DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH
UNIVERSITY OF BORDJ BOU ARRERIDJ
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTEMENT OF INFORMATICS

UNIVERSITE MOHAMED EL BACHIR EL IBRAHIMI
BORDJ BOU ARRERIDJ

THESIS

DISSERTATION SUBMITTED TO THE INFORMATICS DEPARTMENT IN CANDIDACY FOR MASTER DEGREE

SECTION: NETWORKS AND MULTIMEDIA

THEME

Vector Space Models Application for text comparison.

Submitted By:
Bouhafs Aimen

Academic Year: 2019/2020

DEDICATION

Wholeheartedly I am thankful and I am in indebted to my parents.

The source of love, encouragement and all the strength I needed and still in need!

To my Brothers, Sisters and all my family.

To all my friends and colleagues.

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful

All praises to Allah for the strengths and His blessing, who helped me and gave me patience and courage during this year and completing this thesis.

I would first like to warmly thank Mrs. Belalta Ramla who allowed me to benefit from her guidance. The advices she gave me, the patience and the confidence she showed in me were decisive in carrying out this research work.

My sincere thanks also go to the members of the jury, Mrs. Benseffia Hassina and Mrs. Chellakh Hafida for their interest in this research by agreeing to examine my modest work and enrich it with their proposals.

I also want to thank all of my teachers during the school years.

TABLE OF CONTENETS

LIST OF FIGURES

LIST OF EQUATIONS

Chapter 01

General Introduction

CHAPTER 1. GENERAL INTRODUCTION

1.1 Context:

Similarity has been a subject of great interest in human history since a long time ago. Even before computers were made, humans have been interested in finding similarity in everything. Each and every field of study provides their own definition of what similarity is. In Psychology similarity "…refers to the psychological nearness or proximity of two mental representations." while in music it's "…a certain similarity between two or more musical fragments "and in geometry "Two geometrical objects are called similar if they both have the same shape."

Similarity is a broad and abstract topic. Every time Similarity is mentioned, the question pops up: "What kind of Similarity?" The topic is quite big in the Information Retrieval field and lately it has become quite the hype again. People are making search engines, plagiarism programs, optimizing search algorithms, finding out how to specify the searches better and faster, and where to focus whether it be images, sound or strings.

Definitions for similarity are different for every field but what keeps going in each of them is the use of one big field to prove the similarity: Math.

Math is used to calculate similarity where it dominates the field. After the start of two new fields in last century, Information Theory and Computer Science, the topic of similarity has not become smaller at all. Instead by using the computer it has been easier to find out how similar two or more things are to each other.

Textual similarity which is a subcategory of Information Retrieval is quite close to its brother the search engine since both takes a query and find forth the similar texts for the query. Where it is expected from search engines to find the respective query's document of relevance and rank them, it is expected from the text similarity to find out how similar the query is to the documents. Both overlap a lot but are still a bit different. Even when there is no shortage of textual materials on a particular topic, procedures for indexing or extracting the knowledge or conceptual information contained in them can be lacking. Recently developed information retrieval systems (IRS) are based on the concept of a vector space Models.

1.2 Problematic:

The world witnesses a huge informational revolution that brings out lots of information and researches. Researching for this information without using search engines is a hard task or it is impossible to gain accurate information without them. Besides, the importance of information retrieval (IR) sciences has evolved. This science depended only on libraries in the past, but with the widespread and the evolution of digital libraries and the Internet has dramatically transformed the processing, storage, compare and retrieval of information. But this transformation brings a lot of problems, some of them are retrieving information becomes more difficult, also, one of the biggest problems is the plagiarism that becomes more and more withing the growth of information in the digital areas, and its detection becomes harder.

Detection of plagiarism can be undertaken in a variety of ways. Human detection is the most traditional form of identifying plagiarism from written work. This can be a lengthy and time-consuming task for the reader and can also result in inconsistencies in how plagiarism is identified within an organization. So, how can we make the detect the plagiarism without any human interaction easier? and how we can make information retrieving and classification more accurate and efficacy?

1.3 Objectives:

Nowadays, measuring the similarity of documents plays an important role in text related researches and applications such as document clustering, plagiarism detection, information retrieval, machine translation and automatic essay scoring.

Our objective is to develop and evaluate the performance of a text comparison system based on Vector Space Models which indicate the similarity between texts or documents.

1.4 Thesis plan:

In addition to this introductive chapter, our work will be divided into three chapters, 2nd chapter includes a state of art that defines and explains some of the measuring similarity models and give a little brief about the model we choose to study.

In the 3rd chapter, we will look deep in the vector space model we already choose, we will see its definitions, main techniques and approaches, then we will create our own model and explains the steps of creation in the meanwhile.

In the last chapter, we will implement our model using Python as programming language and PyCharm as an IDE, then we will see the interfaces and windows of our application and its results.

Chapter 02

State of Art

CHAPTER 2. STATE OF ART

2.1 Introduction:

Measuring similarity between documents is fundamental to most forms of document analysis. Some of the applications that use document similarity measures include: Information retrieval, text classification, document clustering, topic modeling, topic tracking, matrix decomposition. In the next section, we will see some of the main methods and algorithms that are used in similarity measuring.

2.2 Text similarity methods:

Different approaches have been promoted to measure the similarity between one text with another. The method is divided into four major groups, String-based, Corpus-based, Knowledge-based, and Hybrid text similarities; as shown in Fig. 1. These approaches will be discussed in the following subsections.

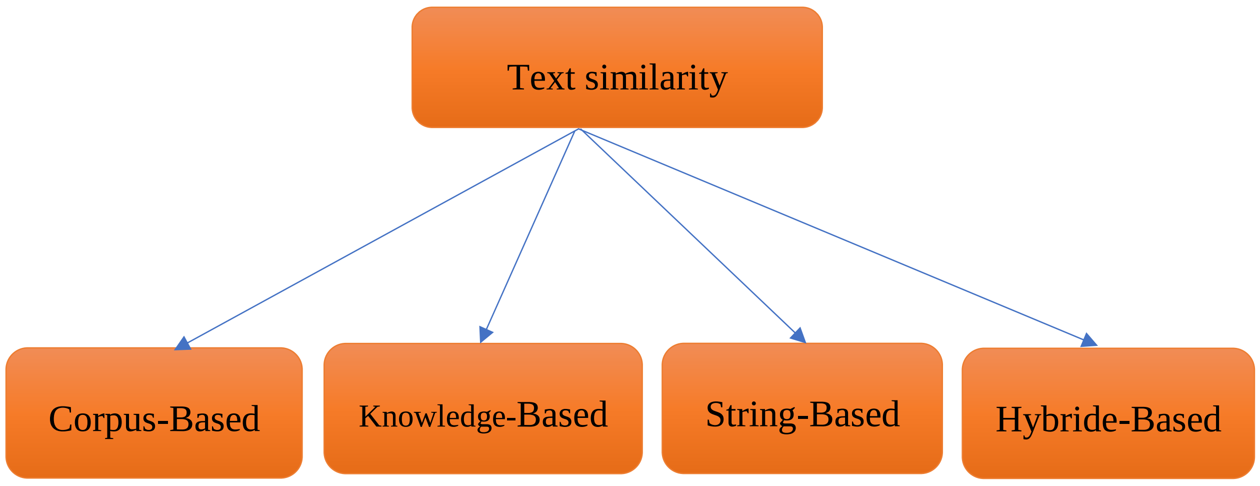


Figure 2. 1 Four major groups of text similarity methods and algorithms

2.2.1 Corpus-based Similarity:

Corpus-based similarity uses a semantic approach. This similarity approach determines the similarity between two concepts based on the information extracted from a large corpora. A corpus (plural corpora) is a large collection of electronic written or spoken text.

Corpus contains a predefined set of sentences and their translation to other language. The aim is to match input text with the text in the corpus and achieve translation [1]. Many corpus-based similarity or relatedness measures are based on concept-based resources, such as Wikipedia.

Some of corpus-based similarity measures are Hyperspace Analogue to Language (HAL), Latent Semantic Analysis (LSA), Explicit Semantic Analysis (ESA), Pointwise Mutual Information (PMI), Normalized Google Distance (NGD), and Extracting Distributional Similar words using Co-occurrence (DISCO). [2]

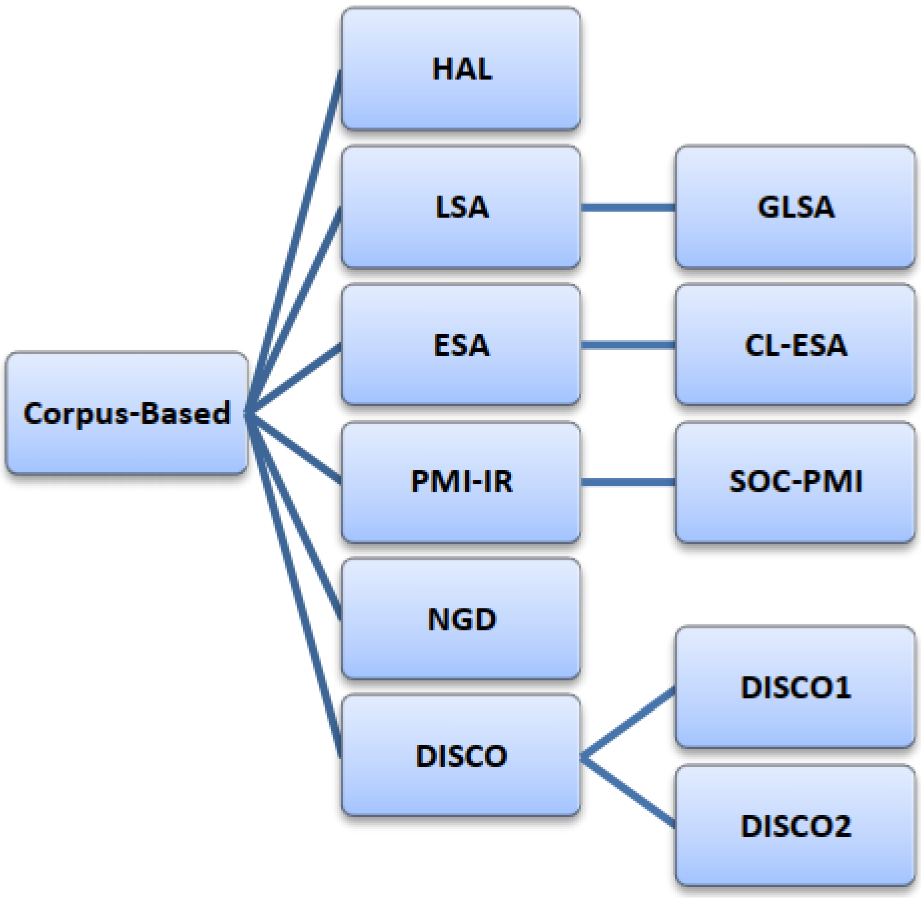


Figure 2. 2 Corpus-Based Similarity Measures[5]

2.2.2 Knowledge-Based Similarity:

Knowledge-Based Similarity is one of semantic similarity measures that bases on identifying the degree of similarity between words using information derived from semantic networks [3]. WordNet [4]is the most popular semantic network in the area of measuring the Knowledge-Based similarity between words; WordNet is a large lexical database of

English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations .As shown in figure 3, Knowledge-based similarity measures can be divided roughly into two groups: measures of semantic similarity and measures of semantic relatedness.[5]

There are six measures of semantic similarity; three of them are based on information content: Resnik (res), Lin (lin) and Jiang & Conrath (jcn). The other three measures are based on path length: Leacock & Chodorow (lch), Wu & Palmer (wup) and Path Length (path). Furthermore, there are three measures of semantic relatedness: St.Onge (hso) , Lesk (lesk) and vector pairs (vector). [5]

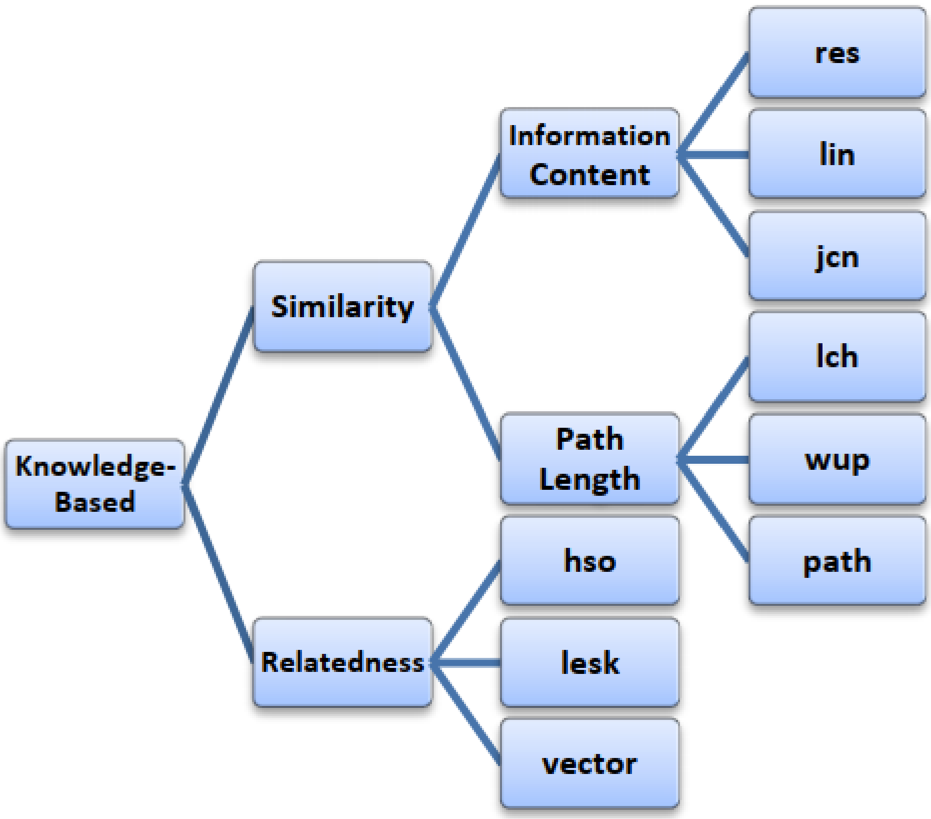


Figure 2. 3 Knowledge-Based Similarity Measures [5]

2.2.3 String-based Similarity:

String-based similarity is the oldest, simplest yet most popular measurement approach. This measure operates on string sequences and character composition. Two main types of string similarity functions are character-based similarity functions, and token-based similarity functions.

2.2.3.1 Character-based Similarity:

Is also called sequence-based or edit distance (ED) measurement. It takes two strings of characters and then calculates the edit distance (including insertion, deletion and substitution) between them. Character-based quantifies character similarity between two strings to quantify the similarity, for instance ED which is the minimum number of single-character edit operations needed to transform one to another [6]. In another word, two strings are similar if the edit distance minimum operation number is smaller than the given threshold. Some examples of this approach are Hamming distance, Levenshtein distance. Damerau-Levenshtein, Needleman-Wunsch, Longest Common Subsequence. Smith-Waterman, Jaro, JaroWinkler, and N-gram.[5]

2.2.3.2 The term-based similarity:

Also known as token-based because it models each string as a set of tokens. The similarity between strings can be assessed by manipulating sets of tokens, such as words. The main idea behind this approach is to perform two string similarity measurement based on general tokens, correspond to its token sets. [7] If the similarity is denoted, the string pair is flagged as being similar or duplicate. Term-based similarity address drawback on character-based when it works on larger string. [8]

The main characteristic of token-based similarity is the use of the overlap of two token sets as likeness quantification. The overlap is computed based on exactly matched token pairs without considering other similar tokens. Token-based similarity approach is useful for recognizing the term rearrangement by breaking the strings into substrings. Vector space models (Jaccard similarity, Dice's coefficient, Cosine similarity) are some examples of these methods. And it will be the core subject that we will treat in the next chapter.

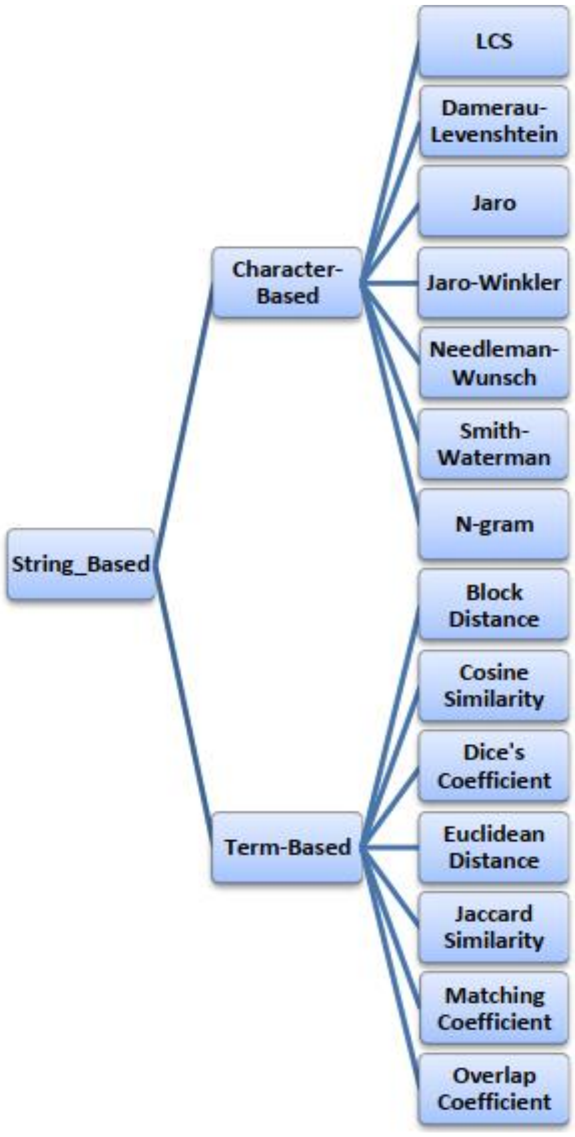


Figure 2. 4 String-Based Similarity Measures [5]

2.2.4 Hybrid Similarities:

In addition to the three categories previously described, there are still several similarity measures that cannot be categorized into any prior family. The idea of this approach is to combine the previously described approaches, including string-based, corpus-based, and knowledge-based similarity to reach a better metric by adopt their advantages.[5]

2.3 Token-based models' objectives:

Token-based similarities are very widely used in different areas. Probably, it is the most well-known approach to work with texts, that's because of its:

• Simplicity, since it is based on a linear algebra model.

• Ability to incorporate term weights any kind of term weight can be added.

• Can measure similarity between almost everything; query and document, document and document, query and query, sentence and sentences and so on.

• Partial matching is allowed.

• Ranking of documents compared to their relevance is allowed.

2.4 Conclusion:

In this chapter, we saw some of the main methods used in textual similarity measuring, and we introduce them and their algorithms, then we have seen the String-Based similarity method and its algorithms that called vector space models which would be our subject in the next chapters.

Chapter 03

Architecture and conception

CHAPTER 3. ARCHITECTURE AND CONCEPTION

3.1 Introduction:

In this chapter we will define the VSM (Vector Space Model), then we will explain the four main techniques of vector space model. After that we try to concept the model and apply those four techniques.

3.2 Vector Space Model:

3.2.1 Definitions:

Definition1: The Vector space model is a model where the document and the query both are represented in vector, each vector constructed of weight in a multidimensional space, whose dimensions are the terms used to build an index that represents documents

Definition2: A content-based model that represents a document as a vector in an n-dimensional space, where each dimension represents a term and similarity between two documents is measured through cosine angle between the two vectors [9]

Definition3: One of classical representations of document content. The documents are points (or vectors rooted in coordinate origin) in this high-dimensional space (spanned by terms being coordinate axes), with the point (vector) coordinates reflecting frequencies of different terms (linearly or in a more complex manner) in a given document [10]

Definition4: Is an algebraic model for representing documents (not only text) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings. Its first use was in the SMART Information Retrieval System [11]

3.2.2 Models and approaches:

As we said before, there is four main techniques in VSM, which are:

- Inner product
- Cosine similarity

- Jaccard index
- Dice index

3.2.2.1 Inner Product:

Inner product is the first technique in the vector space model. This technique considered as a base for other techniques. All the other techniques depend on the results of this technique to compute the results of their functions.

An inner product is a generalization of the dot product. In a vector space, it is a way to multiply vectors together, with the result of this multiplication being a scalar.

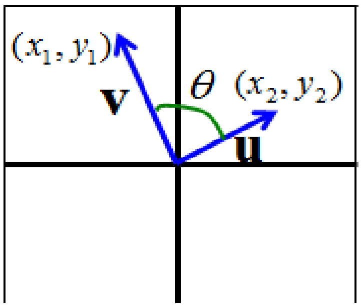


Figure 3. 1 representation of inner product

There are several different ways of representing/calculating the inner product. Equation (1) gives you the geometric meaning of inner product. Equation (2) would not shows you any idea of visualization, but it gives you a way of calculating the inner product with very simple multiplication and sums (Equation (2) would be the most common ways to calculate the inner product in most of the application)

$$U \cdot V = |U||V|\cos\theta$$

Equation 3. 1: Geometric Inner product

$$u \cdot v = x_1 \times x_2 + y_1 \times y_2$$

Equation 3. 2: Inner Product

The idea here is to implement our documents or texts as vectors and to find the similarity between these documents.

3.2.2.2 Cosine Similarity:

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. [12]

Let u and v be two vectors for comparison. Using the cosine measure as a similarity function, we have

$$sim_{cosine}(u,v) = cos\theta = \frac{u \cdot v}{\|u\|\|v\|} = \frac{x_1 \times x_2 + y_1 \times y_2}{\sqrt{{x_1}^2 + {y_1}^2} \times \sqrt{{x_2}^2 + {y_2}^2}}$$

Equation 3. 3: Cosine Similarity

The measure computes the cosine of the angle between vectors u and v. A cosine value of 0 means that the two vectors are at 90 degrees to each other (orthogonal) and have no match, which means the documents we are comparing are quietly different. The closer the cosine value to 1, the smaller the angle and the greater the match between documents.

3.2.2.3 Jaccard Similarity:

" Also known as Jaccard index, the Jaccard similarity coefficient is a statistical measure of similarity between sample sets" [13]

The Jaccard index or Jaccard coefficient [14] is the ratio between the cardinality (the size) of the intersection of the sets considered and the cardinality of the union of the sets. It allows to evaluate the similarity between the sets. The documents d1 and d2 are therefore represented as sets of terms. The similarity obtained $sim_{jaccard} \in [0,1]$

$$sim_{jaccard}(d1, d2) = \frac{\|d1 \cap d2\|}{\|d1 \cup d2\|}$$

Equation 3. 4 : Jaccard Similarity for sets

It is also possible to use vector weighted representation:

$$sim_{jaccard} = \frac{u \cdot v}{u^2 + v^2 - u \cdot v}$$

$$sim_{jaccard} = \frac{x_1 \times x_2 + y_1 \times y_2}{(x_1 + y_1)^2 + (x_2 + y_2)^2 - x_1 \times x_2 + y_1 \times y_2}$$

Equation 3. 5: Jaccard Similarity for vectors

3.2.2.4 Dice Similarity:

The D index measures the similarity between two documents d1 and d2 based on the number of terms common between d1 and d2. Dice measurement is used like Jaccard to find the similarity between two vectors but gives twice the weight to agreements, the dice similarity measure [15] obtained by the formula:

$$sim_{dice}(d1, d2) = \frac{2N_c}{N1 + N2}$$

Equation 3. 6: Dice Similarity for sets

Where N_C is the number of common words between d1 and d2, and N1(resp. N2) is the number of terms in d1(resp. d2)

While the vector weighted calculation can be done with the formula:

$$sim_{dice}(d1, d2) = 2\,\frac{u \cdot v}{u^2 + v^2}$$

$$sim_{dice}(u, v) = 2\,\frac{x_1 \times x_2 + y_1 \times y_2}{(x_1 + y_1)^2 + (x_2 + y_2)^2}$$

Equation 3. 7: Dice Similarity for vectors

3.2.3 Term Weighting:

Term weighting is a procedure that takes place during the text indexing process in order to assess the value of each term to the document. Term weighting is the assignment of numerical values to terms that represent their importance in a document in order to improve retrieval effectiveness [16]

Essentially it considers the relative importance of individual words in an information retrieval system, which can improve system effectiveness, since not all the terms in a given document collection are of equal importance. Weighing the terms is the means that enables the retrieval system to determine the importance of a given term in a certain document or a query. It is a crucial component of any information retrieval system, a component that has shown great potential for improving the retrieval effectiveness of an information retrieval system [17]. One of the most important term weighting method is TF-IDF.

3.2.3.1 TF-IDF weighting:

tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears

3.2.3.2 Term Frequency (TF):

which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$TF(t,d) = \frac{count\ of\ t\ in\ d}{number\ of\ words\ in\ d}$$

Equation 3. 8: Term Frequency

Where t: term, d: document

3.2.3.3 Document frequency (DF):

This measures the importance of document in whole set of corpora, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, where DF is the count of occurrences of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

$$DF(t) = occurrence\ of\ t\ in\ documents$$

3.2.3.4 Inverse Document Frequency (IDF):

The Inverse Document Frequency which is considered to be a discriminating measure for a term in the text collection. It was proposed in 1972, and has since been widely used. IDF in information retrieval is used to distinguish words that have the same frequency. [18]

$$IDF(t) = \log\frac{N}{df_t} + 1$$

Equation 3. 9: Inverse Document Frequency

Where, N is total number of documents, df_t is document frequency of a term t

3.2.3.5 Term frequency–Inverse document frequency(tf.idf):

now combine the definitions of term frequency (the importance of each index term in the document(tf)and inverse document frequency (the importance of the index term in the text collection), to produce a composite weight for each term in each document

$$w_{t,d} = tf(t,d) * idf(t) = tf(t,d) * \log\frac{N}{df_t} + 1$$

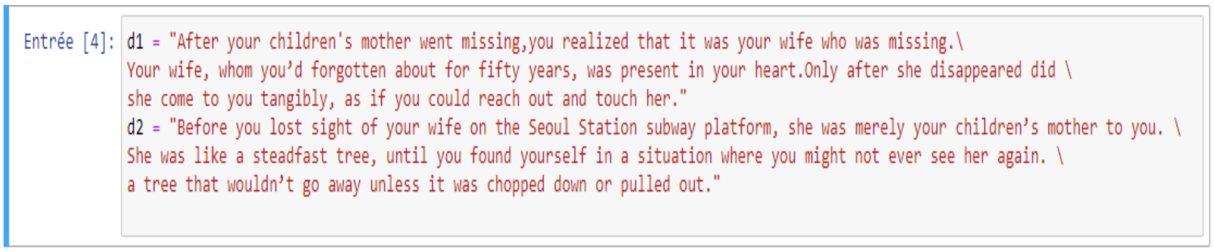
Equation 3. 10: TF.IDF

3.3 Conception:

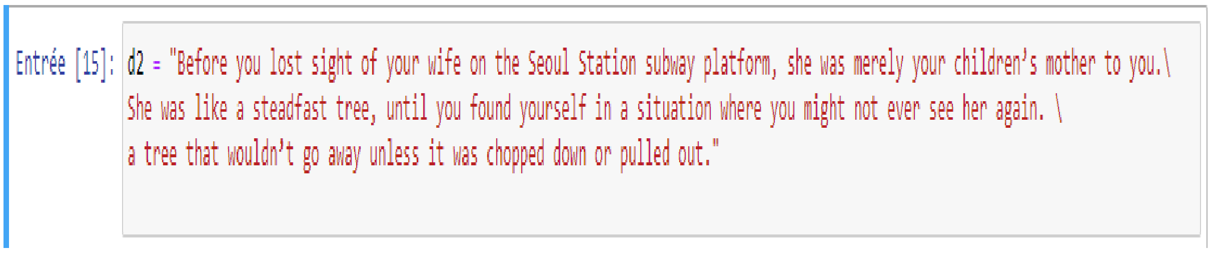
Before building the VSM model to calculate similarity, there are a few preprocessing to do, this steps called the NLP Pipeline. the pipeline includes the following:

- Tokenization
- Punctuation and Stop Words Removal
- Stemming or Lemmatization
- Creating the Bag of Words
- building a VSM model

To show the process of building our mode, we have chosen two sample texts extracted from the book “Please Look After Mom” by “Kyung-sook Shin”



```
Entrée [4]: d1 = "After your children's mother went missing,you realized that it was your wife who was missing.\
Your wife, whom you'd forgotten about for fifty years, was present in your heart.Only after she disappeared did \
she come to you tangibly, as if you could reach out and touch her."
d2 = "Before you lost sight of your wife on the Seoul Station subway platform, she was merely your children's mother to you. \
She was like a steadfast tree, until you found yourself in a situation where you might not ever see her again. \
a tree that wouldn't go away unless it was chopped down or pulled out."
```

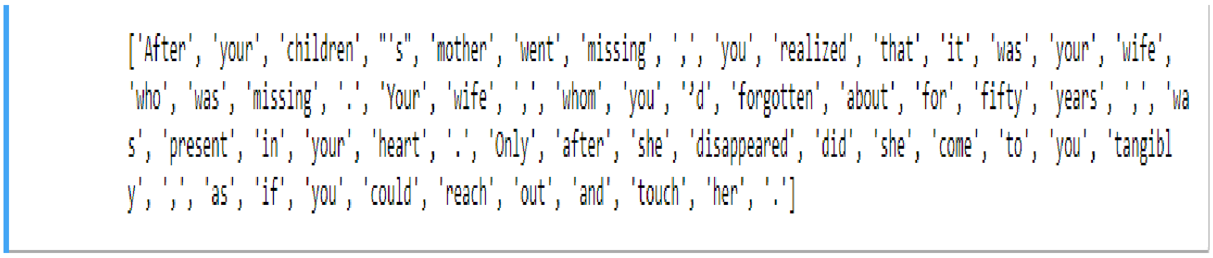


```
Entrée [15]: d2 = "Before you lost sight of your wife on the Seoul Station subway platform, she was merely your children's mother to you.\
She was like a steadfast tree, until you found yourself in a situation where you might not ever see her again. \
a tree that wouldn't go away unless it was chopped down or pulled out."
```

Figure 3. 2 Text samples used for conceptions

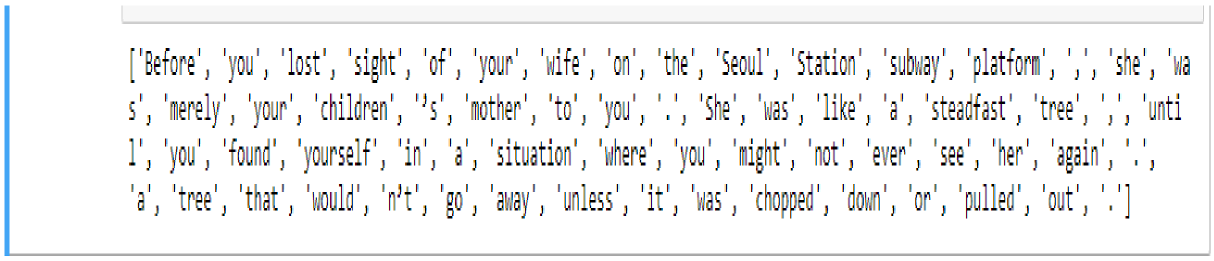
3.3.1 Tokenization:

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, this will produce us a list of single items to process.



```
['After', 'your', 'children', "'s", 'mother', 'went', 'missing', ',', 'you', 'realized', 'that', 'it', 'was', 'your', 'wife',
'who', 'was', 'missing', '.', 'Your', 'wife', ',', 'whom', 'you', ''d', 'forgotten', 'about', 'for', 'fifty', 'years', ',', 'wa
s', 'present', 'in', 'your', 'heart', '.', 'Only', 'after', 'she', 'disappeared', 'did', 'she', 'come', 'to', 'you', 'tangibl
y', ',', 'as', 'if', 'you', 'could', 'reach', 'out', 'and', 'touch', 'her', '.']
```

Figure 3. 3 Tokenization of first document

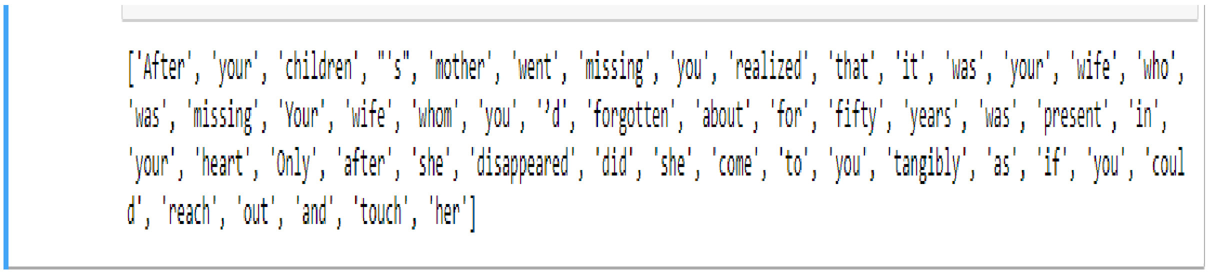


```
['Before', 'you', 'lost', 'sight', 'of', 'your', 'wife', 'on', 'the', 'Seoul', 'Station', 'subway', 'platform', ',', 'she', 'wa
s', 'merely', 'your', 'children', ''s', 'mother', 'to', 'you', '.', 'She', 'was', 'like', 'a', 'steadfast', 'tree', ',', 'unti
l', 'you', 'found', 'yourself', 'in', 'a', 'situation', 'where', 'you', 'might', 'not', 'ever', 'see', 'her', 'again', '.',
'a', 'tree', 'that', 'would', 'n't', 'go', 'away', 'unless', 'it', 'was', 'chopped', 'down', 'or', 'pulled', 'out', '.']
```

Figure 3. 4 Tokenization of second document

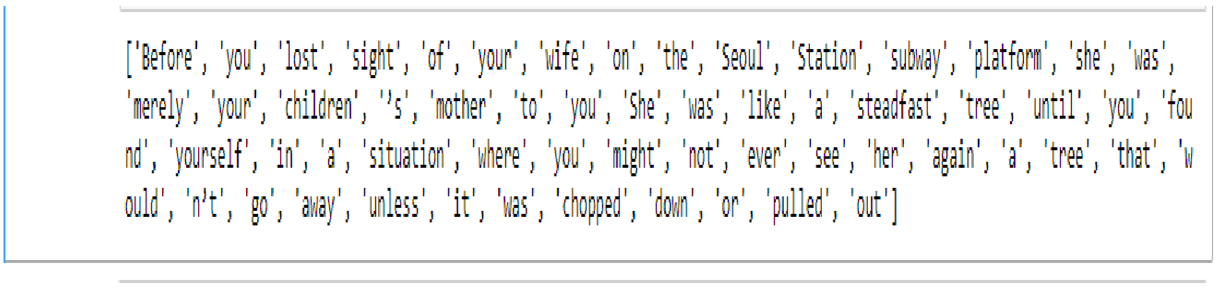
3.3.2 Punctuation Removal:

Removing punctuation is the process of deleting all the punctuation marks as dots, comma, and quotes.



```
['After', 'your', 'children', "'s", 'mother', 'went', 'missing', 'you', 'realized', 'that', 'it', 'was', 'your', 'wife', 'who',
'was', 'missing', 'Your', 'wife', 'whom', 'you', '’d', 'forgotten', 'about', 'for', 'fifty', 'years', 'was', 'present', 'in',
'your', 'heart', 'Only', 'after', 'she', 'disappeared', 'did', 'she', 'come', 'to', 'you', 'tangibly', 'as', 'if', 'you', 'coul
d', 'reach', 'out', 'and', 'touch', 'her']
```

Figure 3. 5 Punctuation removal from d1

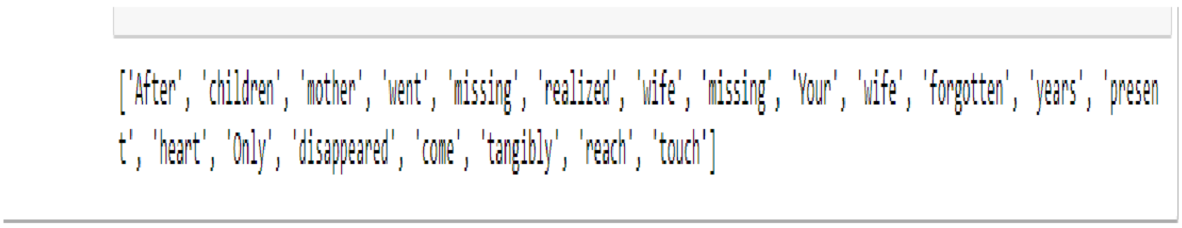


```
['Before', 'you', 'lost', 'sight', 'of', 'your', 'wife', 'on', 'the', 'Seoul', 'Station', 'subway', 'platform', 'she', 'was',
'merely', 'your', 'children', '’s', 'mother', 'to', 'you', 'She', 'was', 'like', 'a', 'steadfast', 'tree', 'until', 'you', 'fou
nd', 'yourself', 'in', 'a', 'situation', 'where', 'you', 'might', 'not', 'ever', 'see', 'her', 'again', 'a', 'tree', 'that', 'w
ould', 'n’t', 'go', 'away', 'unless', 'it', 'was', 'chopped', 'down', 'or', 'pulled', 'out']
```

Figure 3. 6 Punctuation removal from d2

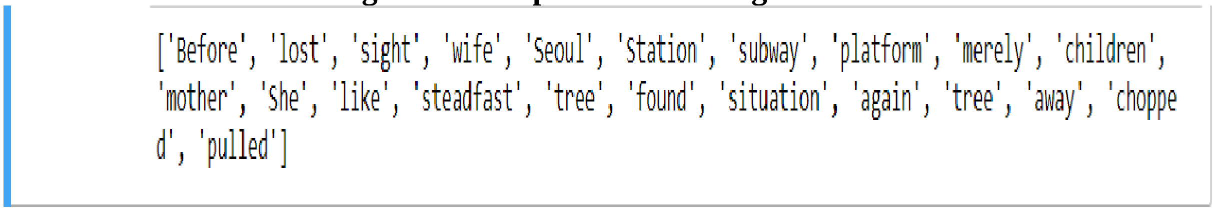
3.3.3 Stop-word removal:

Stop-word removal is the process for deleting all the words that have no meaning. Stop words include the large number of prepositions, pronouns, conjunctions, etc. in sentences. These words need to be removed before we analyze the text, so that the frequently used words are mainly the words relevant to the context and not common words used in the text



```
['After', 'children', 'mother', 'went', 'missing', 'realized', 'wife', 'missing', 'Your', 'wife', 'forgotten', 'years', 'presen
t', 'heart', 'Only', 'disappeared', 'come', 'tangibly', 'reach', 'touch']
```

Figure 3. 7 Stop word removing from d1



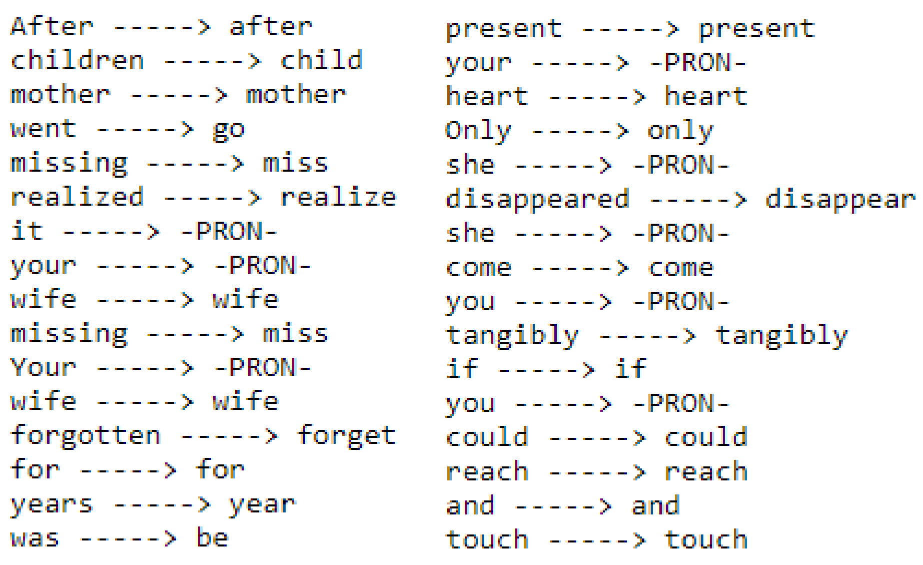
```
['Before', 'lost', 'sight', 'wife', 'Seoul', 'Station', 'subway', 'platform', 'merely', 'children',
'mother', 'She', 'like', 'steadfast', 'tree', 'found', 'situation', 'again', 'tree', 'away', 'choppe
d', 'pulled']
```

Figure 3. 8 Stop word removing from d2

3.3.4 Lemmatization:

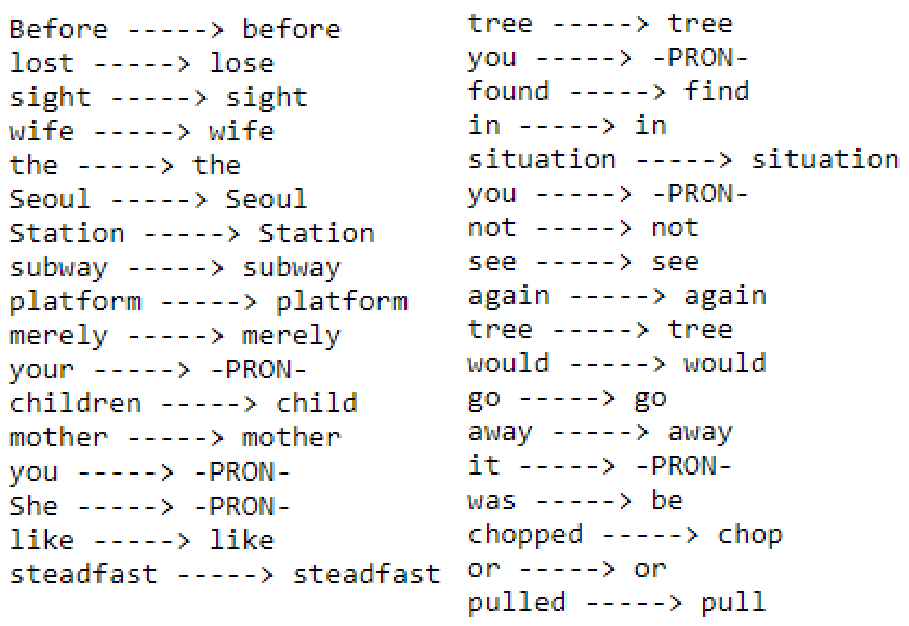
Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base LEMMA or dictionary form of a word, which is known as the lemma

Its result would be:



```
After -----> after
children -----> child
mother -----> mother
went -----> go
missing -----> miss
realized -----> realize
it -----> -PRON-
your -----> -PRON-
wife -----> wife
missing -----> miss
Your -----> -PRON-
wife -----> wife
forgotten -----> forget
for -----> for
years -----> year
was -----> be
present -----> present
your -----> -PRON-
heart -----> heart
Only -----> only
she -----> -PRON-
disappeared -----> disappear
she -----> -PRON-
come -----> come
you -----> -PRON-
tangibly -----> tangibly
if -----> if
you -----> -PRON-
could -----> could
reach -----> reach
and -----> and
touch -----> touch
```

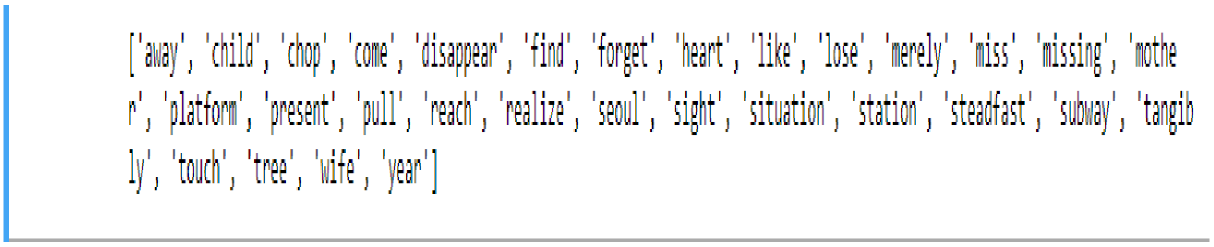
Figure 3. 9 Lemmatization of d1



```
Before -----> before
lost -----> lose
sight -----> sight
wife -----> wife
the -----> the
Seoul -----> Seoul
Station -----> Station
subway -----> subway
platform -----> platform
merely -----> merely
your -----> -PRON-
children -----> child
mother -----> mother
you -----> -PRON-
She -----> -PRON-
like -----> like
steadfast -----> steadfast
tree -----> tree
you -----> -PRON-
found -----> find
in -----> in
situation -----> situation
you -----> -PRON-
not -----> not
see -----> see
again -----> again
tree -----> tree
would -----> would
go -----> go
away -----> away
it -----> -PRON-
was -----> be
chopped -----> chop
or -----> or
pulled -----> pull
```

Figure 3. 10 lemmatization of d2

3.3.5 Creating words-bag:

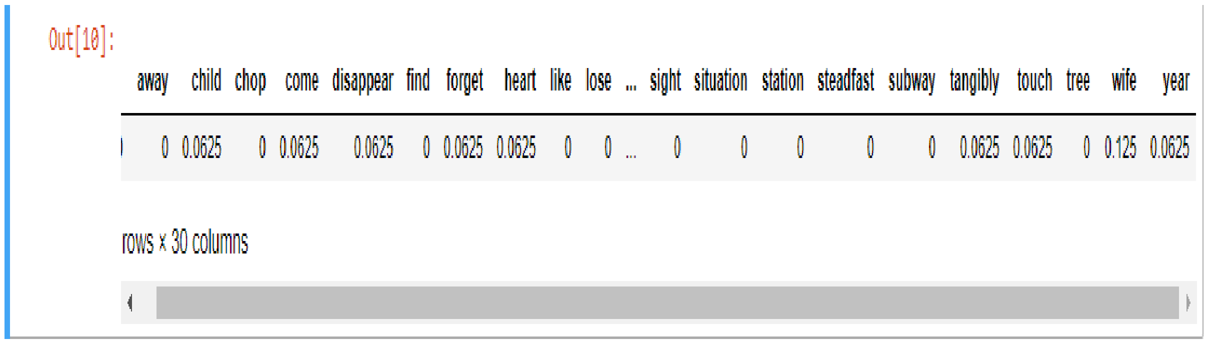


```
['away', 'child', 'chop', 'come', 'disappear', 'find', 'forget', 'heart', 'like', 'lose', 'merely', 'miss', 'missing', 'mothe
r', 'platform', 'present', 'pull', 'reach', 'realize', 'seoul', 'sight', 'situation', 'station', 'steadfast', 'subway', 'tangib
ly', 'touch', 'tree', 'wife', 'year']
```

Figure 3. 11 Creation of words-bag from d1 and d2

After we create our words-bag, and normalize the two texts, now we move to term weighting.

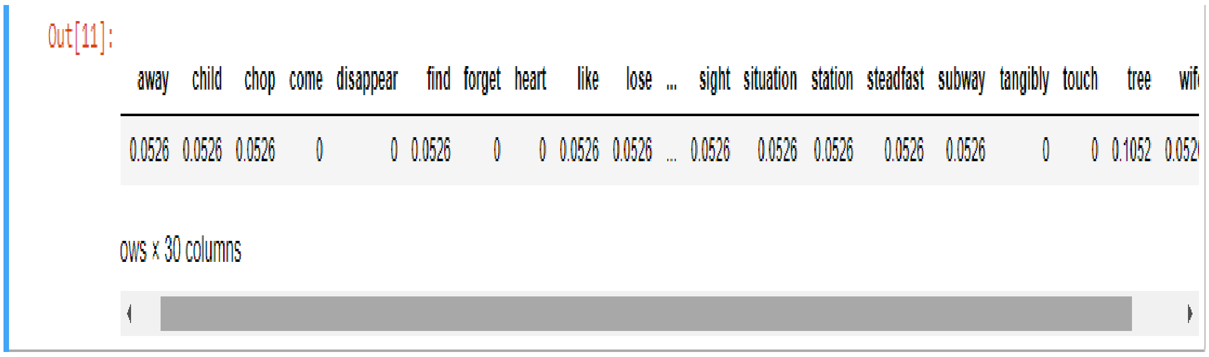
3.3.6 Compute the term frequency:



```
Out[10]:
    away  child  chop  come  disappear  find  forget  heart  like  lose  ...  sight  situation  station  steadfast  subway  tangibly  touch  tree  wife  year
|      0 0.0625     0 0.0625    0.0625     0 0.0625 0.0625     0     0  ...      0          0        0          0       0    0.0625 0.0625     0 0.125 0.0625

rows × 30 columns
```

Figure 3. 12 Term weights of d1

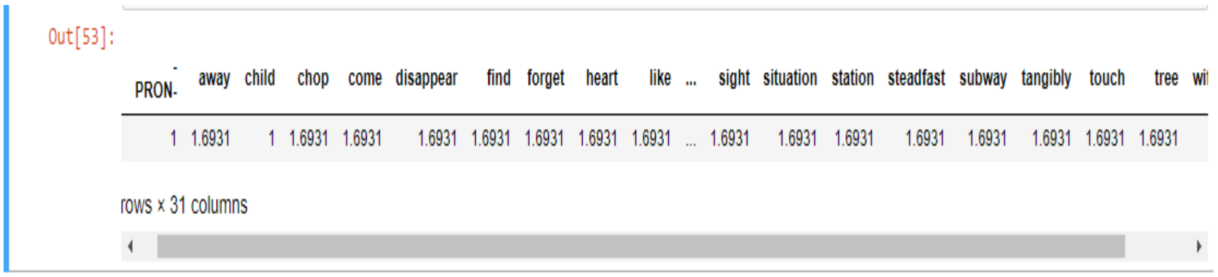


```
Out[11]:
    away  child   chop  come  disappear    find  forget  heart    like    lose  ...   sight  situation  station  steadfast  subway  tangibly  touch    tree   wif
  0.0526 0.0526 0.0526     0          0  0.0526       0      0  0.0526  0.0526  ...  0.0526     0.0526   0.0526     0.0526  0.0526         0      0  0.1052  0.052

ows × 30 columns
```

Figure 3. 13 Term weights of d2

Now, we calculate our IDF, which would be the same for both documents



Out[53]:

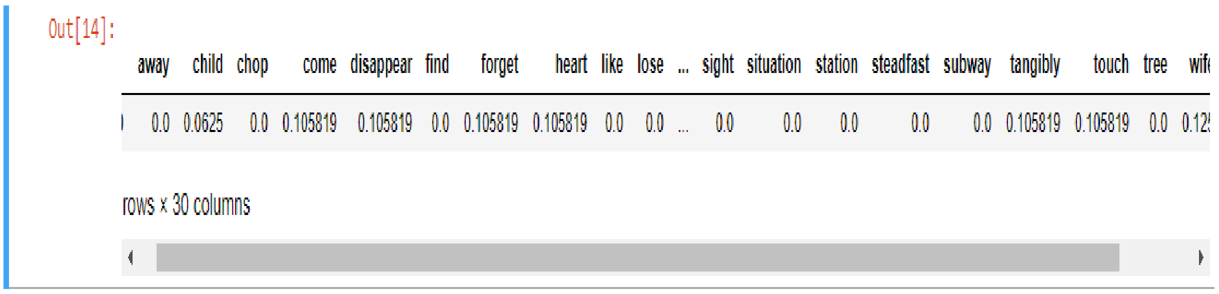
-PRON-	away	child	chop	come	disappear	find	forget	heart	like	...	sight	situation	station	steadfast	subway	tangibly	touch	tree	wi
1	1.6931	1	1.6931	1.6931	1.6931	1.6931	1.6931	1.6931	1.6931	...	1.6931	1.6931	1.6931	1.6931	1.6931	1.6931	1.6931	1.6931	

rows × 31 columns

Figure 3. 14 the inverse document frequency weights

After calculation the Term frequency and inversed term frequency, we compute TF-IDF to get the final indexed files in order to apply the VSM techniques and calculate the similarity between those documents

The TF-IDF for the first file is going to be:



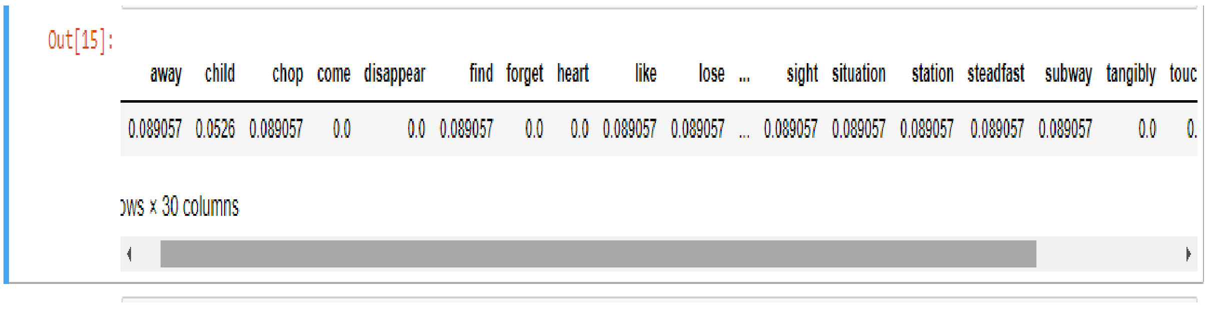
Out[14]:

away	child	chop	come	disappear	find	forget	heart	like	lose	...	sight	situation	station	steadfast	subway	tangibly	touch	tree	wif
0.0	0.0625	0.0	0.105819	0.105819	0.0	0.105819	0.105819	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.105819	0.105819	0.0	0.12

rows × 30 columns

Figure 3. 15 TF-IDF weights of d1

And for the second document:



Out[15]:

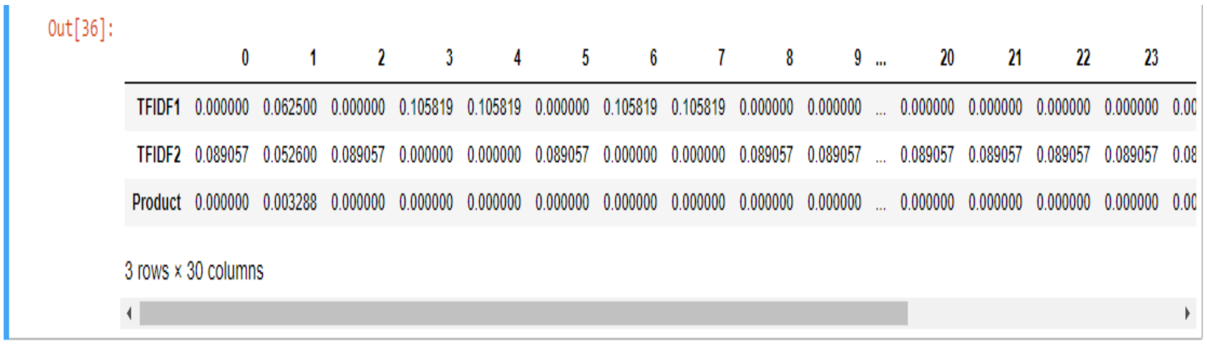
away	child	chop	come	disappear	find	forget	heart	like	lose	...	sight	situation	station	steadfast	subway	tangibly	touc
0.089057	0.0526	0.089057	0.0	0.0	0.089057	0.0	0.0	0.089057	0.089057	...	0.089057	0.089057	0.089057	0.089057	0.089057	0.0	0.

ws × 30 columns

Figure 3. 16 TF-IDF weights of d2

3.4 Applying the vector space model Techniques:

Like that, we are ready to measure similarity using deferent techniques, but in order to do that, we have first to calculate the length of our vectors and the dot product of them



Out[36]:

	0	1	2	3	4	5	6	7	8	9	...	20	21	22	23	
TFIDF1	0.000000	0.062500	0.000000	0.105819	0.105819	0.000000	0.105819	0.105819	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00
TFIDF2	0.089057	0.052600	0.089057	0.000000	0.000000	0.089057	0.000000	0.000000	0.089057	0.089057	...	0.089057	0.089057	0.089057	0.089057	0.08
Product	0.000000	0.003288	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.00

3 rows × 30 columns

$dot(d1, d2) = \sum w_{i,d1} * w_{i,d2}$ where $w_{i,\,d1}$ is the weight of term i in document d1, and $w_{i,d2}$

Is the weight of term i in document d2

$dot(d1, d2) = 0.0625 * 0.0526 + 0.10581 * 0.0000 + \cdots + 0.125 * 0.0256$
$dot\ (d1, d2) = 0.01315$

Now, the length of d1 and d2:

$$len(d) = \sqrt{{x_1}^2 + {x_2}^2 + \cdots + {x_i}^2}$$

Equation 3. 11: Vectors Length

$len(d1) = \sqrt{0.0625^2 + 0.105819^2 + \cdots + 0.125^2}$

$len(d1) = 0.3972$

$len(d2) = \sqrt{0.08905^2 + 0.05260^2 + 0.08905^2 + \cdots + 0.0526^2}$

$len(d2) = 0.3886$

Cosine Similarity:

As we introduced the cosine similarity, it is used to compute the angel between the vectors for documents and query. We will apply the below equation of cosine similarity on our example:

$$Sim_{cosine}(d1, d2) = cos\theta = \frac{\overrightarrow{d1} \cdot \overrightarrow{d2}}{\|\overrightarrow{d1}\|\|\overrightarrow{d2}\|}$$

$$Sim_{cosine}(d1,d2) = cos\theta = \frac{\overrightarrow{d1}\cdot\overrightarrow{d2}}{\|\overrightarrow{d1}\|\|\overrightarrow{d2}\|} = \frac{0.01315}{0.3972 * 0.3886}$$

$$Sim_{cosine}(d1,d2) = 0.08516$$

$$Sim_{cosine}(d1,d2) = 08{,}52\%$$

Jaccard Similarity:

$$sim_{jaccard}\frac{\overrightarrow{d1}\cdot\overrightarrow{d2}}{\overrightarrow{d1}^2+\overrightarrow{d2}^2-\overrightarrow{d1}\cdot\overrightarrow{d2}}$$

$$sim_{jaccard} = \frac{0.01315}{0.1578+0.15106-0.01315}$$

$$sim_{jaccard} = 0.0444$$

$$sim_{jaccard} = 04{,}44\%$$

Dice Similarity:

$$sim_{dice}(d1,d2) = 2\frac{\overrightarrow{d1}\cdot\overrightarrow{d2}}{\overrightarrow{d1}^2+\overrightarrow{d2}^2}$$

$$sim_{dice}(d1,d2) = 2\frac{0.01315}{0.1578+0.15106}$$

$$sim_{dice}(d1,d2) = 0.08514$$

$$sim_{dice}(d1,d2) = 08{,}51\%$$

3.5 Conclusion:

In this chapter, we have defined what is a vector space model, it’s main techniques and approaches, then the main steps for implementing it to build a comparison system. We will show on the next chapter the tool used to apply those techniques in vector space model and it’s results as well.

Chapter 04

Implementation

CHAPTER 4. IMPLEMENTATION

4.1 Introduction:

This chapter represents the last step of our work, which mean the implementing of our model and discuss the results in the right environment. In this step we will choose the environment, the tools and the language used to develop our model, then we will give an overview of the work done in form of codes then the last result of our work as a desktop application for comparing two texts or documents.

4.2 Programming Language:

To develop our model, we choose Python as a programming language because it's easy to use, powerful and versatile (image processing, game developing, data science analyzing...)

4.2.1 Python Presentation:

Python is an interpreted, high-level and general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. [19]

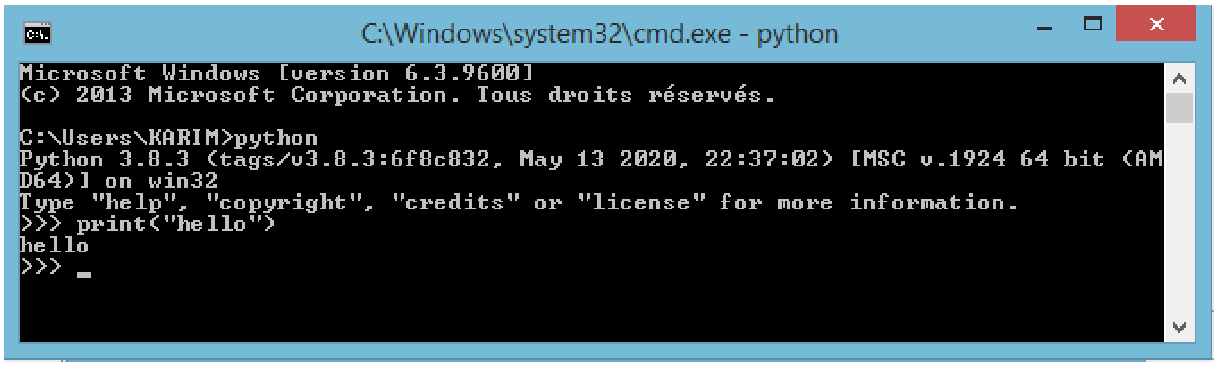
Python has so many features, we mention:

- Built-in high-level data types: strings, lists, dictionaries, etc.
- The usual control structures: if, ifelse, ifelifelse, while, plus a powerful collection iterator (for).
- Multiple levels of organizational structure: functions, classes, modules, and packages. These assist in organizing code.
- Compile on the fly to byte code Source code is compiled to byte code without a separate compile step.

4.2.2 Interactive Python:

If you execute Python from the command line with no script (no arguments), Python gives you an interactive prompt. This is an excellent facility for learning Python and for trying small snippets of code.

Start the Python interactive interpreter by typing “python” with no arguments at the command line. For example:



```
Microsoft Windows [version 6.3.9600]
(c) 2013 Microsoft Corporation. Tous droits réservés.

C:\Users\KARIM>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello")
hello
>>> _
```

Figure 4. 1 Executing Python from CMD

4.3 Environment and tools:

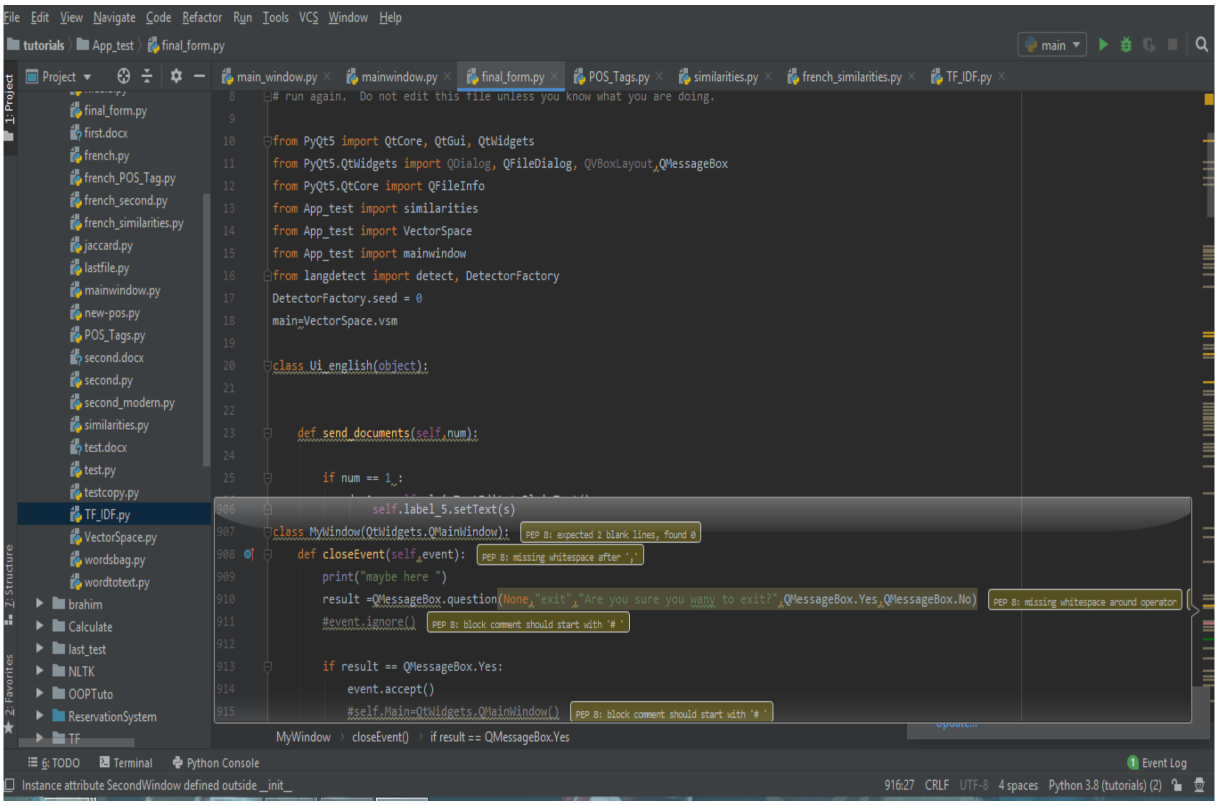
We have chosen PyCharm IDE (integrated development environment), version 2019.3 under windows 8.1 as an operation system. We choose it because it’s the most widely used IDEs for Python programming language, and it provides coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes. We also use PyQt designer to create the application interfaces, we choose it because it’s easy and handy to use and use drag to design style.

4.3.1 Presentation of PyCharm:

PyCharm Is an IDE used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

Chapter 4
Implementation

PyCharm was developed by jetBrains as a cross-platform IDE for Python. In addition to supporting versions 2.x and 3.x of Python, PyCharm is also compatible with Windows, Linux, and macOS.



```
# run again.  Do not edit this file unless you know what you are doing.

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QDialog, QFileDialog, QVBoxLayout,QMessageBox
from PyQt5.QtCore import QFileInfo
from App_test import similarities
from App_test import VectorSpace
from App_test import mainwindow
from langdetect import detect, DetectorFactory
DetectorFactory.seed = 0
main=VectorSpace.vsm

class Ui_english(object):


    def send_documents(self,num):

        if num == 1 :
                self.label_5.setText(s)
class MyWindow(QtWidgets.QMainWindow):      PEP 8: expected 2 blank lines, found 0
    def closeEvent(self,event):      PEP 8: missing whitespace after ','
        print("maybe here ")
        result =QMessageBox.question(None,"exit","Are you sure you wany to exit?",QMessageBox.Yes,QMessageBox.No)      PEP 8: missing whitespace around operator
        #event.ignore()      PEP 8: block comment should start with '# '

        if result == QMessageBox.Yes:
            event.accept()
            #self.Main=QtWidgets.QMainWindow()      PEP 8: block comment should start with '# '
```

Figure 4. 2 PyCharm interface

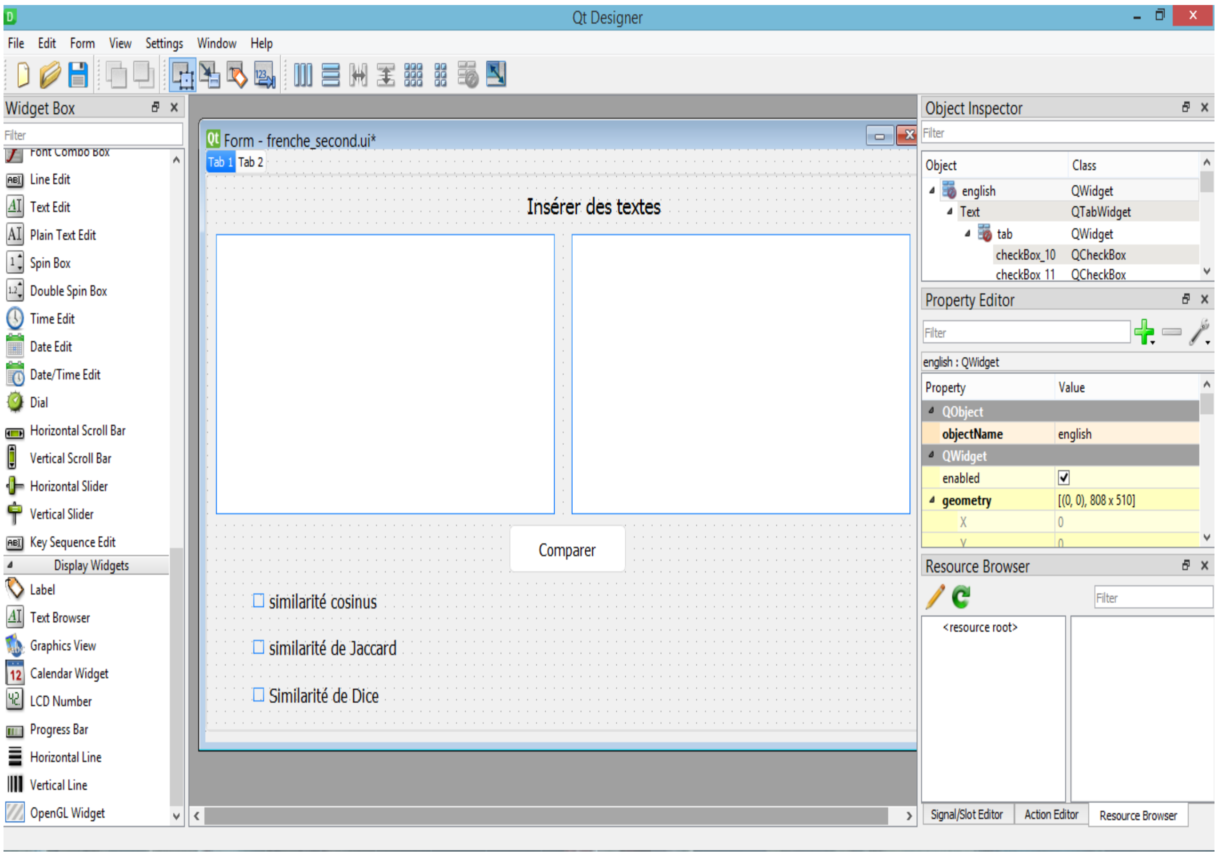
4.3.2 PyQt presentation:

PyQt is one of the most popular Python bindings for the Qt cross-platform C++ framework, implemented as a Python plug in. PyQt is free software developed by the British firm Riverbank Computing. PyQt supports Microsoft Windows as well as various editions of UNIX, including Linux and MacOS.

PyQt is available in two editions: PyQt4 which will build against Qt 4.x and 5.x and PyQt5 which will only build against 5.x. Both editions can be built for Python 2 and 3. PyQt contains over 620 classes that cover graphical user interfaces, XML handling, network communication, SQL databases, Web browsing and other technologies available in Qt.

4.3.3 PyQt designer:

Qt Designer is the Qt tool for designing and building graphical user interfaces. It allows you to design widgets, dialogs or complete main windows using on-screen forms and a simple drag-and-drop interface. It has the ability to preview your designs to ensure they work as you intended, and to allow you to prototype them with your users, before you have to write any code.



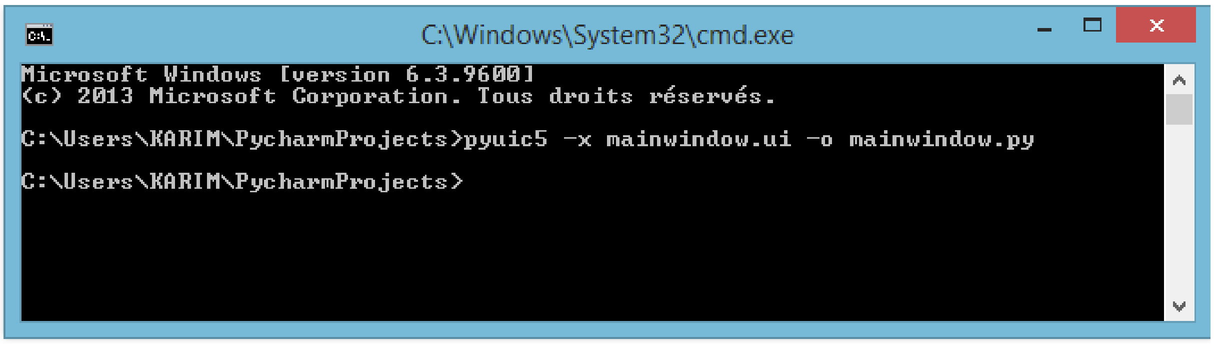
Object	Class
english	QWidget
Text	QTabWidget
tab	QWidget
checkBox_10	QCheckBox
checkBox_11	QCheckBox

english : QWidget

Property	Value
QObject	
objectName	english
QWidget	
enabled	✓
geometry	[(0, 0), 808 x 510]
X	0
Y	0

Figure 4. 3 PyQt designer

Qt Designer uses XML ".ui" files to store designs and does not generate any code itself. Qt includes the "uic" utility that generates the C++ code that creates the user interface. Like the "uic" utility it can also generate the Python code that will create the user interface. PyQt5's "pyuic5" utility is a command line interface to the "uic" module.



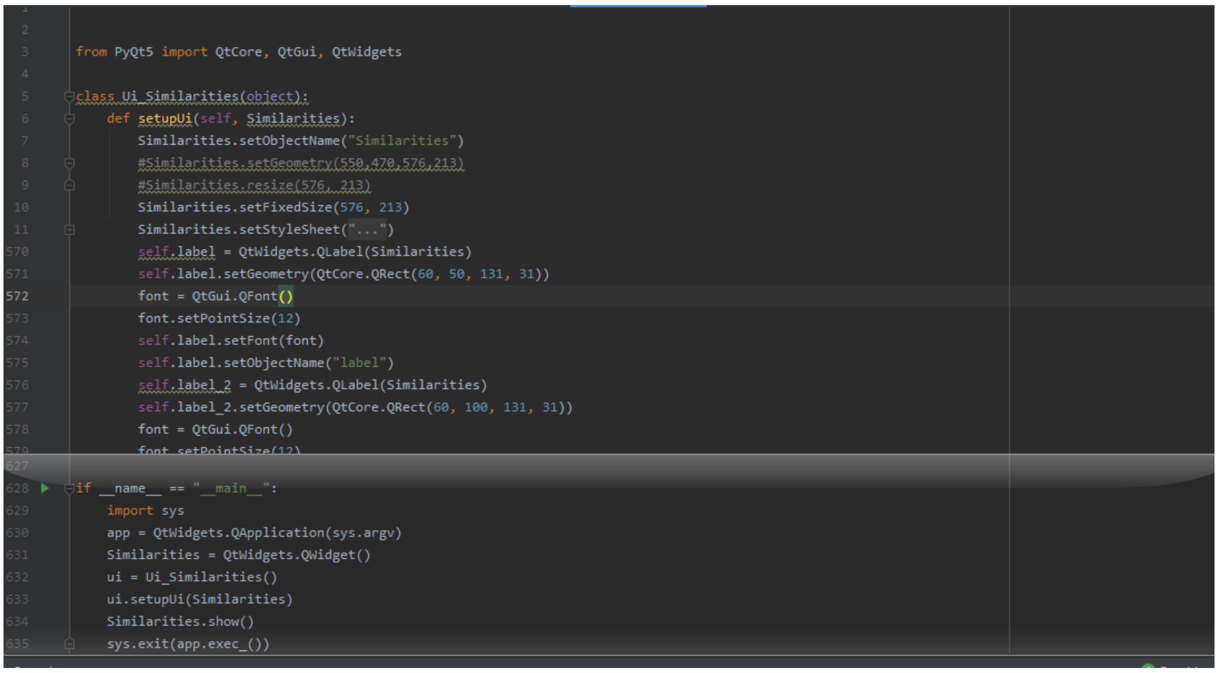
```
Microsoft Windows [version 6.3.9600]
(c) 2013 Microsoft Corporation. Tous droits réservés.

C:\Users\KARIM\PycharmProjects>pyuic5 -x mainwindow.ui -o mainwindow.py

C:\Users\KARIM\PycharmProjects>
```

Figure 4. 4 Generating python code from .ui

The code that is generated has an identical structure to that generated by Qt's "uic" and can be used in the same way.



```python

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Similarities(object):
    def setupUi(self, Similarities):
        Similarities.setObjectName("Similarities")
        #Similarities.setGeometry(550,470,576,213)
        #Similarities.resize(576, 213)
        Similarities.setFixedSize(576, 213)
        Similarities.setStyleSheet("...")
        self.label = QtWidgets.QLabel(Similarities)
        self.label.setGeometry(QtCore.QRect(60, 50, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(Similarities)
        self.label_2.setGeometry(QtCore.QRect(60, 100, 131, 31))
        font = QtGui.QFont()
        font.setPointSize(12)

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Similarities = QtWidgets.QWidget()
    ui = Ui_Similarities()
    ui.setupUi(Similarities)
    Similarities.show()
    sys.exit(app.exec_())
```

Figure 4. 5 Python code generated from interface design

4.4 Application Presentation:

Our application is a desktop application for comparison between two texts or two documents, it uses the Vector Space Models to calculate and compare the similarity. It's made up of few windows which we will describe.

4.4.1 Main window:

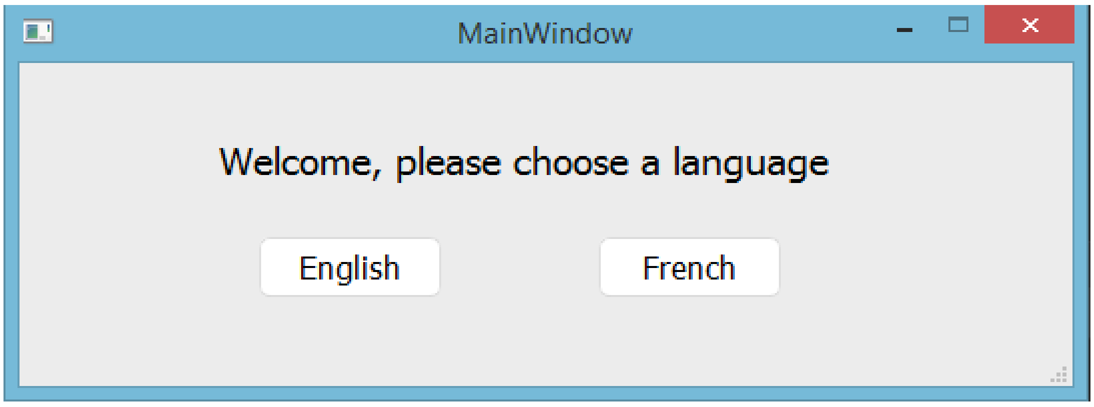
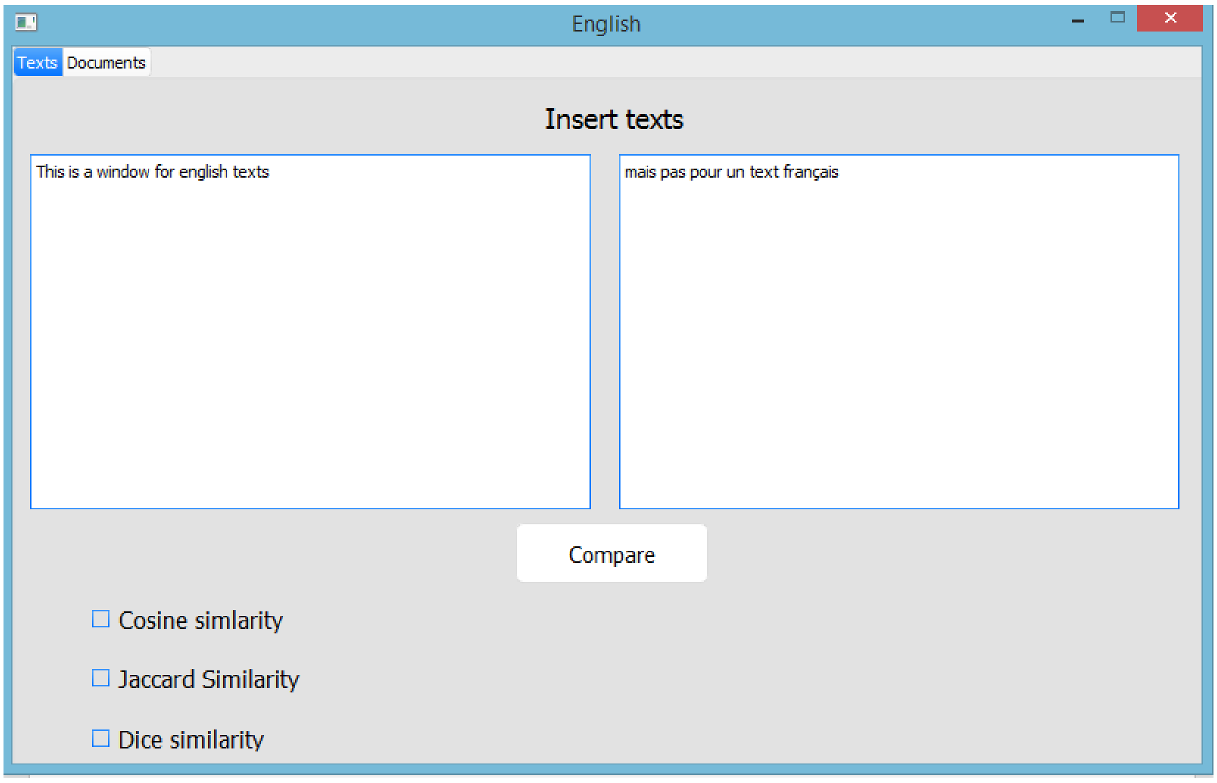


Figure 4. 6 Main Window of application

This window is the first one of our application, from it the user can choose the language of the next interfaces and also, the language of the texts he wants to compare or documents. If he clicks the English button, the next windows will be completely in English as well as for if he chooses French.

4.4.2 English window:



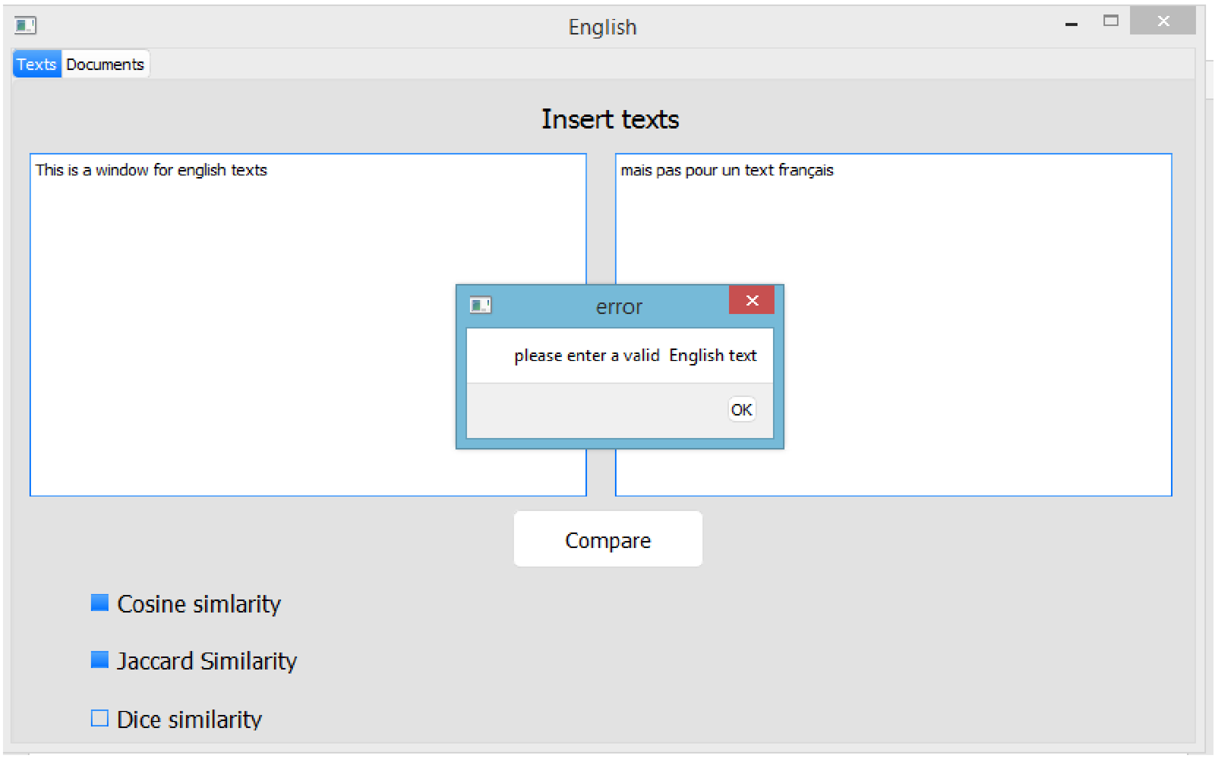
This is a window for english texts

mais pas pour un text français

Figure 4. 7 Texts tab from English Window of application

This is the Second window of our application, it’s divided into two tabs, the first one is for texts, and the second one is for uploading documents.

The first tab is consisting of two plain text edits which in them the user write or copy the texts he want to calculate the similarity between them, and as we said before, this window is only for English texts, so if he write a French one, and click compare, a message box will show to inform him to insert an English one as in the figure below



This is a window for english texts

mais pas pour un text français

please enter a valid English text

Figure 4. 8 Error message from English window

Also, from this window, the user can choose which type of similarity he want to use, either a Cosine, Jaccard or Dice similarity or all of them, but he has to choose at least one, if he didn’t another message box will appear to tell him that he must choose at least one similarity as in the figure below

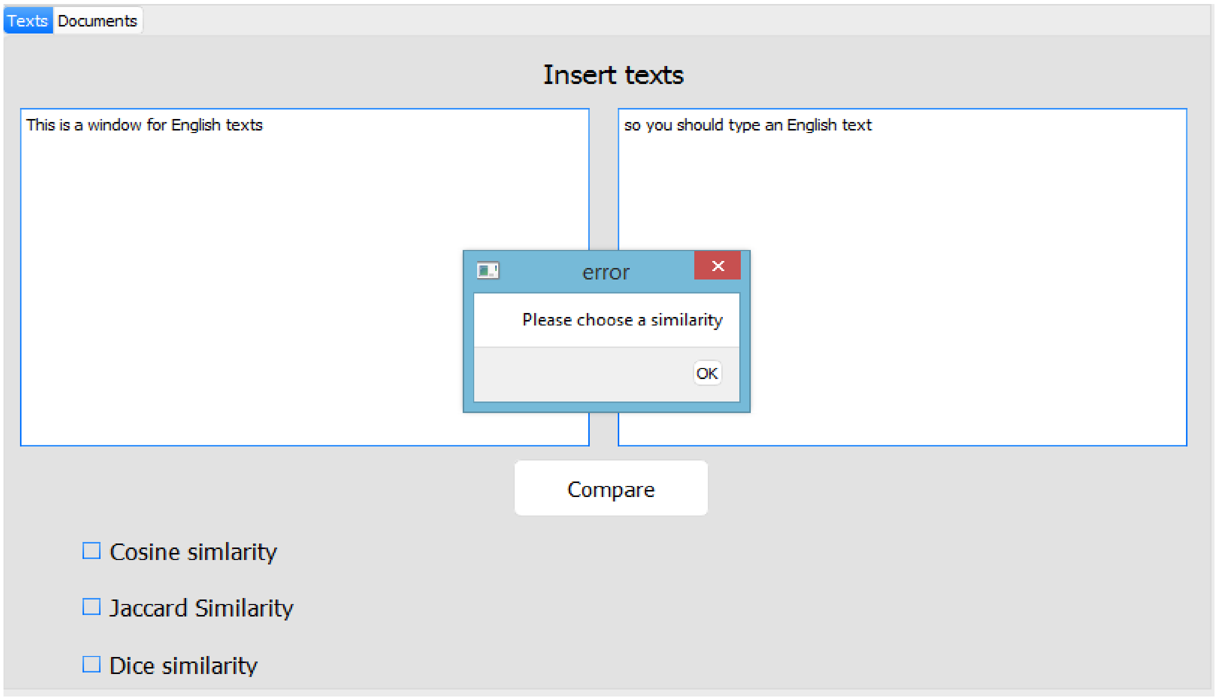

Texts
Documents
Insert texts
This is a window for English texts
so you should type an English text
error
Please choose a similarity
OK
Compare
Cosine simlarity
Jaccard Similarity
Dice similarity

Figure 4. 9 Error message from English window

So, the user has to enter English texts, and choose one similarity option or more before he pushes the button compare. If all the requirements are satisfied, a push up window will appear to show the results of the similarities he chooses.

The Second tab is documents,

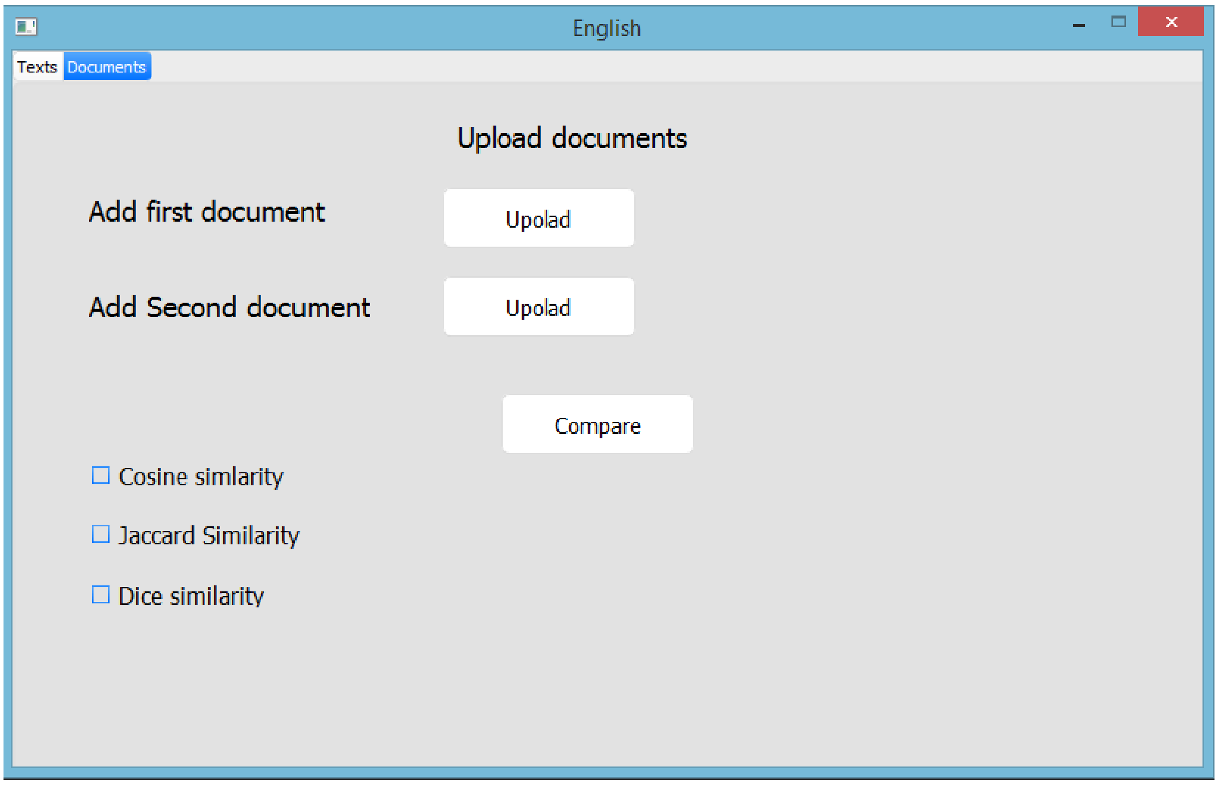


Figure 4. 10 Documents tab from English window

This window is consisting of two buttons to upload documents with ".docx" format or ".txt" format. When the user pushes the upload button a dialog window pops up and tell him to choose a file, and when he chooses one, the name of this file is going to appear next to the button he pushes as in the figure below

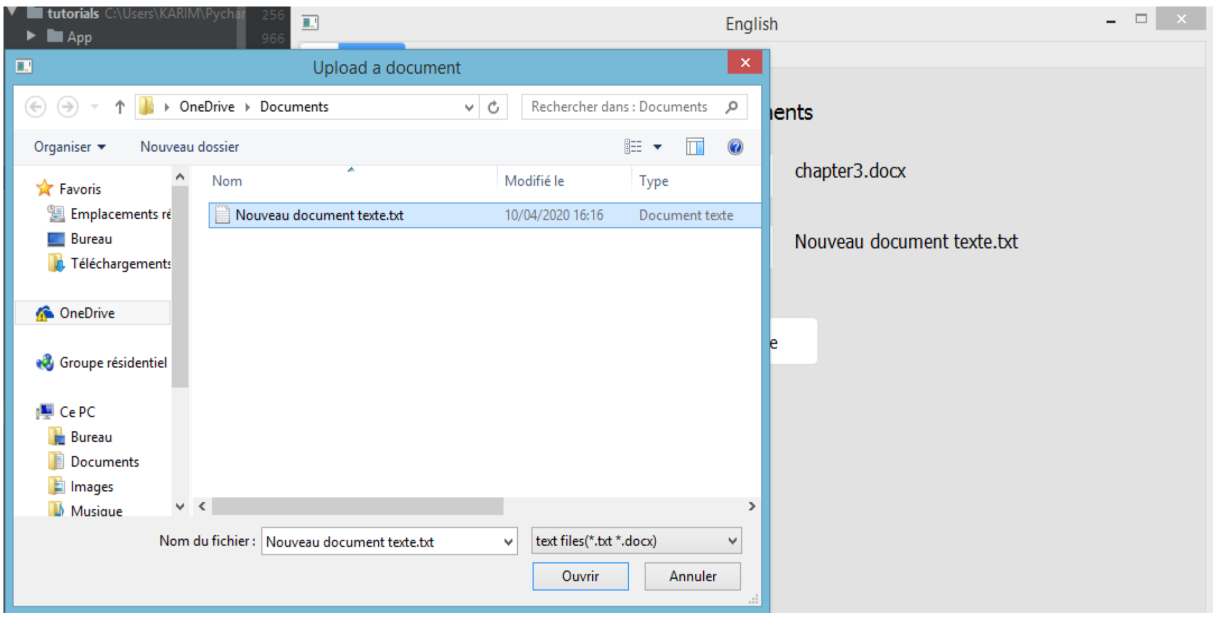
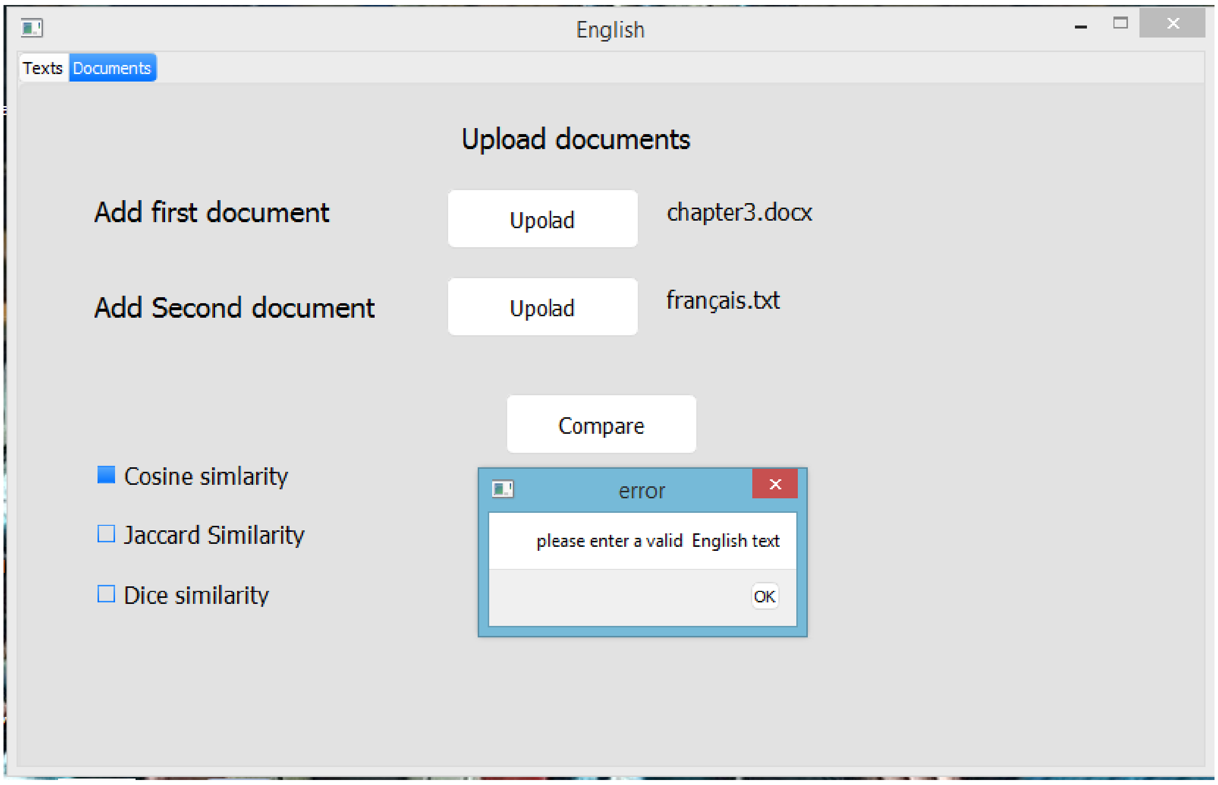


Figure 4. 11 Dialog window to choose documents

Those documents must be both in English, or a message box will appear to tell him to choose a valid English document



please enter a valid English text

Figure 4. 12 Error message from Documents tab

Then the user must at least check one checkbox to choose a similarity or more than one to compare between documents

4.4.3 French window:

This window is almost like the English one, when the user clicks the French button from the main window, it appears completely in French and the documents or texts mush be in French too.

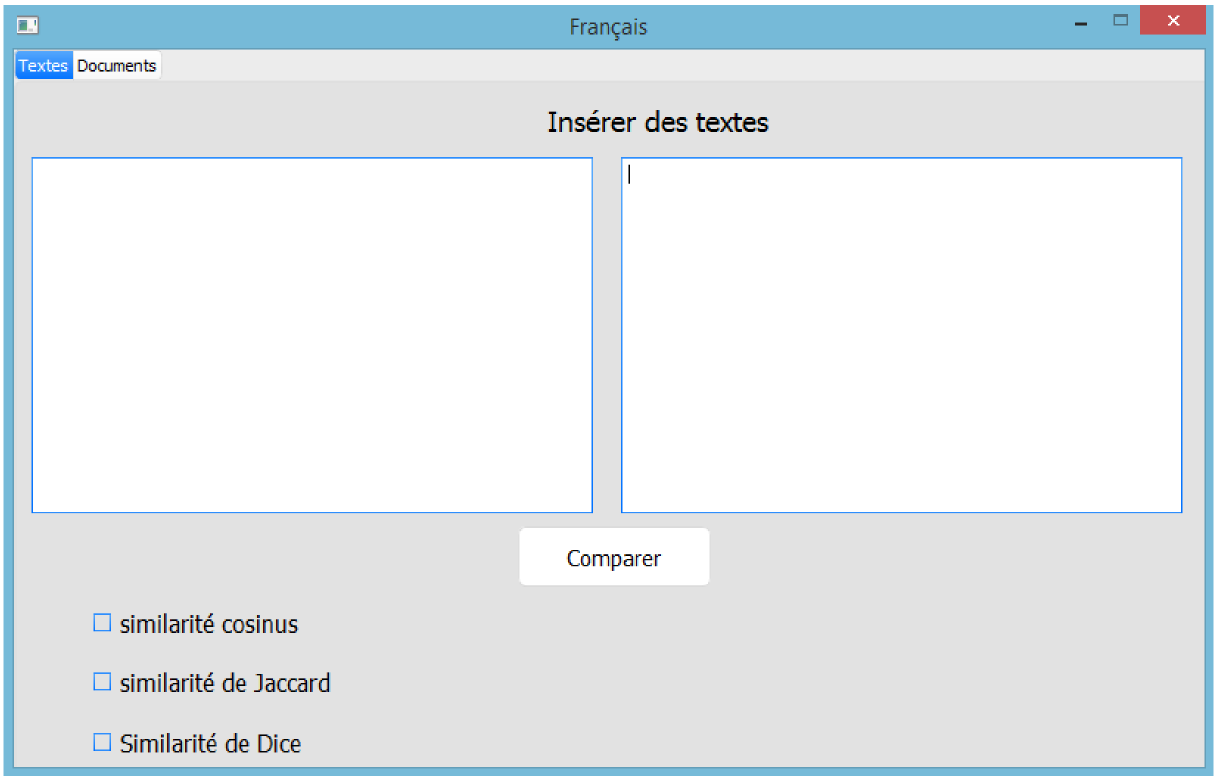


Figure 4. 13 Texts tab from French window

The first tab consists of two plain text entries where the user writes or copy only French texts, otherwise, he got an error to re-enter a French one, also, from this tab the user must choose at least one similarity type to calculate,

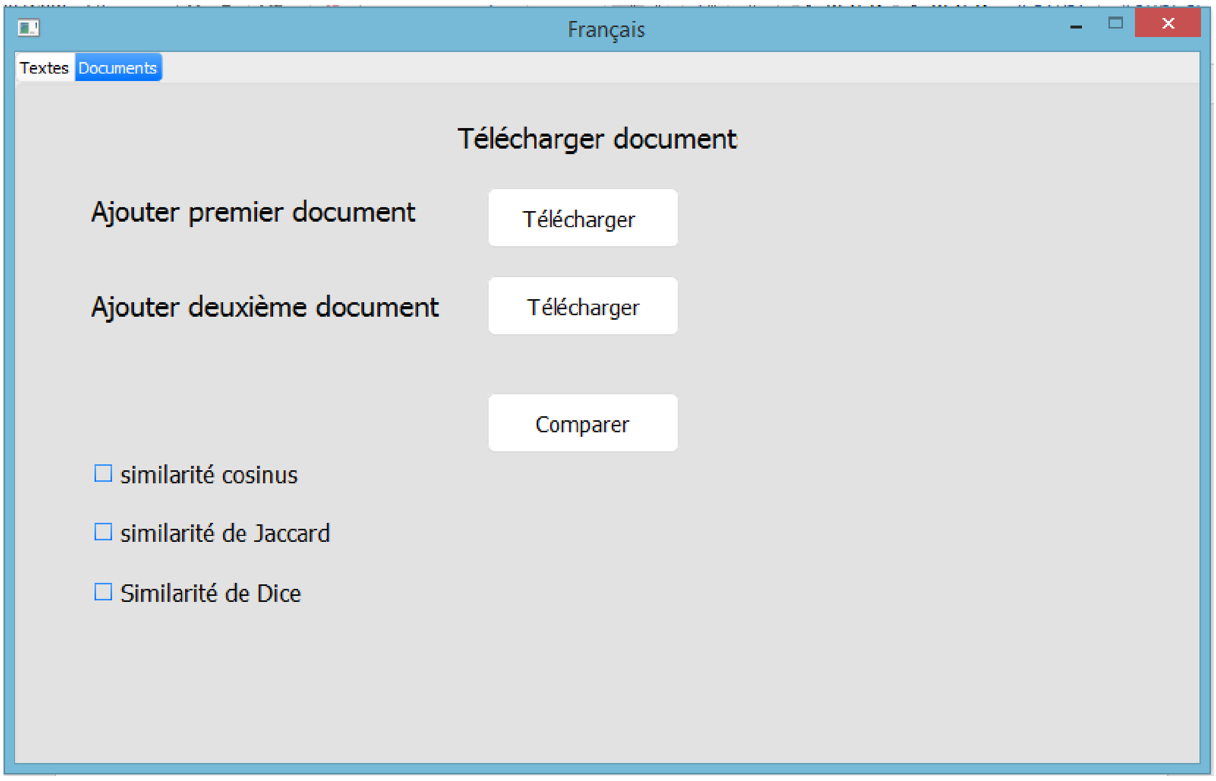
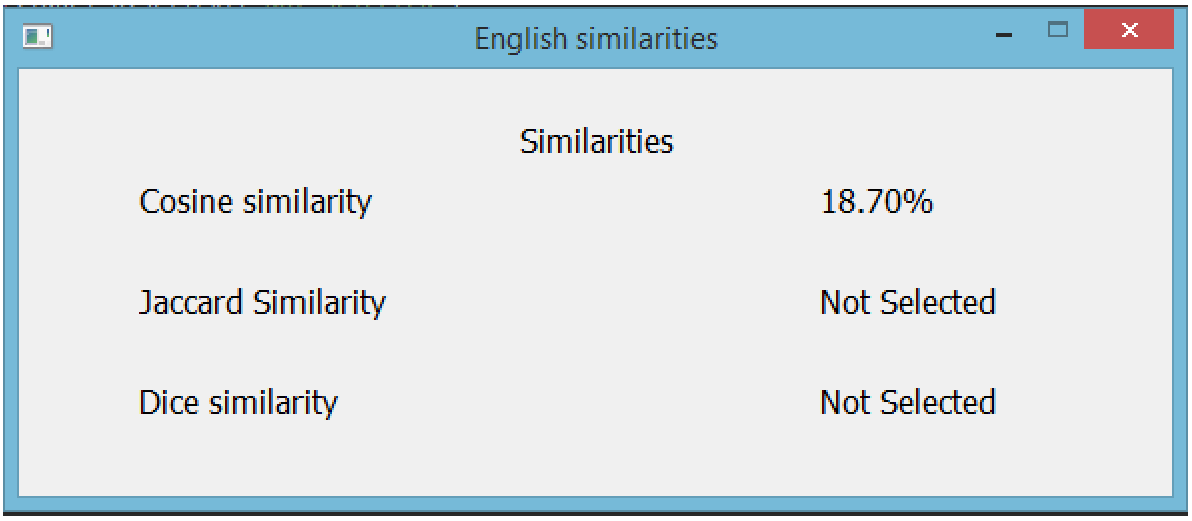


Figure 4. 14 Documents tab from French window

From this window the user can choose files or documents to upload, those documents must also be in French, and then he has to choose a similarity to calculate

4.5 Result discussion:



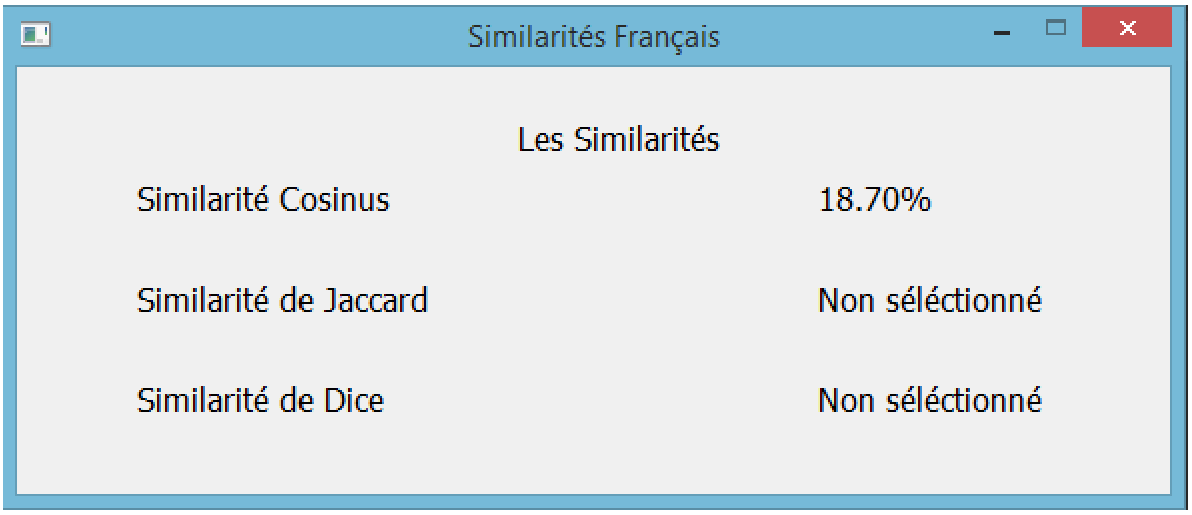
English similarities

Similarities

Cosine similarity	18.70%
Jaccard Similarity	Not Selected
Dice similarity	Not Selected

Figure 4. 15 English similarity window

This window shows the results of the similarities that user selected from the previous English window, if he didn't choose one of them, the window returns "Not selected". A same window will appear if the user chooses to work with French documents as in the figure below



Similarités Français

Les Similarités

Similarité Cosinus	18.70%
Similarité de Jaccard	Non séléctionné
Similarité de Dice	Non séléctionné

Figure 4. 16 French similarity window

4.6 Conclusion:

In this chapter, we describe and present the programming language and tools we have used to implement our model, then we see the application of the model by showing the deferent interfaces and windows.

GENERAL CONCLUSION

In this paper, we have developed and implemented an application for comparing and finding the similarity between texts or documents. This paper is significant in many terms, we have presented the main features of Vector Space Model and we have implemented these features.

This work can the backbone of successful text mining operations such as searching and information retrieval (IR), text classification, information extraction (IE), document clustering, sentiment analysis, machine translation, text summarization, and natural language processing (NLP). This work also can be a great help for detecting plagiarism.

We will work in the future to improve the tool in order to enlarge its features, such as covering multiple file formats like pdf, html and other file formats. Also, we try to implement this application to work in multiple platforms as Linux, macOS, and phone operating systems. Moreover, we will try to make this work supports and covers other Languages like Arabic Language and many others. Also, we will try to develop this application in order to work with it online.

We are glad to build such tool and we do appreciate all the readers of this report to try it out and helps us in order to enhance the current tool.

REFERENCES

[1] A. Kulkarni, C. More, M. Kulkarni and V. Bhandeka, Text Analytic Tools for Semantic Similarity, vol. 2, Imp. J. Interdiscip. Res, 2016.

[2] D. D. PRASETYA, A. PRASETYA WIBAWA et T. HIRASHIMA, The performance of text similarity algorithms, vol. 4, International Journal of Advances in Intelligent Informatics, 2018, pp. 63-69.

[3] Mihalcea R, Corley, C. et Strapparava, Corpus based and knowledge-based measures of text semantic similarity, Boston, MA: In Proceedings of the American Association for Artificial Intelligence, 2006.

[4] Miller, G.A., Beckwith, R., Fellbaum, C.D. , Gross,D et Miller, K, WordNet: An online lexical database, vol. 3, 1990, pp. 235-244.

[5] Fahmy, et W. H. Gomaa and A. A., A survey of text similarity approaches, vol. 68, Int. J. Comput. Appl, 2013.

[6] J. Wang, G. Li et J. Fe, “Fast-join: An efficient method for fuzzy token matching based string similarity join, IEEE 27th International Conference on Data Engineering, 2011, p. 458–469.

[7] M. Yu, G. Li, D. Deng et J. Feng, String similarity search and join: a survey, vol. 10, Front. Comput. Sci, 2016, p. 399–417.

[8] M. Y. Bilenko, Learnable similarity functions and their application to record linkage and clustering, 2006.

[9] R. Ali and M. M. S. Beg, Modified rough set based aggregation for effective evaluation of web search systems, Cincinnati, OH: Annual Meeting of the North American Fuzzy Information Processing Society, 2009.

[10] Ciesielski K., Wierzchoń S.T. et Kłopotek M.A., An Immune Network for Contextual Text Data Clustering, vol. 4163, Springer, Berlin, 2006.

[11] L. M., Thesaurus-Based Automatic Indexing, Handbook of Research on Text and Web Mining Technologies, 2009, pp. 331-345.

[12] Jiawei Han, Micheline Kamber et Jian Pei, Data Mining: Concepts and Techniques, 2012.

[13] Bank, J. et Cole, B, Calculating the jaccard similarity coefficient with map reduce for entity pairs in wikipedia, 2008.

[14] P. Jaccard, étude Comparative de la distribuition florale dans une portion des Alpes et des Jura, vol. 7, Bulletin de la Société Vaudoise des Sciences Naturelles, 1901, pp. 547-579.

[15] L. Dice, Measures of amount of ecologic association between species, vol. 26, Ecology, 1945, p. 297–302.

[16] Salton, G and McGill, M.J, Introduction to Modern Information Retrieval, New York: McGraw-Hill Book Co, 1983.

[17] Gerard Salton and Christopher Buckley, Term-weighting approaches in automatic text retrieval, vol. 24, Information Processing & Management, 1988, pp. 513-523.

[18] Kenneth Church and William A. Gale, Inverse Document Frequency (IDF): A Measure of Deviations from Poisson, 3rd workshop ed., Very Large Corpora, 1995.

[19] D. Kuhlman, A Python Book Beginning Python, Advanced Python, and Python Exercises, 2015.

Abstract:

The study and comparison of documents has proven to be a very important task for the detection of plagiarism, the retrieval of new information as well as the categorization of documents

VSMs (Vector Space Models) are one of the most efficient models of the information retrieval (IR) system, These models allow to represent complex information in a relatively simplistic form, which makes it possible to apply vector computation for text analysis.

This project aims to develop a text comparison system based on VSMs which allows to indicate the correspondence rate (similarity) between two texts or given documents using the python language.

Key Words:

VSM- Vector Space Models- Document analysis- Plagiarism detection- Python

ملخص :

أثبتت دراسة الوثائق ومقارنتها أنها مهمة بالغة الأهمية للكشف عن السرقات الأدبية واسترجاع المعلومات الجديدة وكذلك تصنيف الوثائق.

تعد VSM(Vector Space Models) واحدة من أكثر نماذج نظام استرجاع المعلومات كفاءة. (IR) تسمح هذه النماذج بتمثيل المعلومات المعقدة في شكل مبسط نسبيًا ، مما يسمح بتطبيق حساب المتجه على تحليل النصوص.

يهدف هذا المشروع إلى تطوير نظام مقارنة نص يعتمد على VSMs التي يمكن أن تشير إلى معدل التطابق (التشابه) بين نصين أو مستندات معينة باستخدام لغة.python

كلمات مفتاحية :

Résumé :

L'étude et la comparaison des documents s'est montrée une tâche très importante pour la détection de plagiat, la récupération de nouvelles informations ainsi que la catégorisation des

documents

Les VSM(Vector Space Models) sont l'un des modèles les plus efficaces du système de recherche d'informations (IR) (information retrieval, Ces modèles permettent de représenter des informations complexes sous une forme relativement simpliste, ce qui permet d'appliquer le calcul vectoriel à l'analyse de textes.

Ce projet vise à développer un système de comparaison de textes basés sur les VSM et qui permet d'indiquer le taux de correspondance (similarité) entre deux textes ou documents donnés en utilisant la langages python.

Mots clés :

VSM -Vector Space Models- Analyse des documents – détection de plagiat- Python