PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
FERHAT ABBAS UNIVERSITY OF SETIF

FACULTY OF SCIENCES-DEPARTMENT OF COMPUTER SCIENCES

Ferhat Abbas University Setif 1

THESIS

Submitted to the Faculty of Sciences
Department of Computer Sciences
to obtain a Master's Degree **in**
Data Engineering and Web Technology
Networks and Distributed Systems

TITLE

Context-independent word embedding methods in Machine Learning for Arabic Document Classification.

Submitted By

BELBEDAR Mohamed Akram

DIFALLAH Racha

Supervisor

Dr. KARA MOHAMED Chafia.

Session 2022/2023

Table of Contents

List of Figures

List of Tables

Dedication

First I would love to give my special thanks to our supervisor Dr. KARA MOHAMED Chafia, wish her the best & To my parents may god bless them and friends for their support to accomplish this work, and to my beloved sister who was and will always be with me here to support me , thank you .

Belbedar Mohamed akram

For first I would like to give my special thanks to my family my two great brothers, specially my parents for their support through my education May god bless them, to our supervisor Dr.KARA MOHAMED Chafia for the efforts with us, to my best friend the best sister in whole world.

Difallah racha

Abstract

With the rapid development of Artificial Intelligence, the research on text information processing began to get researchers' attentions. A huge amount of textual data is available online; therefore, automatic text classification is more than necessary. In Natural language processing, in general, and in text classification precisely, the main issue is the curse of dimensionality. Documents are represented by huge and sparse vectors. Reduce this dimensionality without affecting the information amount in the document is an active research area.

Word embeddings is the representation of the text using vectors such that the words that have similar semantic will have similar vector representation. Models like FastText, Glove and the two approaches of word2vec model called CBOW and Skip-gram. In our project, we have studied word embedding methods. Then we have used word embeddings as a method to reduce the dimensionality of the documents. Therefore, similar words are clustered and the clusters' centres are used to represent the whole cluster. To highlight the effect of this method we have compared a text classification system with no dimension reduction and one with dimension reduction.

الملخص

مع التطور السريع للذكاء الاصطناعي، بدأ البحث في معالجة المعلومات النصية في جذب انتباه الباحثين. يتوفر قدر هائل من البيانات النصية عبر الإنترنت؛ لذلك، تصنيف النص التلقائي أصبح أكثر من ضروري.

في معالجة اللغة الطبيعية، بشكل عام، وفي تصنيف النص الطبيعي بدقة، فإن القضية الرئيسية هي ما يسمي بلعنة الأبعاد. يتم تمثيل المستندات بواسطة مصفوفات كبيرة ومتفرقة. يعد تقليل هذه الأبعاد دون التأثير على مقدار المعلومات في المستند مجال بحث نشط.

تضمين الكلمة هو تمثيل النص باستخدام المتجهات بحيث يكون للكلمات التي لها دلالات متشابهة تمثيل متجه مماثل. نماذج مثل FastText وGlove وطريقتين لنموذج word2vec تسمى CBOW و Skip-gram.

لقد درسنا في مشروعنا طرق تضمين الكلمات. ثم استخدمنا تضمين الكلمات كطريقة لتقليل أبعاد المستندات. لذلك ، يتم تجميع الكلمات المتشابهة ويتم استخدام مراكز المجموعات لتمثيل المجموعة بأكملها.

لتسليط الضوء على تأثير هذه الطريقة، قمنا بمقارنة نظام تصنيف النص مع عدم تقليل الأبعاد والآخر مع تقليل الأبعاد.

Keywords: word embeddings , dimension reduction , text classification, word2vec , Glove , semantic similarity , CBOW, Skip-gram , arab fast text .

Introduction

Nowadays, text classification is becoming more and more challenging due to the large amount of text documents available in the electronic forms. To solve this problem, we need effective automatic text classification techniques to save time and be able to organize extensive collection of text documents.

Text classification is the process of assigning a label to a document based on its content. This classification can be bi-class with two classes or multi-class if the number of possible classes is more than two. Before passing the textual data to the classifiers, a step of pre-processing is performed to prepare the text for the classification.

Classifiers get the features of the documents in numerical vectors. These vectors are characterised by their huge size and sparsity. This issue is called the curse of dimensionality. To reduce this dimension, different approaches are used. Feature selection where some pertinent features have to be selected and useless one are discarded. Feature extraction are methods where new features are extracted based on some transformations.

TF-IDF is the more used method to represent text for classification. Here, each feature is represented by its inverse document frequency. But, in this model, words with similar meaning have different representations.

To solve this problem, word embeddings is the representation of the text using vectors such that the words that have similar semantic will have similar vector representations. Models like FastText, Glove and the two approaches of word2vec model called CBOW and Skip-gram were developped.

In our project, we have used word embeddings as a tool for dimension reduction in Arabic Language.

Objectives of the work

-Study the word embedding representations

-Use word embeddings as a tool to reduce the dimension of Arabic textual documents

-Compare text classification with TF_IDF and word embeddings-based reduced dimension

Organization of the manuscript

Chapter 1: word embeddings systems are studied

Chapter 2: Text classification system is presented in details.

Chapter 3: Deals with the implementation and the results obtained

Conclusion: Here we conclude our work and set some future works to be continued

Chapter One Word embeddings

1- Introduction

In the last few years, the benefits of using distributed word representations (embeddings) have been illustrated and highlighted in many different NLP tasks including but not limited to, While these benefits were accompanied by the provision of several open source word representation models in English, the same cannot be said to be true with respect to Arabic.

The representation of words and documents is part and parcel of most, if not all, Natural Language Processing (NLP) tasks. In general, it has been found to be useful to represent them as vectors, and lend themselves well to be used in many Machine Learning (ML) algorithms and strategies.

In this chapter we will see the definition of word embedding with the three models suggested.

2- Definition

Word embeddings also known as word representation is one of the important hypothesis representations used to represent words, phrases and sentences to be used in several natural language processing (NLP) applications , is used to represent the words by low dimensional vectors representation, such that the syntax and semantic relationship between words can easily be measured. [1]

There are several models for generating word embeddings. In order to be useful, these models must be trained using very large corpus to determine the semantic relationship between words since the semantic similarity is important for several applications. Recently, four word embeddings models were proposed that played a significant role in variety of NLP applications called *word2vec* model, *Glove* model, *Aravec* model and Arabic *Fasttext* model.

3- Word Embeddings methods

In our graduation thesis, we study the word embeddings models like: *word2vec*, *Glove* and *Arabic fast text* models .

a) Word 2 Vec:

Word2vec model is a neural network that resolve the two problems of "one-hot" representation and consists of one input layer, one output layer and one hidden layer. Hidden layer has no activation functions.[1] *Word2vec* consists of two approaches including continuous bag-of-words model *(CBOW)* and *Skip-gram* models by (mkolov et al-2013) where both of these models are log-linear models and achieved improvements in term of accuracy and computational cost.
The two approaches use the same hyper parameters such as the window size and the vocabulary size. The window size represents the number of words in the context and denoted by c and the vocabulary size denoted by |v|. The two approaches of word2vec model which are *CBOW* and *Skip-gram* will be covered in the following subsections:

1-Continuous Bag-of-Words Approach (CBOW)

The Continuous bag-of-words(*CBOW*) model aims to learn the embeddings by predicting the center word in a context given the other words in the context without regard to their order in the sentence , uses log-linear classifier to classify the predicted middle word given the surrounding future and history words, *CBOW* maximizes equation (1) [1] where W(t) represents the current word while the context words are represented using the following symbols{ w(t-c) , ... ,w(t-2) , w(t-1) , w(t+1) , w(t+2) , ... , w(t+c) }.

$$\frac{1}{|V|}\sum_{t=1}^{|V|} \log\left[p\left(wt \middle| \begin{matrix} wt-c\,, \dots\,, wt-2\,, wt-1\,, \\ wt+1\,, wt+2\,, \dots\,\,, wt+c \end{matrix}\right)\right] \qquad (1)$$

The size of the sliding window determines the number of the words in the context, such that if the size of the sliding window is five then the number of the context words is four and the value of c will be equal to four. Moreover, in order to predict a word, the preceding two words and the following two words, of the middle word to be predicted, must be considered in the context.

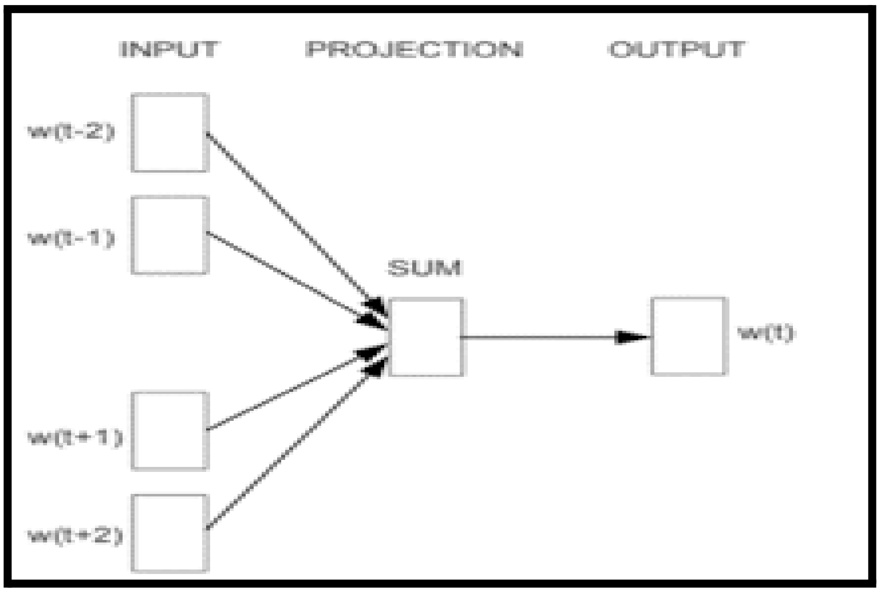


Figure 1:CBOW model architecture.

2-Continuous Skip-gram Approach (Skip-gram)

CBOW and *Skip-gram* approaches of *word2vec* model are very similar in the structure. However, there is only one difference between the two approaches. While in *CBOW* the input for neural network are the context words and the output is the middle word, in *Skip-gram* model the input is the current word (middle word) and the output are the context words ,The *Skip-Gram* model is the reverse of the *CBOW.* The context words are represented using the following symbols { w(t-c) , ... ,w(t-2) , w(t-1) , w(t+1) , w(t+2) , ... , w(t+c) }, while w(t) represents the current word.
As in *CBOW* the objective of the model is to maximize the log of the probability in equation (2)[1] .

$$\frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{j=t-c,j<>t}^{t+c}\log[p(wj|wt)] \qquad (2)$$

The sliding window is used to predict the next word.

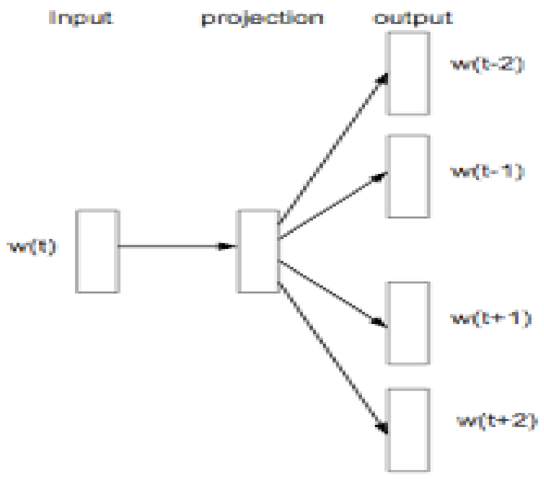


Figure 2: Skip-Gram model architecture.

For conclusion, in both models the input for the neural network will be the one-hot representation of the words.

There are two evaluation methods in the *Word2Vec* model, namely Hierarchical Soft max and Negative Sampling.

- *Hierarchical Soft max* was first introduced Morin and Bengio . Hierarchical Soft max uses binary tree representations of the output layer with words as leaves and each node, explicitly representing the relative probabilities of their child nodes.

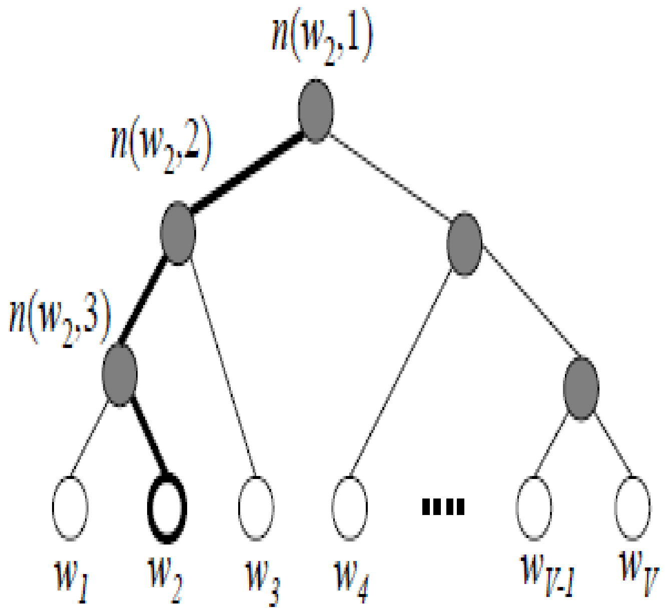
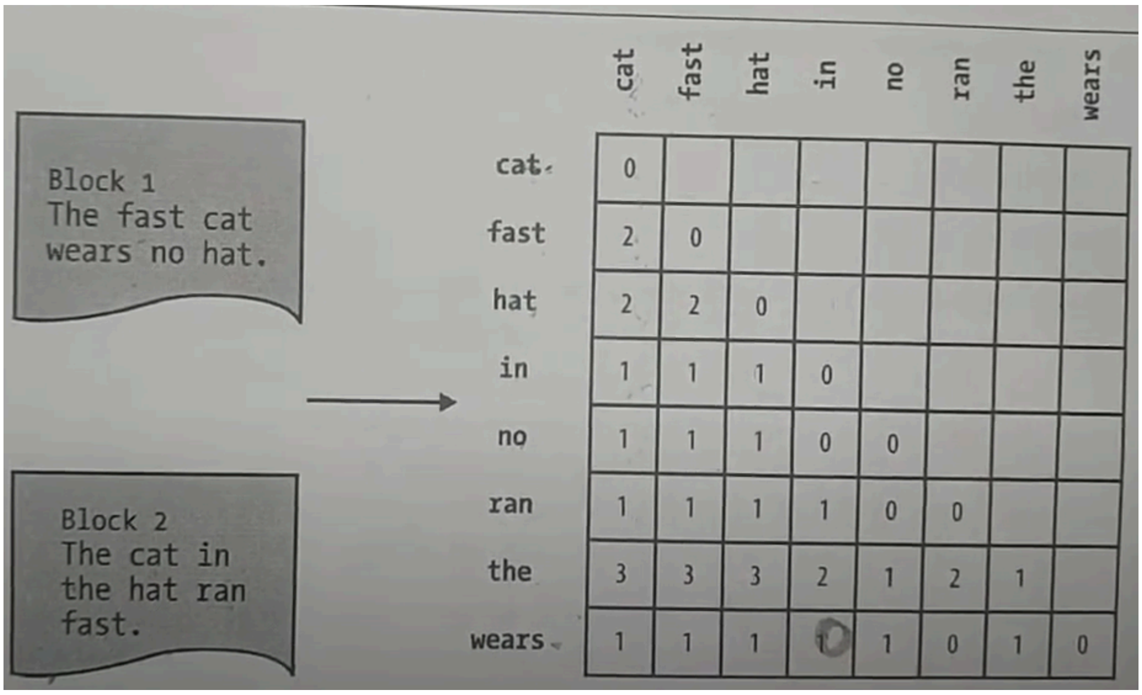


Figure 3: Example binary tree for the hierarchical softmax model**.**

b) Vectors for word representation (Glove) method

The unsupervised word embedding model for word representation, which is introduced by Pennington et al. is known as *GloVe.*
GloVe word embeddings stands for Global vector for word representation since it captures the statistics of the global corpus directly from the model, instead of depending on local context windows like *word2vec* model , Glove uses the statistics efficiently by training the model on the global count of word-to-word co-occurrence.
The basic idea behind the *GloVe* word embedding is to derive the relationship between the words from statistics. Unlike the occurrence matrix, the co-occurrence matrix tells you how often a particular word pair occurs together. Each value in the co-occurrence matrix represents a pair of words occurring together. So,word embeddings is a global log-bilinear regression model and is based on co-occurrence and factorization of matrix in order to get vectors

Example for *The co-occurrence matrix.*



Block 1
The fast cat
wears no hat.

Block 2
The cat in
the hat ran
fast.

	cat	fast	hat	in	no	ran	the	wears
cat	0							
fast	2	0						
hat	2	2	0					
in	1	1	1	0				
no	1	1	1	0	0			
ran	1	1	1	1	0	0		
the	3	3	3	2	1	2	1	
wears	1	1	1	1	1	0	1	0

Figure 4: The co-occurrence matrix[4].

Now when you compute the ratio of probabilities between two pairs of words say you have probabilities of (cat/fast) = 1 and (cat/the)=0.5, when you compute the ratio of these, the result is 2 it says 'fast' is more relevant than 'the' . This is the idea behind the *GloVe* pre-trained word embeddings, and it is expressed in this equation[5]:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \qquad (3)$$

Where Wi refers to the word vector of word i , and Pik denotes to the probability of word k to occur in the context of word i.

The vectors for each word is assigned randomly. Then we take two pairs of vectors and see how close they are to each other in space. If they occur together more often or have a higher value in the co-occurrence matrix and are far apart in space then they are brought close to each other.

If they are close to each other but are rarely or not frequently used together then they are moved further apart in space.

After many iterations of the above process, we'll get a vector space representation that approximates the information from the co-occurrence matrix. The performance of *GloVe* is better than Word2Vec in terms of both semantic and syntactic capturing.
The *GloVe* model aims to study the vector of words in such a way that the dot product of those words is equal to the logarithm of the probability of words to appear together or the probability of their co-occurrence .this Model can be written down as follows[6] :

$$w_i^T + \vec{w}_k + b_i + \vec{b}_k = log(X_{i_k}) \qquad (4)$$

Where w is the word vector, vector W is the context word vector, bi and bk are scalar biases for the i-word.

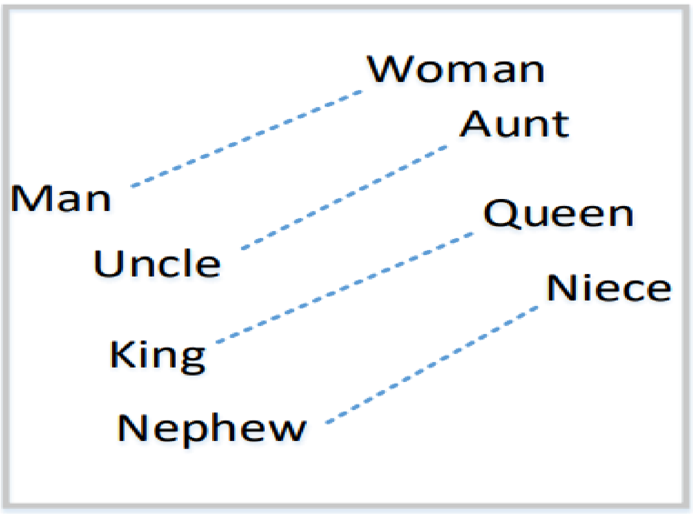


Figure 5: Global Vectors for Word Representation.

c) Arabic fast-text Method

FastText can be defined as a word embedding method which is part of the word2vec development, this model managed to show that their model was able to train at 1 billion words within 10 minutes, within performing quality of results compared to other models.

FastText model architecture is similar to the *CBOW* architecture in *Word2Vec* , it has a hierarchical structure and represents words in dense vector form. In addition, there is a hidden layer between the input layer and the output layer, as shown in the following figure:

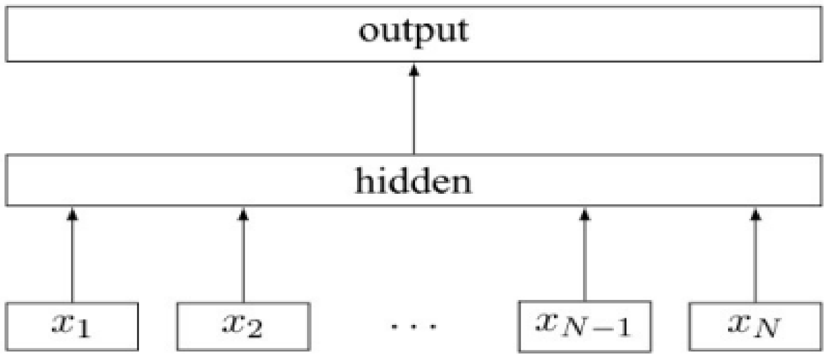


Figure 6 : Model architecture of fastText.[6]

FastText Instead of learning vectors for words directly, which can represents each word as an n-gram of characters, for example, take the word, "artificial" with n=3 we choose the gram that we need , the *FastText* representation of this word is <ar, art, rti, tif, ifi, fic, ici, ial, al>, where the angular brackets indicate the beginning and end of the word.
This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. Once the word has been represented using character n-grams, a skip-gram model is trained to learn the embeddings. This model is considered to be a bag of words model with a sliding window over a word because no internal structure of the word is taken into account. As long as the characters are within this window, the order of the n-grams doesn't matter.
FastText works well with rare words. So even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings.
The authors of *FastText* proposed different scoring functions to preserve the subword information. Given the word w, the set of n-grams appearing in w

will be Nw ⊂ {1, ..., N}, Where N is the dictionary size of n-grams. Vector representation Zg is assigned for each n-gram n. the function[7] becomes:

$$s(w, c) = \sum_{n \in N_w} Z_g^T V_c \qquad (5)$$

With : c = context word , Vc = context vector

In the end, *FastText* is an excellent performer, capable of rapidly training models on huge datasets and providing representations for words that do not exist in the training data. When a word does not occur during model training, its vector embedding may be determined by breaking it down into n-grams.

Chapter two Text Classification

1- Introduction

In recent years, there has been an exponential growth in the number of complex documents and texts that require a deeper understanding of machine learning methods to be able to accurately classify texts in many applications. Many machine learning approaches have achieved surpassing results in natural language processing. The success of these learning algorithms relies on their capacity to understand complex models and non-linear relationships within data. However, finding suitable structures, architectures, and techniques for text classification is a challenge for researchers. In this chapter, we will see the text classification meaning, the Arabic language in text classification brief overview of text classification and the dimensionality reduction methods.

2- Definition

Text classification, also called text categorization is a classical problem in natural language processing (NLP)[8] , is the process of automating a set of documents into specific groups based on the content of the text itself through the use of certain technologies and algorithms. It's also defined by Elhassan and ahmed like a method of searching for data and exploring them among large data and classifying them into groups for easy reference.[8]others mentioned that there was a slight difference between text categorization and classification. the categorization of texts entailed sorting them according to their content, their classification placed them in certain groups that suited their content according to author, subject, title, language, and other classifications. Specialized research on the classification of texts has increased significantly due to the enormous data available from many sources, including Internet pages, e-mail messages, and news pages, texts circulated through social media, reports, and journal articles. Therefore, this research focused attention in order to make the best possible use of all these data and classify them [8].TC tasks include sentiment analysis, news categorization and topic classification.

a) Sentiment Analysis

This is the task of analyzing people's opinions in textual data, and extracting their polarity and viewpoint. The task can be cast as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities.

b) News Categorization

News contents are among the most important information sources. A news classification system helps users obtain information of interest in real-time by e.g., identifying emerging news topics or recommending relevant news based on user interests.

c) Topic Analysis

The task, also known as topic classification, aims to identify the theme or topics of a text.

The most text classification and document categorization systems can be deconstructed into the following four phases: Feature extraction, dimension reductions, classifier selection, and evaluations. us it showing in the figure 6.

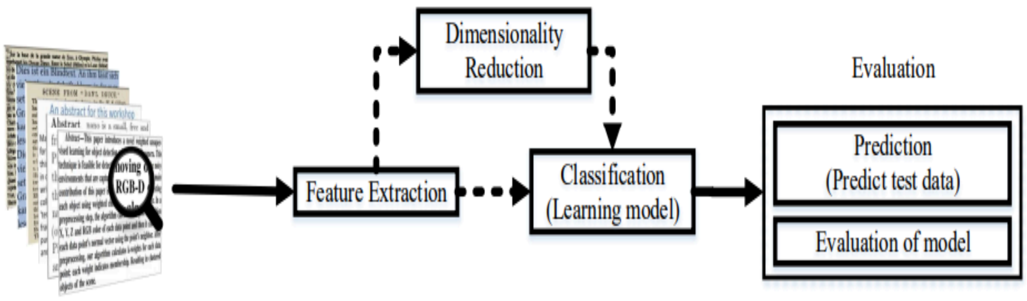


Figure 7: Overview of text classification pipeline.[9]

The initial pipeline input consists of some raw text data set. In general, text data sets contain sequences of text in documents as D={X1, X2, . . ., XN}

where Xi refers to a data point, with s number of sentences such that each sentence includes Ws words with lw letters. Each point is labeled with a class value from a set of k different discrete value indices.[10]
Then, we should create a structured set for our training purposes which call this section Feature Extraction. The dimensionality reduction step is an optional part of the pipeline which could be part of the classification system. The most significant step in document categorization is choosing the best classification algorithm. The other part of the pipeline is the evaluation step.

1. Feature Extraction

In general, texts and documents are unstructured data sets. However, these unstructured text sequences must be converted into a structured feature space when using mathematical modeling as part of a classifier.

First, the data needs to be cleaned to omit unnecessary characters and words, because most text and document data sets contain many unnecessary words such as stop words, misspelling, slang, etc. This called the preprocessing text it's a crucial steps for text classification applications, we will introduce methods for cleaning text data set for informative featurization .

1.1Tokenization

Tokenization is a pre-processing method which breaks a stream of text into words, phrases, symbols, or other meaningful elements called tokens [11]. The main goal of this step is the investigation of the words in a sentence[12]. Both text classification and text mining require a parser which processes the tokenization of the documents, for example:

After sleeping for four hours, he decided to sleep for another four.

In this case, the tokens are as follows :

{ "After" "sleeping" "for" "four" "hours" "he" "decided" "to" "sleep" "for" "another" "four" }.

1.2 Stop Words

Stop words are words that are particularly common in a text corpus and thus considered as rather un-informative (e.g., words such as so, and, or, the, ..."). One approach to stop word removal is to search against a language-specific stop word dictionary. An alternative approach is to create a stop list by sorting all words in the entire text corpus by frequency. The stop list — after conversion into a set of non-redundant words — is then used to remove all those words from the input documents that are ranked among the top n words in this stop list[13].

for example:

.
A swimmer likes swimming, thus he swims.

In this case, the will be as follows:

{ “swimmer” “likes” “swimming” “,” “swims” }.

1.3 Capitalization

Text and document data points have a diversity of capitalization to form a sentence. Since documents consist of many sentences, diverse capitalization can be hugely problematic when classifying large documents. The most common approach for dealing with inconsistent capitalization is to reduce every letter to lower case. This technique projects all words in text and document into the same feature space, but it causes a significant problem for the interpretation of some words .Slang and abbreviation converters can help account for these exceptions.

1.4 Slang and Abbreviation

Slang and abbreviation are other forms of text anomalies that are handled in the pre-processing step. An abbreviation[14] is shortened forms of a word or phrase which contain mostly first letters from the words, such as SVM which stands for Support Vector Machine. Slang is a subset of the language used in informal talk or text that has different meanings such as “lost the plot”, which essentially means that they’ve gone mad .A common method for dealing with these words is converting them into formal language[15].

1.5 Noise Removal

Most of the text and document data sets contain many unnecessary characters such as punctuation and special characters. Critical punctuation and special characters are important for human understanding of documents, but it can be detrimental for classification algorithms.

1.6 Spelling Correction

Spelling correction is an optional pre-processing step. Typos (short for typographical errors) are commonly present in texts and documents, especially in social media text data. Many algorithms, techniques, and methods have addressed this problem in NLP (Natural Language Processing).

Many techniques and methods are available for researchers including hashing-based and context-sensitive spelling correction techniques , as well as spelling correction using Trie and Damerau–Levenshtein distance bigram.

1.7 Steaming and lemmatization

Stemming describes the process of transforming a word into its root form. The original stemming algorithm was developed my Martin F. Porter in 1979 and is hence known as Porter stemmer [13]
, for example :

A swimmer likes swimming, thus he swims.

The result is ↓:

{ “ a ” “swimmer” “ like” “swim” “ ,” “thu” “he” “swim ” “.” }

Stemming can create non-real words, such as ”thu” in the example above.

In contrast to stemming, lemmatization aims to obtain the canonical (grammatically correct) forms of the words, the so-called lemmas. Lemmatization is computationally more difficult and expensive than stemming, and in practice, both stemming and lemmatization have little impact on the performance of text classification [16].

for example :

A swimmer likes swimming, thus he swims.

The result is ↓:

{ “ a ” “swimmer” “ like” “swimming” “ ,” “thus” “he” “swim ” “.” }

1.8 the N-Gram

The n-gram technique is a set of n-word which occurs “in that order” in a text set. This is not a representation of a text, but it could be used as a feature to represent a text. BOW is a representation of a text using its words (1-gram) which loses their order .This model is very easy to obtain and the text can be represented through a vector, generally of a manageable size of the text. On the other hand, n-gram is a feature of BOW for a representation of a text using 1-gram. It is very common to use 2-gram and 3-gram. In this way, the text feature extracted could detect more information in comparison to 1-gram[17].
An example of 2-Gram:

After sleeping for four hours, he decided to sleep for another four.

In this case, the results as follows:

{ “After sleeping”, “sleeping for”, “for four”, “four hours”, “four he” “he decided”, “decided to”, “to sleep”, “sleep for”, “for another”, “another four” }.

An example of 3-Gram:

After sleeping for four hours, he decided to sleep for another four.

In this case, the results as follows:

{ “After sleeping for”, “sleeping for four”, “four hours he”, “ hours he decided”, “he decided to”, “to sleep for”, “sleep for another”, “for another four” }.

2-Dimensionality Reduction

The text or document data sets often contain many unique words, data pre-processing steps can be lagged by high time and memory complexity. In order to avoid the decrease in performance, many researchers prefer to use dimensionality reduction to reduce the time and memory complexity for their applications. Using dimensionality reduction for pre-processing could be more efficient than developing inexpensive classifiers. Dimensionality reduction methods help us create neater and cleaner models that are free of noise and unnecessary features.

3-Classifier

A classifier is an algorithm that produces class labels as output, from a set of features of an object. It can automatically categories data into one or more of a set of classes. text classifiers can be used to organize, structure, and categorize pretty much any kind of text.

These are the four top classifier types we will be using in our study:

3.1) Random Forest (RF)

Random forest is another versatile supervised machine learning technique that may be used for classification and regression. The term ”forest” refers to a set of uncorrelated decision trees that are subsequently blended to minimize variation and produce more accurate data predictions.[18]

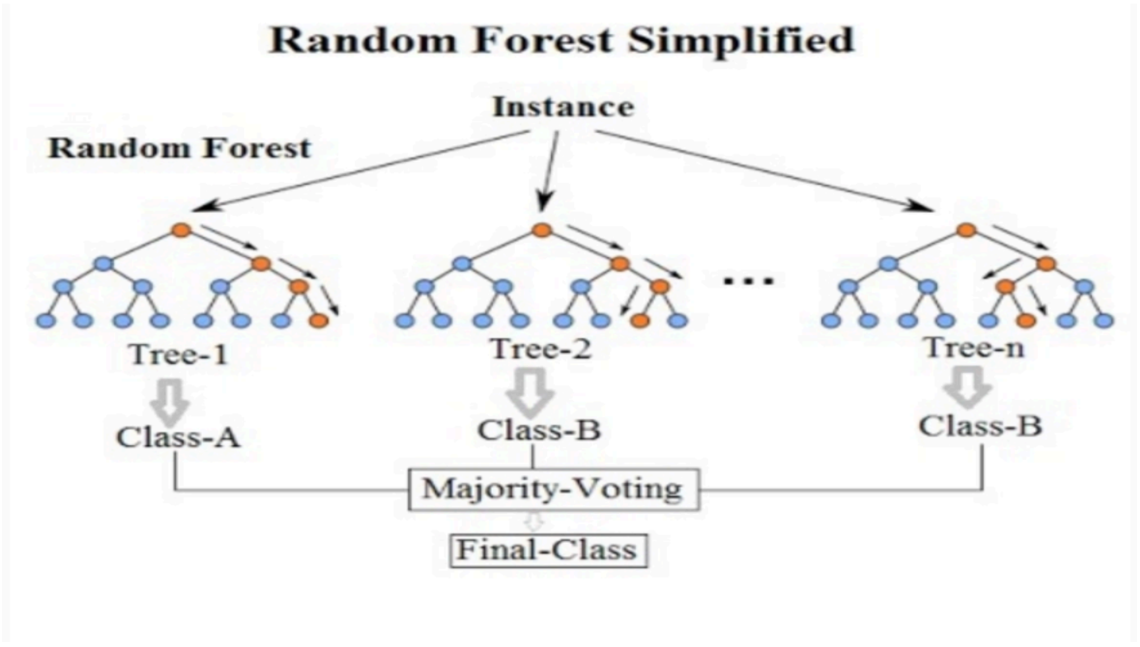


Figure 8: Random Forest[18].

3.2) Support Vector Machines (SVM)

A support vector machine (SVM) is a common supervised learning model that is used for classification as well as regression problems. However, it is often used for classification problems, creating a hyperplane with the greatest distance between two classes of data points. This hyperplane is known as the decision boundary, and it separates the data point classes.

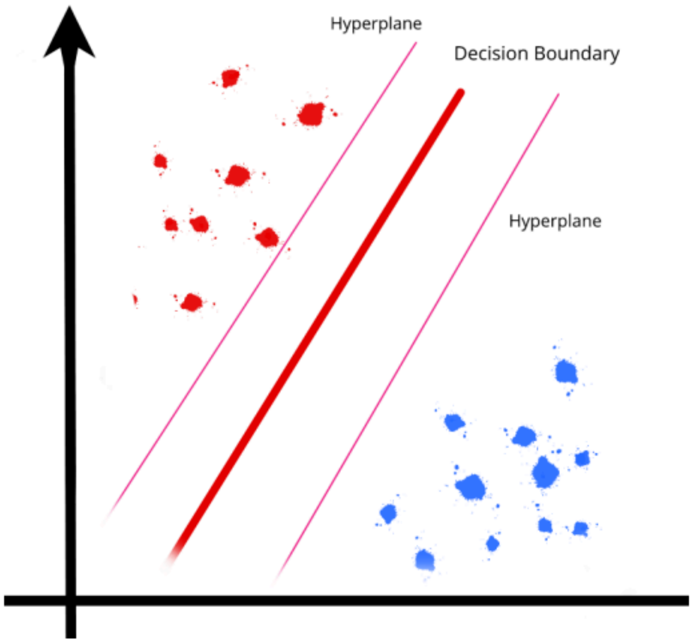


Figure 9: Support Vector Machines[19].

3.3) K-Nearest Neighbors (KNN)

KNN classifier determines the neighbors of an input document. The predicted class label is collectively computed by all determined neighbors, where each one votes for the closest label. The class-label with the maximum ballots is adopted. This is another major vote classifier.

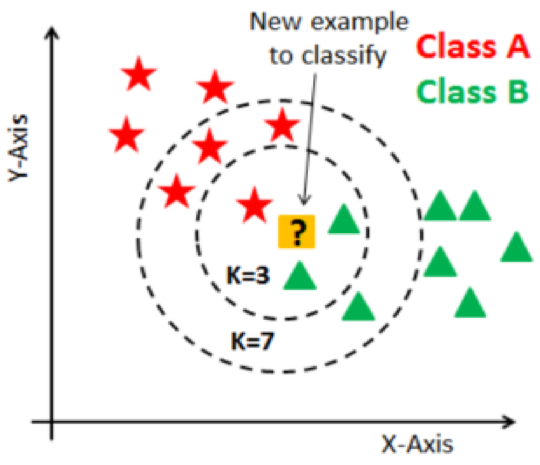


Figure 10: K-Nearest Neighbors Classifier[20].

3.4) **Logistic Regression (LR)**

is a this is a predictive model. It is a statistical learning technique used for the task of classification. Even though the name of the classifier has the word ‘Regression’ in it, it is used to produce discrete binary outputs.

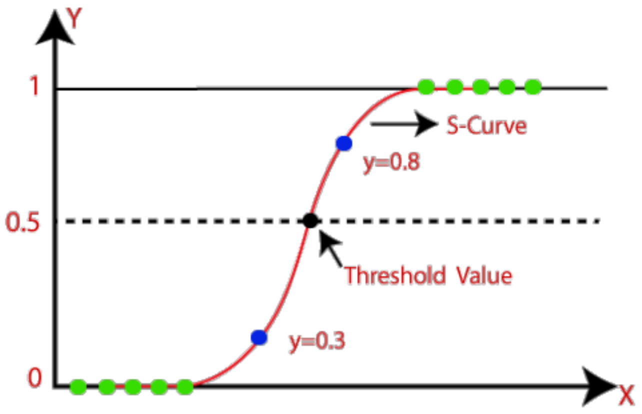


Figure 11: Logistic Regression[21] .

Chapter Three Implementation and Results

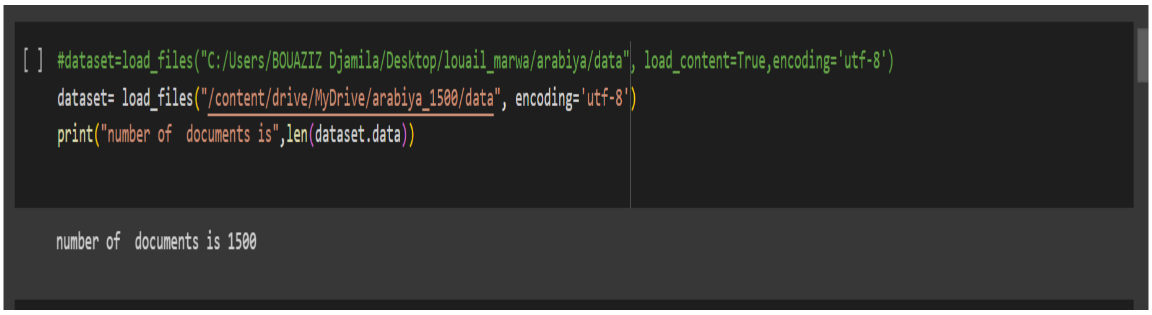
1 Dataset used

We used Al-Arabiya dataset, which is a dataset for text classification that is composed of 5 categories and a total of 1500 documents.

Al-Arabiya dataset was used in this work to evaluate the proposed FS methods. The features are selected from the training set only and, TFIDF for the training and test sets are calculated separately using the final sets of features obtained from either KNN, *Logistic regression*, *SVM* or *Random forest*. Table 1 shows the distribution of documents for the dataset .

Table 1: Al-Arabiya dataset.

Dataset	Train	Test	Total	No-classes
Al-Arabiya	12OO	300	1500	5



```
[ ] #dataset=load_files("C:/Users/BOUAZIZ Djamila/Desktop/louail_marwa/arabiya/data", load_content=True,encoding='utf-8')
    dataset= load_files("/content/drive/MyDrive/arabiya_1500/data", encoding='utf-8')
    print("number of  documents is",len(dataset.data))
```

```
number of  documents is 1500
```

Figure 12 : The dataset.

2 Development environment

We will talk in this section about the software we used to develop our program.

2.1-Python language

Python, created by Guido van Rossum, and released in 1991.[22] An interpreted language is a programming language which is generally interpreted, without compiling a program into machine instructions. It is one where the instructions are not directly executed by the target machine, but instead read and executed by some other program.[23]

2.2-Matplotlib

Matplotlib is one of the most successful and commonly used libraries in Python, it was created by John D. Hunter in Python programming language in 2003. It provides various tools to create 2D plots from the data in lists or arrays in python and makes use of NumPy.

2.3-NumPy

NumPy, stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. It was created in 2005 by Travis Oliphant. Numpy, is an open source project that can be used freely by developers and allows them to perform mathematical and logical operations on arrays.

3 Performance evaluation

We'll explain the technique how we can evaluate our machine learning classifiers, and how differentiate between.

- S: group of texts.

- True positives (TP): refers to the set of text documents that are related to a particular category **S**, and were truly forecast to be in category **S**. ˆ
- True negatives (TN): refers to the set of text documents that are not present in a particular category **S**, and were forecast to be in another category than **S**. ˆ
- False positives (FP): indicates the set of text documents that were set to be in a particular category **S**, but these texts actually belong to another category than **S**. ˆ

- False negatives (FN): indicates the set of text documents that were incorrectly forecast not related to a particular category **S**, but they actually belong to category **S**.

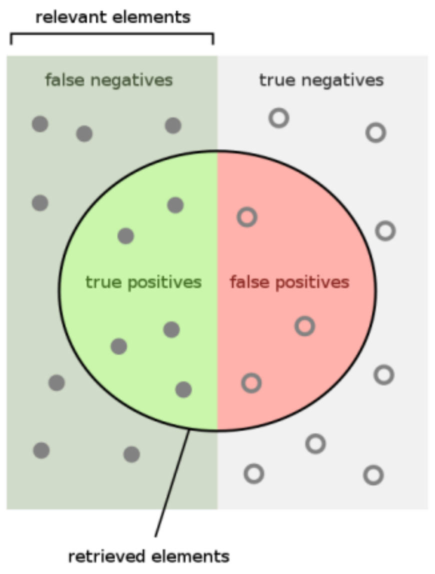


Figure 13 : Precision and recall.

3.1 Precision

Precision is defined as the percentage of correctly forecast texts of category S, and can be calculated as follows :

Precision= T P /(T P +F P) .

3.2 Recall

Recall is known as the percentage of real text documents of category S that were correctly forecast, and can be calculated as follows:

Recall = T P/(T P +F N).

4 Machine Learning Approaches

We trained four machine learning classification algorithms(KNN, SVM, LR, RF), and we compared the results obtained from each classifier by their accuracy and training time.

a- Text classification with TF-IDF

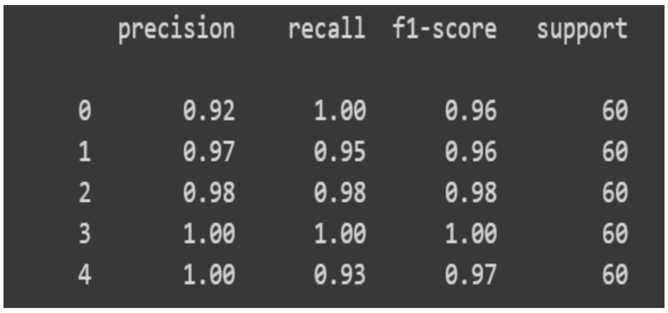
a.1 Prediction with linear SVM model

We divided our data into training and testing set (20%, 80%).
The results are in the following table:

Table 2 : SVM training \ testing time.

Training time	864.164 s.
Testing time	7.555 s.
Accuracy	0.98

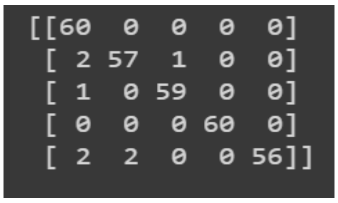
We also used some metrics in order to evaluate our model, such as precision, recall and f1-score for our 5 classes, the confusion matrix for our SVM model:



```
     precision    recall  f1-score   support

0         0.92      1.00      0.96        60
1         0.97      0.95      0.96        60
2         0.98      0.98      0.98        60
3         1.00      1.00      1.00        60
4         1.00      0.93      0.97        60
```

Figure 14: SVM metrics evaluations



```
[[60  0  0  0  0]
 [ 2 57  1  0  0]
 [ 1  0 59  0  0]
 [ 0  0  0 60  0]
 [ 2  2  0  0 56]]
```

Figure 15 : SVM Confusion Matrix.

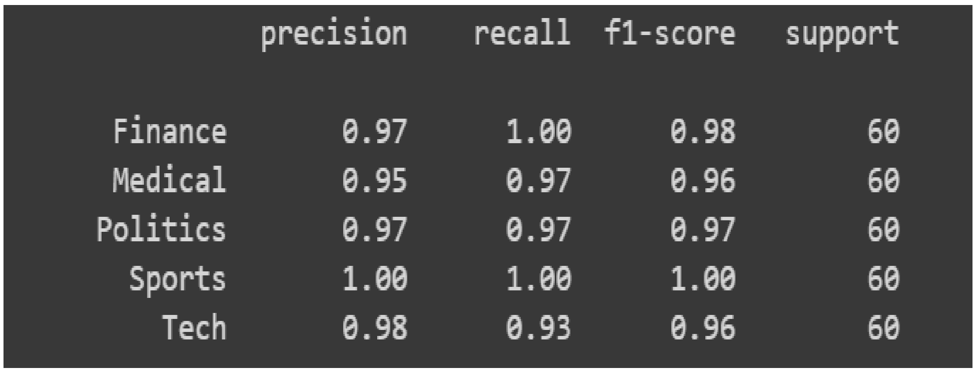
a.2 Prediction with LR model

The   Logistic Regression class from sklearn.linear model is the class responsible for solving regression problems via logistic regression. The results are:

Table 3 : Testing/training time and accuracy LR.

Training time	19.807s
Testing time	0.043s
Accuracy	0.973

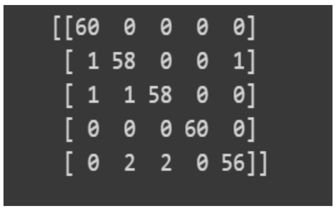
Our metrics evaluation for the 5 classes and the confusion matrix for our LR model:



```
              precision    recall  f1-score   support

     Finance       0.97      1.00      0.98        60
     Medical       0.95      0.97      0.96        60
    Politics       0.97      0.97      0.97        60
      Sports       1.00      1.00      1.00        60
        Tech       0.98      0.93      0.96        60
```

Figure 16 : LR metrics evaluation .



```
[[60  0  0  0  0]
 [ 1 58  0  0  1]
 [ 1  1 58  0  0]
 [ 0  0  0 60  0]
 [ 0  2  2  0 56]]
```

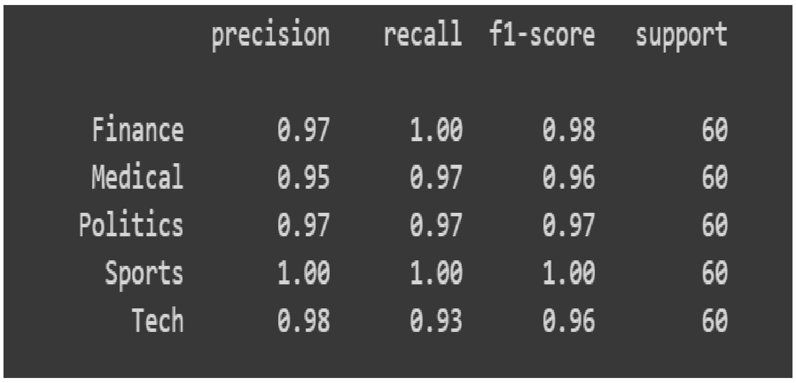
Figure 17: LR Confusion Matrix.

a.3 Prediction with RF model

The Random Forest Regressor class of the sklearn.ensemble library is used to solve regression problems via random forest. The most important parameter of the Random Forest Regressor class is the n -estimators parameter. This parameter defines the number of trees in the random forest. We set the n estimators parameter to 100 and these were the results we got:

Table 4 : Testing/training time and accuracy RF.

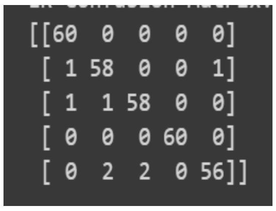
We used others metrics as mentioned previously for our 5 classes, alongside with the confusion matrix for our RF model:



```
              precision    recall  f1-score   support

     Finance       0.97      1.00      0.98        60
     Medical       0.95      0.97      0.96        60
    Politics       0.97      0.97      0.97        60
      Sports       1.00      1.00      1.00        60
        Tech       0.98      0.93      0.96        60
```

Figure 18 : RF metrics evaluation.



```
[[60  0  0  0  0]
 [ 1 58  0  0  1]
 [ 1  1 58  0  0]
 [ 0  0  0 60  0]
 [ 0  2  2  0 56]]
```

Figure 19 : RF Confusion Matrix.

a.4 Prediction with the KNN model

We used the K-Nearest-Neighbors-Classifier (KNN) class from the sk-learn-neighbors library to get the following results:

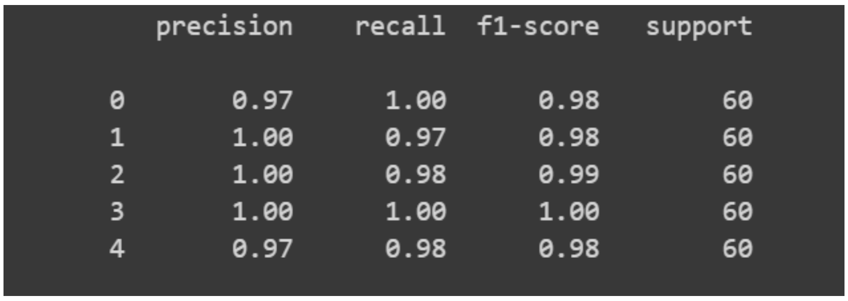
Table 5: Testing/training time and accuracy KNN.

Training time	19,804 s,
Testing time	0,043 s.
Accuracy	0,97

Training time	85,686 s.

Testing time	1,058 s.
Accuracy	0,99

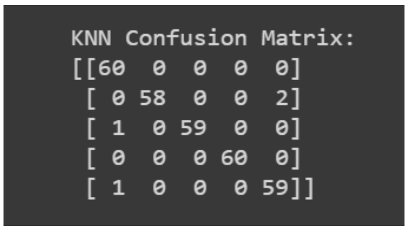
Our metrics evaluation for the 5 classes and the confusion matrix for our KNN model:



```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        60
           1       1.00      0.97      0.98        60
           2       1.00      0.98      0.99        60
           3       1.00      1.00      1.00        60
           4       0.97      0.98      0.98        60
```

Figure 20: KNN metrics evaluation.



```
KNN Confusion Matrix:
[[60  0  0  0  0]
 [ 0 58  0  0  2]
 [ 1  0 59  0  0]
 [ 0  0  0 60  0]
 [ 1  0  0  0 59]]
```

Figure 21: KNN Confusion Matrix.

The figure below summarizes the comparison between different classifiers based on training and testing times.

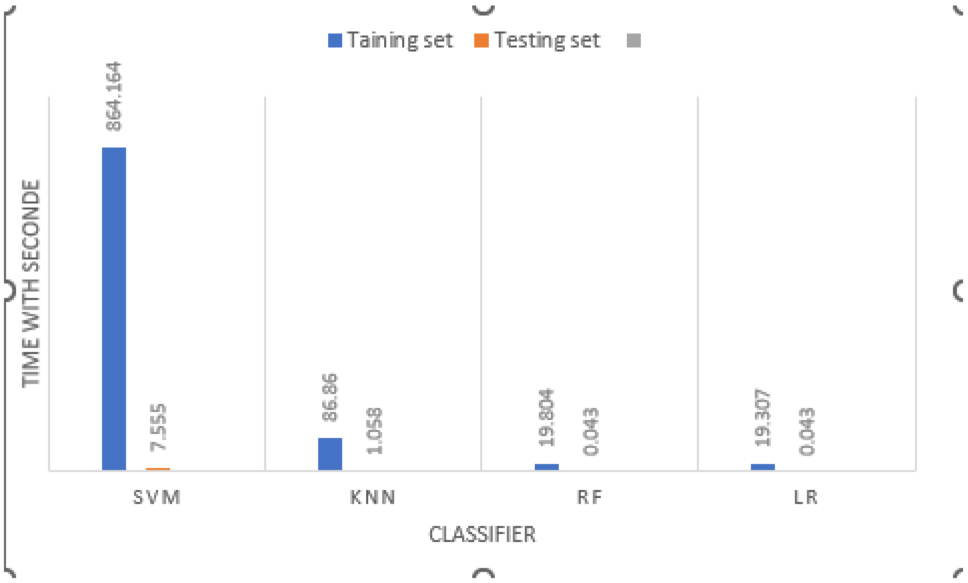


Figure 22 :Training and Testing time result.

All this result are those for the normal text classification, we want to have a good result for the accuracy of each classifier, so we will apply the dimensional reduction which is to minimize the words of our data to improve the accuracy of the classifiers.

b- Text classification with dimensionality reduction

b.1 CBOW part

After the part of preprocessing of our data arabiya 1500, we apply the CBOW code to give the vectors of each word in the data after the preprocessing.

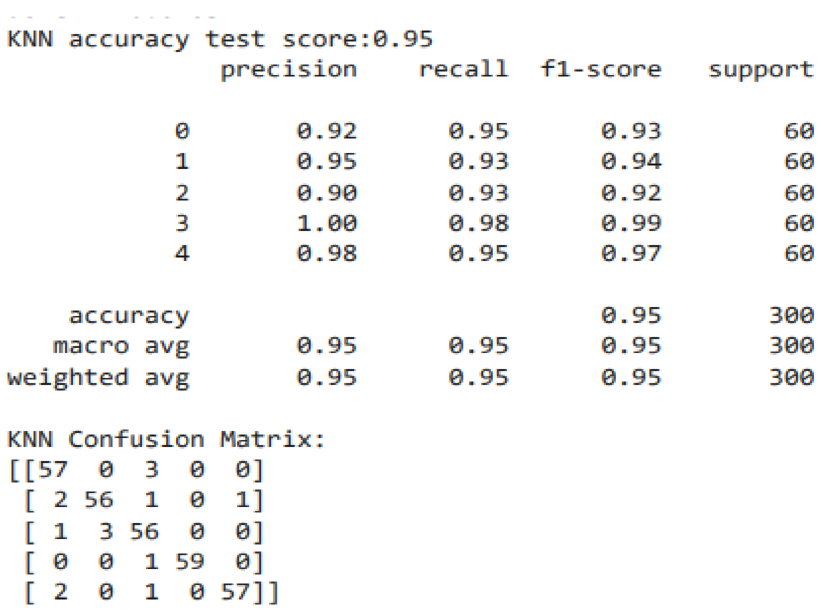
b.2 Clustring part

After we finished the words embedding with the CBOW code, we apply clustering, with the k-means method, based on similarity for result vectors (words). Then each cluster is represented by its center.

b.3 Our two methods

After the clustering, we proceed in two different ways for our system. The first one is to apply our four classifiers and see the result of the accuracy after the dimensionality reduction with camel tools as a natural language preprocess tokenizer. In the second way, we use another tokenizer which is python Arabic.
The results of the two methods are in the following figures (all this results in those figures are with 100 clusters):

- The first method: the accuracy in the four classifier is:



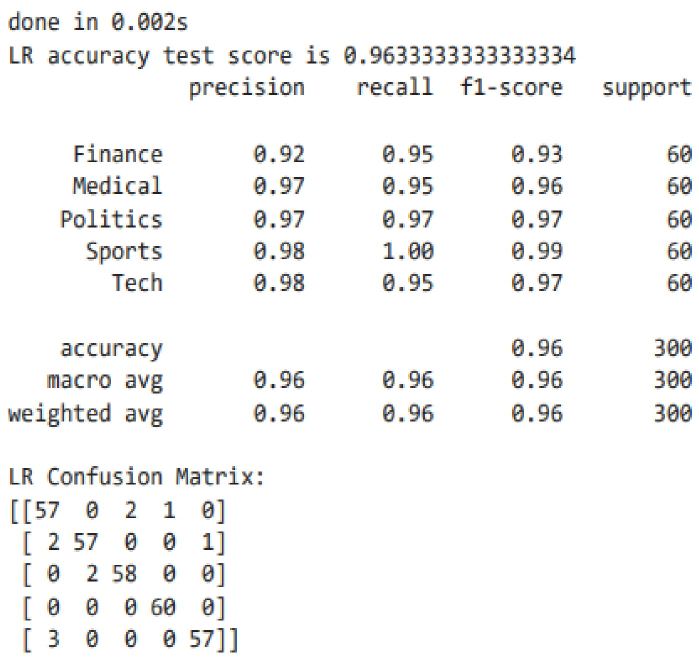
```
KNN accuracy test score:0.95
              precision    recall  f1-score   support

           0       0.92      0.95      0.93        60
           1       0.95      0.93      0.94        60
           2       0.90      0.93      0.92        60
           3       1.00      0.98      0.99        60
           4       0.98      0.95      0.97        60

    accuracy                           0.95       300
   macro avg       0.95      0.95      0.95       300
weighted avg       0.95      0.95      0.95       300

KNN Confusion Matrix:
[[57  0  3  0  0]
 [ 2 56  1  0  1]
 [ 1  3 56  0  0]
 [ 0  0  1 59  0]
 [ 2  0  1  0 57]]
```

Figure 23:Knn accuracy result.



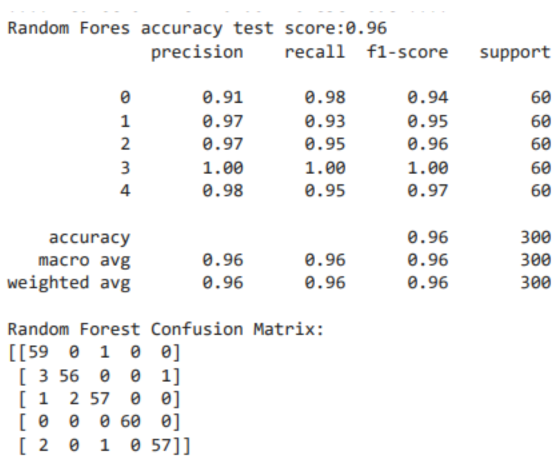
```
done in 0.002s
LR accuracy test score is 0.9633333333333334
              precision    recall  f1-score   support

     Finance       0.92      0.95      0.93        60
     Medical       0.97      0.95      0.96        60
    Politics       0.97      0.97      0.97        60
      Sports       0.98      1.00      0.99        60
        Tech       0.98      0.95      0.97        60

    accuracy                           0.96       300
   macro avg       0.96      0.96      0.96       300
weighted avg       0.96      0.96      0.96       300

LR Confusion Matrix:
[[57  0  2  1  0]
 [ 2 57  0  0  1]
 [ 0  2 58  0  0]
 [ 0  0  0 60  0]
 [ 3  0  0  0 57]]
```

Figure 24:LR accuracy result.



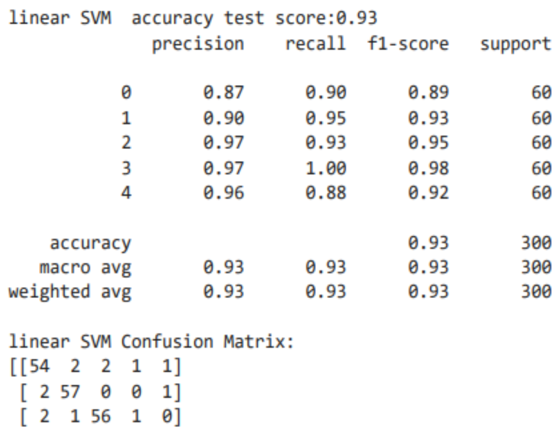
```
Random Fores accuracy test score:0.96
              precision    recall  f1-score   support

           0       0.91      0.98      0.94        60
           1       0.97      0.93      0.95        60
           2       0.97      0.95      0.96        60
           3       1.00      1.00      1.00        60
           4       0.98      0.95      0.97        60

    accuracy                           0.96       300
   macro avg       0.96      0.96      0.96       300
weighted avg       0.96      0.96      0.96       300

Random Forest Confusion Matrix:
[[59  0  1  0  0]
 [ 3 56  0  0  1]
 [ 1  2 57  0  0]
 [ 0  0  0 60  0]
 [ 2  0  1  0 57]]
```

Figure 25:RF accuracy result.



```
linear SVM  accuracy test score:0.93
              precision    recall  f1-score   support

           0       0.87      0.90      0.89        60
           1       0.90      0.95      0.93        60
           2       0.97      0.93      0.95        60
           3       0.97      1.00      0.98        60
           4       0.96      0.88      0.92        60

    accuracy                           0.93       300
   macro avg       0.93      0.93      0.93       300
weighted avg       0.93      0.93      0.93       300

linear SVM Confusion Matrix:
[[54  2  2  1  1]
 [ 2 57  0  0  1]
 [ 2  1 56  1  0]
```

Figure 26:SVM accuracy result.

- **The second method (with py-arabic tokenizer) :**

The results of all the clusters and all the classifiers are in **the table 8 .**

We tried to explain and detail the experiments by illustrating the results in the following tables show us all the results:

- The first table is for accuracy without dimensionality reduction :

The classifiers	Accuracy
KNN	0,99
SVM	0,98
LR	0,973
RF	0,97

Table 6 : Accuracy without dimensionality reduction.

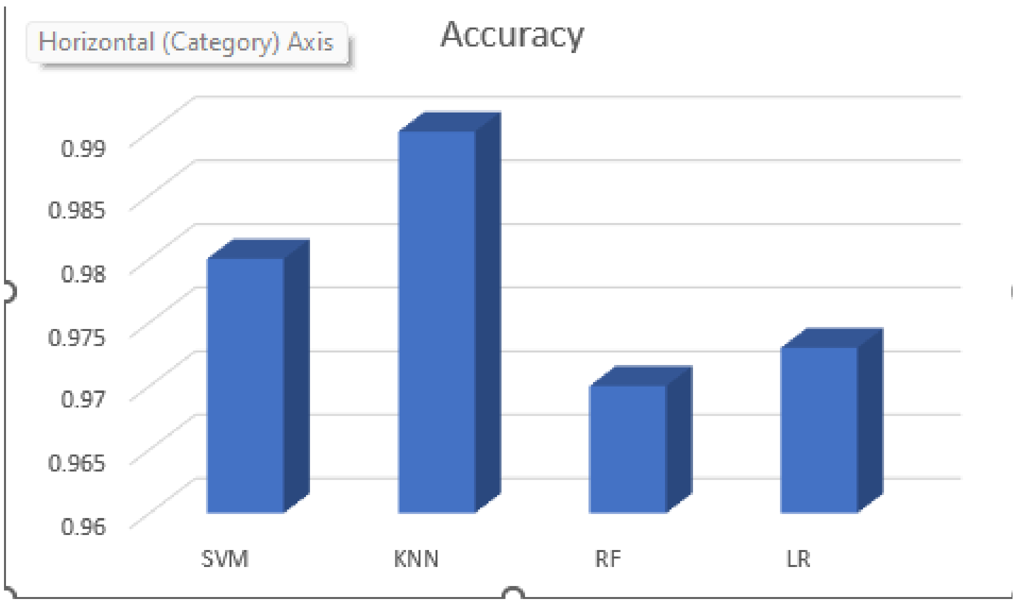


Figure 27: Result of accuracy.

- The second one is for the result after the dimensionality reduction with our *two methods* :

The first one: with camel-tools tokenizer.

Table 7:Accuracy with Camel.

	100clusters	TS/TR Time	200cluster	TS/TR Time	300cluster	TS/TR Time	400clusters	TS/TR Time
Knn	0.95	1.098s/ 3.614s	0.95	1.098s/ 8.589s	0.95	1.108s/8.596s	0.94	1.110s/8.940s
Svm	0.93	1.029s/ 107.75s	0.96	1.034s./ 181.137 s	0.94	1.098s/ 192.200s	0.95	1.123s/198.597s
Lr	0.96	0.002s/ 19.807s	0.96	0.001s/ 19.809s	0.96	0.002s/19.811s	0.98	0.001s/19.812s

Rf	0.93	0.043s/ 19.804s	0.97	0.043s/ 19.805s	0.96	0.044s/19.807s	0.96	0.045s/19.810s

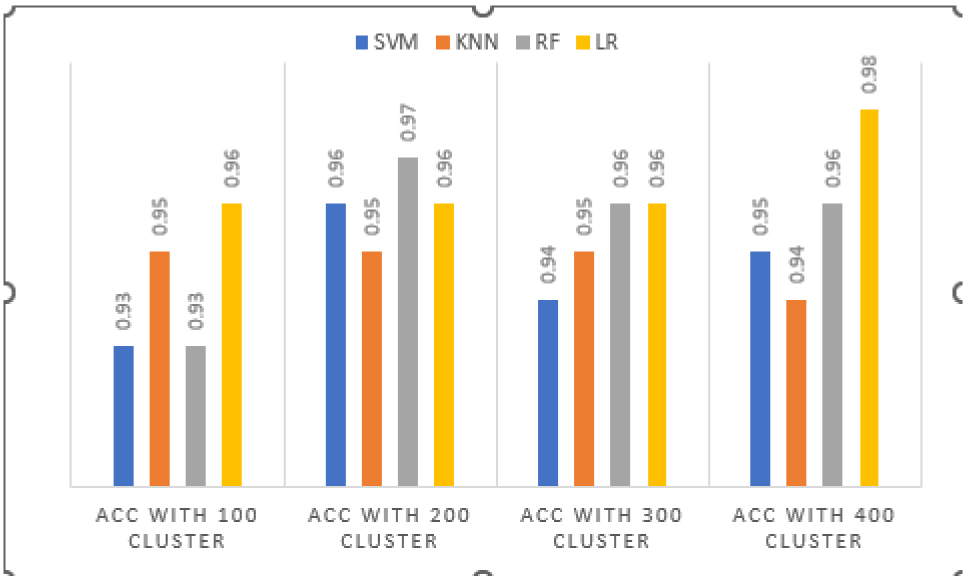


Figure 28: Histogramme accuracy result First method.

The second one : with pyarabic tokenizer

Table 8: Accuracy with PyArbic.

	100clusters	TS/TR Time	200cluster	TS/TR Time	300cluster	TS/TR Time	400clusters	TS/TR Time
Knn	0.95	0.012s/ 3.614	0.95	0.021s/ 2.850s	0.95	0.019s/1.698S	0.94	0.10s/ 7.641S
Svm	0.93	1.040s /107.75 s	0.96	1.032s/ 18.137s	0.94	1.020s/ 19.200s	0.95	1.005s /11.597s
Lr	0.97	0.002s/ 19.807s	0.97	0.001S/ 19.813s	0.97	0.002S/19.814s	0.98	0.001S /19.8019s
Rf	0.96	0.041s/ 19.805s	0.96	0.041s/ 19.809s	0.96	0.043s/19.811s	0.96	0.044s /19.815s

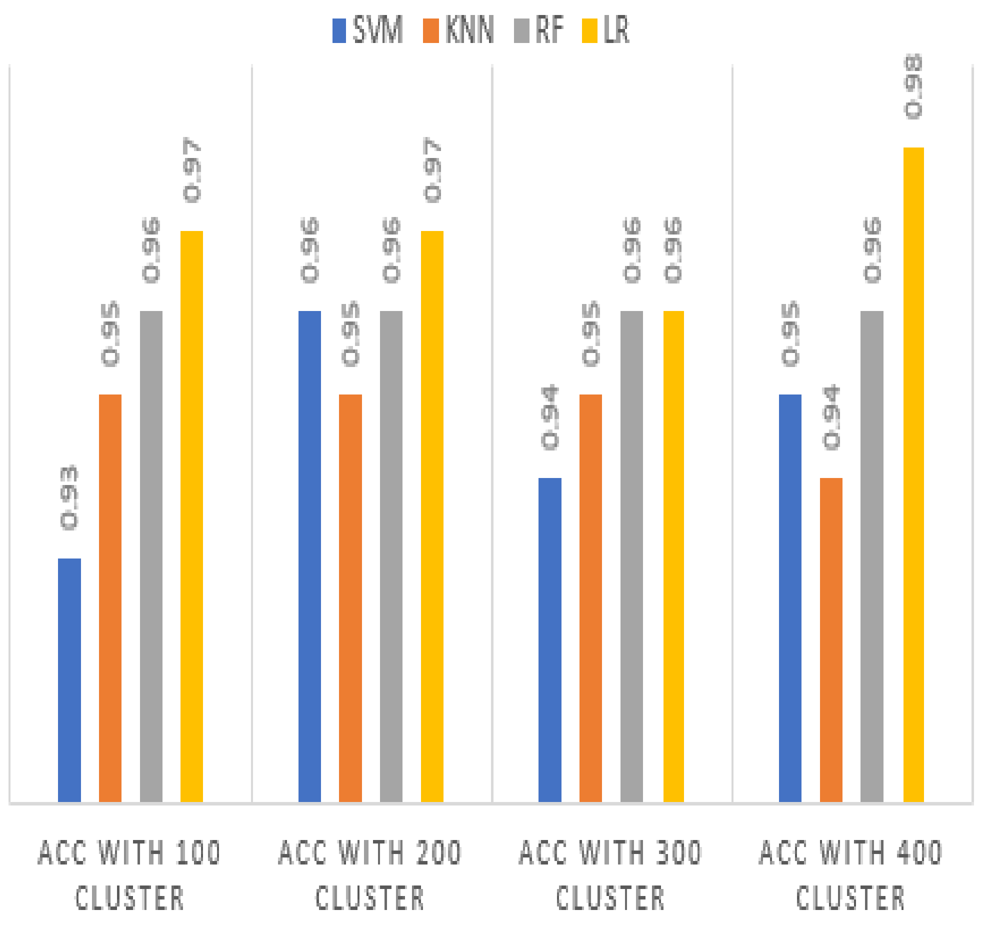


Figure 29: Histogramme Accuracy result method two.

The reported results show that classifiers perform well with TF-IDF. Accuracy vary from 97% to 99%, but they take an important time during training phase. Using our embeddings-based reduction, we notice that all classifiers pass to lower accuracy. We can explain this by the fact that we have reduced 29298 to 100, 200 300 and 400 feature. We think that this number of features may be small. Second argument is that each cluster is represented by its center which is first randomly chosen by k-means. The training time decreases with reduction and this is a straightforward consequence. We see that the LR classifier is the unique classifier which makes an accuracy closer to the normal text classification.

General Conclusion and Future Works

In our work we have studied an active research problem in Natural Language Processing which is dimensionality reduction. For this purpose, we have studied in first the different word embeddings methods. We have also reviewed all steps required for a text classification system: from cleaning, tokenizing texts, representing text as numerical vectors to classifiers' models. Hence, we have used SVM, KNN, LR and RF to classify our dataset.
We have use reduced set of Al-Arabyia dataset: 1500 documents. These documents are written in Arabic language and belong to 5 different classes.
As a baseline, we have first tested a text classification using TF_IDF a method to represent the documents.
Then, we have used CBOW-based embeddings, as a dimensionality reduction method, in more precise words, as feature selection method. Therefore, embeddings of the total vocabulary is clustered using K-means and only the centers of the clusters are kept to represent the whole features' set. Because K-means algorithm, requires the input of k (the number of clusters to be generated), we have tested the system on different values of K (100, 200, 300 and 400).
We have also noticed that there is a difference in the result of preprocessing when we use Camel tools or PyArabic tokenizers. So we have tested the system with both tools.
The reported results show that classifiers perform well with TF-IDF. Accuracy vary from 97% to 99%, but they take an important time during training phase. Using our embeddings-based reduction, we notice that all classifiers pass to lower accuracy. We can explain this by the fact that we have reduced 29298 to 100, 200 300 and 400 feature. We think that this number of features may be small. Second argument is that each cluster is represented by its center which is first randomly chosen by k-means. The training time decreases with reduction and this is a straightforward consequence.

As future work, we can extend the research to more than 400 clusters. Because, we think that 400 feature is so small compared to 29298 features generated by TF-IDF.

References

(1) Suleiman, D., Awajan, A. Comparative Study of Word Embeddings Models and Their Usage in Arabic Language Applications. In 2018 International Arab Conference on Information Technology (ACIT), IEEE: Werdanye, Lebanon, 2018, pp 1–7. https://doi.org/10.1109/ACIT.2018.8672674.

(2) Dutta, M. Word2Vec For Word Embeddings -A Beginner's Guide. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/ (accessed 2023-04-25).

(3) Lenny 2: Autoencoders and Word Embeddings | by Lenny Khazan | A Year of Artificial Intelligence. https://ayearofai.com/lenny-2-autoencoders-and-word-embeddings-oh-my-576403b0113a (accessed 2023-04-25).

(4) Word Co-Occurrence Matrix Example, 2020. https://www.youtube.com/watch?v=-F-PlgX8GcY (accessed 2023-04-25).

(5) Pennington, J., Socher, R., Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics: Doha, Qatar, 2014, pp 1532–1543. https://doi.org/10.3115/v1/D14-1162.

(6) 5Vol100No2.Pdf. http://www.jatit.org/volumes/Vol100No2/5Vol100No2.pdf (accessed 2023-05-06).

(7) Mojumder, P., Hasan, M., Hossain, F., Hasan, K. M. A Study of FastText Word Embedding Effects in Document Classification in Bangla Language, 2020, pp 441–453. https://doi.org/10.1007/978-3-030-52856-0_35.

(8) Wahdan, A., Al Hantoobi, S., Salloum, S. A., Shaalan, K. A Systematic Review of Text Classification Research Based on Deep Learning Models in Arabic Language. IJECE 2020, 10 (6), 6629. https://doi.org/10.11591/ijece.v10i6.pp6629-6643.

(9) Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, Brown. Text Classification Algorithms: A Survey. Information 2019, 10 (4), 150. https://doi.org/10.3390/info10040150.

(10)Aggarwal, C. C., Zhai, C. A Survey of Text Classification Algorithms. In Mining Text Data, Aggarwal, C. C., Zhai, C., Eds., Springer US: Boston, MA, 2012, pp 163–222. https://doi.org/10.1007/978-1-4614-3223-4_6.

(11)Pahwa, B., Taruna, S., Kasliwal, N. Sentiment Analysis- Strategy for Text Pre-Processing. IJCA 2018, 180 (34), 15–18. https://doi.org/10.5120/ijca2018916865.

(12)Christanti Mawardi, V., Susanto, N., Santun Naga, D. Spelling Correction for Text Documents in Bahasa Indonesia Using Finite State Automata and Levinshtein Distance Method. MATEC Web Conf. 2018, 164, 01047. https://doi.org/10.1051/matecconf/201816401047.

(13)Raschka, S. Naive Bayes and Text Classification I - Introduction and Theory. arXiv February 14, 2017. http://arxiv.org/abs/1410.5329 (accessed 2023-05-07).

(14)Whitney, D. L., Evans, B. W. Abbreviations for Names of Rock-Forming Minerals. American Mineralogist 2010, 95 (1), 185–187. https://doi.org/10.2138/am.2010.3371.

(15)Dhuliawala, S., Kanojia, D., Bhattacharyya, P. SlangNet: A WordNet like Resource for English Slang.

(16)Rudner, L. M., Liang, T. Automated Essay Scoring Using Bayes' Theorem. The Journal of Technology, Learning and Assessment 2002, 1 (2).

(17)Keselj, V., Peng, F., Cercone, N., Thomas, C. N-GRAM-BASED AUTHOR PROFILES FOR AUTHORSHIP ATTRIBUTION.

(18)Introduction to Supervised Learning - Coding Ninjas. https://www.codingninjas.com/codestudio/library/introduction-to-supervised-learning (accessed 2023-05-09).

(19)Mishra, S. Breaking Down the Support Vector Machine (SVM) Algorithm. Medium. https://towardsdatascience.com/breaking-down-the-support-vector-machine-svm-algorithm-d2c030d58d42 (accessed 2023-05-09).

(20)Shah, R. Introduction to k-Nearest Neighbors (kNN) Algorithm. Medium. https://ai.plainenglish.io/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8 (accessed 2023-05-09).

(21)Logistic Regression in Machine Learning - Javatpoint. https://www.javatpoint.com/logistic-regression-in-machine-learning (accessed 2023-05-09).

(22)What is Python. https://www.pythontutorial.net/getting-started/what-is-python/ (accessed 2023-05-17).

(23)Difference between Compiled and Interpreted Language. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/ (accessed 2023-05-17).