our memory, we have introduced the application of the information retrieval system. The development of information retrieval systems is one of the continuous researches that deal with documents written in a specific language.

In our memory we have shown the main features of Vector Space Model and using web technologies we have implemented these features to cover documents written in English.

Improving performance to expand the features of the Vector Space Model is one of the things we will be working on in the future Which is considered a perspective, for example, we try to cover documents written in Arabic languages and finding a new SMART notation that gives us better search results and higher accuracy ratio. We also cover all PDF, HTML and other file formats, and make our site safe from any unwanted access to features or information on the site.

DECLARATION

Put your declaration here.

M'sila, 2019

saadoune billal, June 29, 2019

BIBLIOGRAPHY

[1] Jon Bentley. *Programming Pearls*. Addison–Wesley, Boston, MA, USA, 2nd edition, 1999.

[2] Robert Bringhurst. *The Elements of Typographic Style*. Version 3.2. Hartley & Marks Publishers, Point Roberts, WA, USA, 2008.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA, 2nd edition, 2001.

[4] Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, December 1974.

[5] Donald E. Knuth. Big Omicron and Big Omega and Big Theta. *SIGACT News*, 8(2):18–24, April/June 1976.

[6] Ian Sommerville. *Software Engineering*. Addison-Wesley, Boston, MA, USA, 4th edition, 1992.

Abstract

The great acceleration and wonderful developments in the processing of natural languages have facilitated many things. One of the areas that benefit from this development is information retrieval systems, so that the capabilities of the search engines are increased. There is also a development in the information retrieval models. In end of this memory,We have developed a system of information retrieval based on the technology of the vector space model, and works to measure the similarity between documents and query.

key words: information retrieval, vector space model, term frequency, term frequency weight, inverse document frequency, similarity ranking.. . .

Abstract arabic

Résumé

La grande accélération et les merveilleux développements dans le traitement des langues naturelles ont facilité beaucoup de choses. L'un des domaines qui bénéficient de ce développement est des systèmes de recherche d'information, de sorte que augmentent les capacités des moteurs de recherche. Il y a également un développement dans les modèles de recherche d'information. A la fin de cette mémoire, nous avons développé un système de recherche d'informations basé sur la technologie du modèle d'espace vectoriel et travaillons à la mesure de la similarité entre les documents et les requêtes.

Mots clés: recherche d'informations, modèle d'espace vectoriel, fréquence terme, poids fréquence terme, inverse frequence document, classement similarité.