Identification of the class of English text among several classes

Graduation project documentation

ID	Name
4142001	Islam Samir Sayed
4142009	Ahmed Nagy Radwan
4141066	Mohamed Rashad
4141069	Mohamed El Kesay
4141179	Ahmed Abboud Hanafy
4141126	Gehad Khaled

Under the supervision of

Dr.Ahmed Diaa

2018

Abstract

Automatic text classification is one of the important research issues and it has very wide uses such as in information retrieval(IR).Nowadays, Information retrieval has several applications in modern technology, one of them is web search engines. Our project is a research-based one that aims to apply different approaches of machine learning to get the highest model which can classify texts automatically under predefined classes. All experiments were done on the same dataset to make a fair comparsion between the proposed approaches.The selected dataset is a dataset for news and articles collected and released by BBC. Proposed approaches that are applied on the dataset contains different combinations of learning algorithms such as Naïve Bayes, Decision tree, Support Vector machines, and Random forests. Also other techniques of stemming and feature selection are applied such as Chi-square, and information gain in feature selection and Porter Stemmer and Stanford CoreNLP in stemming. Text representation also has a vital role in the text classification process, so we

applied different text representation techniques such as term frequency weighting , TF-IDF, and N-grams. The testing cases are sampled from the original dataset such that we divided that dataset into 70% traning set and 30% testing set. The best approach came up with 97.5% accuracy.

Table of contents

Chapter 1

Introduction

In this chapter , we describe the idea behind the project in order to answer questions like , what's the purpose of the project? what's the importance of it?. We also discuss text classification issue and how this issue was solved in the past and how can be solved in this era. Also we discuss how we would apply experiments through the study.

Section 1.1 illustrates text classification process and how it could be solved using different techniques. Section 1.2 states the objectives of the study. Section 1.3 illustrates how we selected a dataset to apply our experiments.

1.1 Overview

Automatic text classification is one of the important research issues and also has very wide applications such as in information retrieval(IR) or in text mining in general. Nowadays, information retrieval has different implementations in modern technology, one of them is web search engines.

From several decades , a lot of tries were spent to accomplish text classification task in automated manner. Until late 80's, the task was implemented by knowledge –based systems/expert systems where a set of rules were defined manually by knowledge engineers to construct knowledge repository and then inference engine should conduct from knowledge to classify documents under the given categories .This approach was an inefficient since it requires human intervention in knowledge storing.

In 90's, many machine learning techniques were arisen and proposed to solve many problems which were depended on expert systems. The advantages of machine learning techniques over expert systems are the accuracy and no human intervention needed to construct text classification tools.

Machine learning provided different techniques to solve many problems not only text classification, these techniques can help in text analysis or data mining generally and also they are very helpful in natural language processing(NLP) and in many other fields, but this study focus only text classification.

Text classification is the task of classifying a document under a predefined classes, this definition is an intuitive one. A formal definition is needed for clear understanding of text classification problem.

In text classification, we are given a description $d_i \in X$ of a document, where X is the document space and a fixed set of classes $C = \{c1, c2, \dots, cn\}$. Classes are also called categories or labels. Typically, the document space X is a high-dimensional space. The text classification task is the process of assigning one class C_j to a document d_i .

This study concentrates to machine learning techniques in text classification, this problem in the statistical learning theory is considered to be classification problem which is know as a supervised-learning model.

For clear understanding of the learning problem in general and the classification problem in particular from the perspective of statistical learning theory, formalization is needed. The learning problem or the

classification problem can be described as in Figure 1.1 that illustrates the basic setup or basic procedure for the learning problem.

Learning problem can be characterized by a group of components as follows :

- Input: x (Vector of features values)
- Output: y (The output class)
- Target function $F: x \rightarrow y$ (an ideal formula that maps x to y)
- Training data: $(x_1,y_1) \ldots (x_n,y_n)$ (a matrix of observations)
- Hypothesis $g: x \rightarrow y$,where $g \approx F$

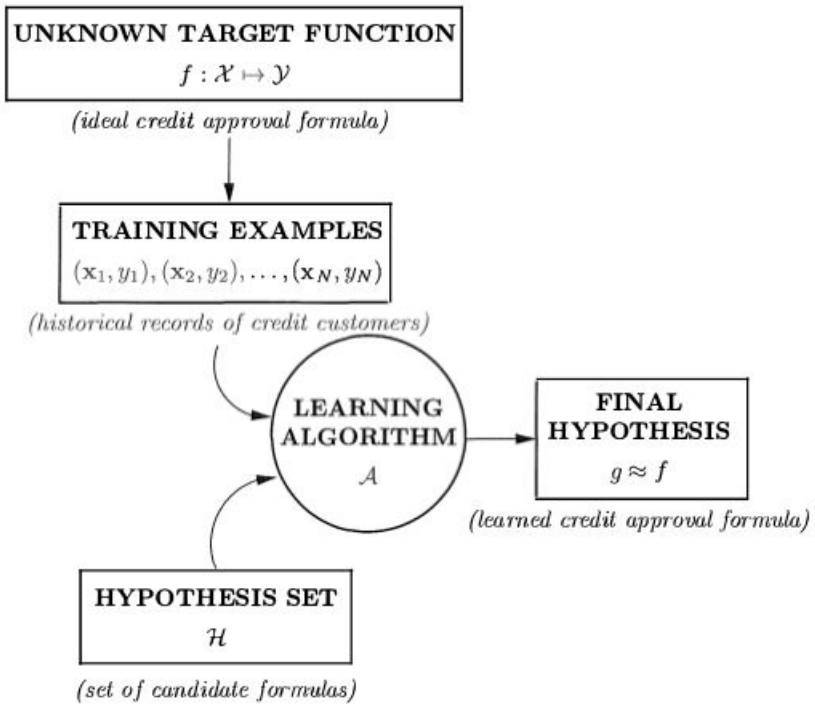

Figure 1.1: Basic procedure of Learning/Classification problem

Figure 1.1 and the learning components can be explained as follows:

- Training set: it's a dataset D of input-output observations/examples/instances (x_1,y_1) … (x_n,y_n) where n is the number of observations and x is the vector of values that corresponds to some feature space, and y is the class of each dataset observation.
- Target function: is an assumption of ideal formula that can predict the class y_i for any input x_i.
- Hypothesis: it's a trained model that approximates the target function.
- Learning Algorithm: it's an algorithm which should pick up one hypothesis $h_k \in H = \{h1, h2, \dots, hz\}$ that approximates as possible the target function F.

1.2 Objectives

This study aims to apply many experiments using different machine learning techniques on a dataset to investigate the impact of each approach to enhance the accuracy of classification, and determining the expected accuracy of each approach , these proposed approaches will be declared and discussed in detail in the next chapter. Although No

Free Lunch Theorem states that there's no approach will be the best fit for all situations, controlled experiments of machine learning techniques are very useful to conduct pros and cons of each technique.

At the end of the study, we should be able to evaluate the accuracy of each approach that should be applied on the selected dataset such that all experiments should be applied on the same dataset.

1.3 Dataset

In order to achieve the objectives of the study, the dataset should satisfies some characteristics. Firstly, the selected dataset should have one or more predefined class for its instances also it's better to be multinomial categorized rather than binomial one to make the differences among approaches clear. Secondly, the balanced distribution property is needed for some categories of classifiers such as probabilistic models to enhance the prediction accuracy of the model, thus the balanced distribution of the classes also is required. Generally, there's no plenty of datasets for classification purpose, most of available dataset either related to news and articles or semantic analysis or spamming messages.

Our choice was a dataset for news/articles collected and released by BBC. Figure 1.2 shows how the documents in the dataset is organized and

categorized into five classes(multinomial property). Figure 1.3 shows a sample of the dataset documents which clarifies that the dataset is ready for text processing without cleaning drawbacks such as the date of articles or the author name or any undesired texts are involved in the documents. Figure 1.4 shows a histogram of class distribution which ensures that the dataset satisfies the balanced property.

Class Name	Number of documents
Business	510
Entertainment	386
Politics	417
Sport	511
Tech	401

Figure 1.2: BBC dataset description table

```
Japan narrowly escapes recession

Japan's economy teetered on the brink of a technical recession in the three months to
September, figures show.

Revised figures indicated growth of just 0.1% - and a similar-sized contraction in the
previous quarter. On an annual basis, the data suggests annual growth of just 0.2%,
suggesting a much more hesitant recovery than had previously been thought. A common
technical definition of a recession is two successive quarters of negative growth.

The government was keen to play down the worrying implications of the data. "I maintain the
view that Japan's economy remains in a minor adjustment phase in an upward climb, and we
will monitor developments carefully," said economy minister Heizo Takenaka. But in the face
of the strengthening yen making exports less competitive and indications of weakening
economic conditions ahead, observers were less sanguine. "It's painting a picture of a
recovery... much patchier than previously thought," said Paul Sheard, economist at Lehman
Brothers in Tokyo. Improvements in the job market apparently have yet to feed through to
domestic demand, with private consumption up just 0.2% in the third quarter.
```

Figure 1.3: sample document of business class in BBC dataset

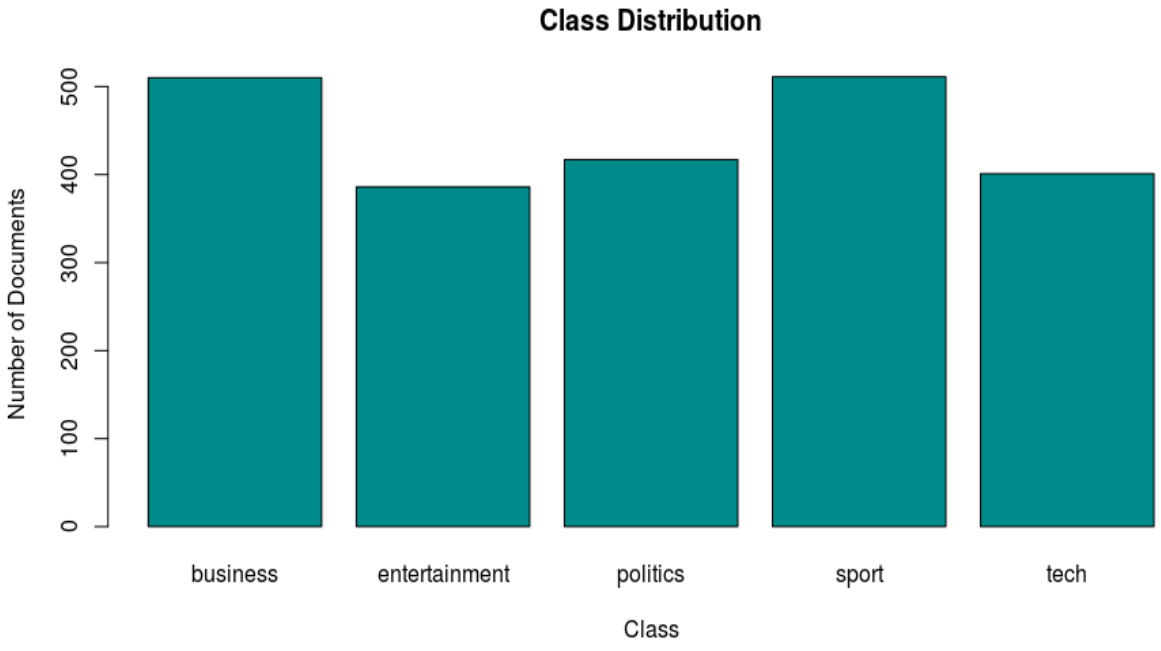

Figure 1.4: histogram for class distribution of BBC dataset

Chapter 2

Theory

In this chapter, we discuss proposed approaches to solve the problem and how it can be implemented theoretically. Also we explain each different technique separately in terms of its technical aspects and also a an intuitive idea and the motivation behind it. The proposed approaches to solve the problem that discussed in this chapter will be later tested through the study will be declared.

Section 2.1 illustrates text classification process and its phases with an introduction for each phase that would be discussed in detail in next sections. Section 2.2 illustrates text representations techniques the could be used through text classification process. Section 2.3 illustrates stemming and lemmatization concepts and how they can be implemented using different algorithms and different tools. Section 2.4 illustrates the process of feature selection and how it can be implemented using different techniques. Section 2.5 illustrates in detail the concept of learning algorithm with explanation for different algorithms that will be used through the study. Section 2.6 states our proposed apporaches that will be experimented on the dataset through the project.

2.1 Text classification process

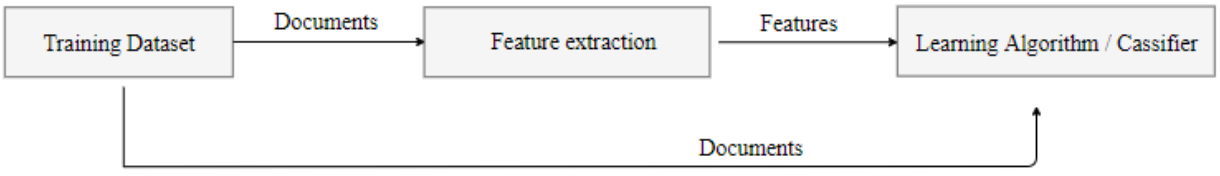

Figure 2.1: Basic setup for text classification process

In text classification the file which contains the description of the observation is knows as document. Figure 2.1 shows the basic setup to accomplish classification task as block diagram. First task should be accomplished in text classification process is feature extraction. In text classification feature extraction also is known as constructing vocabulary, vocabulary is symbolized by V is a set of words/features that contains the dimensions of the document space such that each document can represented as a vector in this space. More formally, each document $d_i \in R^{|v|}$, where |V| is the size of vocabulary/feature space and each word or term in the vocabulary is considered as a dimension in the feature space.

The previous process of constructing vocabulary is called Bag of word(BOW) or vector space model(VSM) both can be used interchangeably, this issue will be discussed in detail in the next chapter.

The learning algorithm or classifier is an algorithm that usually take a matrix A as an input, A is a $m \times n$ matrix ,where m is the number of rows and n is the number of columns. Figure 2.1 shows that the classifier takes two inputs, they are feature set and documents, these two inputs are augmented in the matrix A by representing the features as columns and the observations as rows. Figure 2.2 shows the form of matrix A, each element $a_{i,j}$ represents the value of feature j of the observation i.

$$\mathbf{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Figure 2.2: matrix A

Through text classification process there are some techniques which can be represented as phases, as we will see later, these techniques can enhance classification results in terms of accuracy. These techniques such as stemming and feature selection, also we will discuss them in detail in next sections.

The process of feature extraction contains different tasks that should be done to build the feature space/vocabulary. Firstly, we will start with raw documents such as the sample document which is displayed in Figure

1.3, then we should transform all training dataset documents to just a set of words which is vocabulary. Figure 2.2 shows the phases of feature extraction phases.

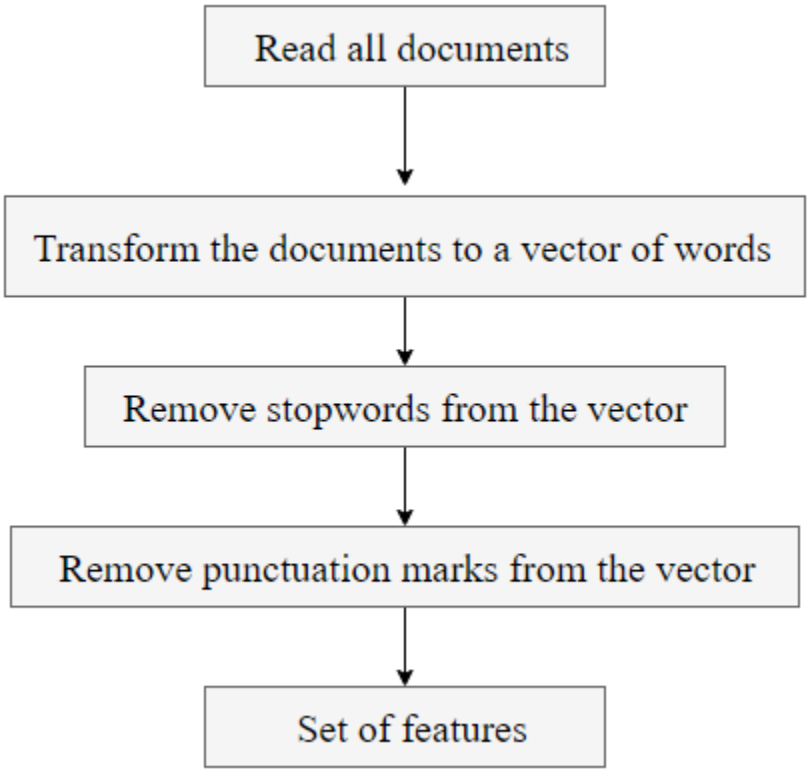

Figure 2.2: feature extraction process

Firstly, all training dataset documents should be loaded together and then tokenized by separating words by the spaces among them, then finally we will get overall training dataset as a vector of words, such a representation is ready for text processing.

The next step is a filtration process to get only the meaningful elements from the vector and remove the useless words and punctuation

marks or numbers and symbols. The filtration should be done as shown in Figure 2.2 by two steps, firstly we should remove stopwords. Stopwords are words with no meaning on the context such as what, who, which, were, and, or, she, and he. Those words are stored as a list and each word in the vector should be comparing against this list, and if any of the vector words were similar to any in the stopwords list the word should be removed from the vector. The next step in the filtration is to remove any symbol or number from each word in the vector, those symbols and numbers also known as punctuation marks.

After these steps a set of words/features will be obtained. These steps should be done accurately since constructing vocabulary is very sensitive process that will affect the classification negatively.

2.2 Text Representation

Text classification is a concept consider about how the document can be represented, in other words how a set of sentences can be represented for text classification purpose. In the previous section we discussed bag of word model that describe each document as a set of words with respect to some dimensions(vocabulary/feature space). In this section we will discuss the idea of term frequency weighting and also we will discuss the idea of tf-idf weighting and n-gram.

2.2.1 Term frequency weighting

According to bag of word model, each document will be represented as a vector of values with respect to a feature space/vocabulary. Now, the question is about what's the value of each dimension? Here is the answer term frequency weighting is the process of giving a weight for each term in the vocabulary based on the number of how often the term occurs in the document. For example:

- given a vocabulary V = { good, country, cat, feeling }.
- given a document d_1 : Egypt is a good country with good places.
- given a document d_2 : the cat had a good feeling.

According to term frequency weighing we should represent the two documents as in Figure 2.3, each value represents the number of occurrences of this feature/term/word inside the vocabulary.

	good	country	cat	feeling
d1	2	1	0	0
d2	1	0	1	1

Figure 2.3: Document term matrix for two documents

In section 2.1 , we discussed the idea of the matrix of observation, Figure 2.3 displays a real example for this matrix which in this case is called document term matrix. Document term matrix(DTM) is the conventional representation of the documents according to the feature space that implement the idea of bag of word such that each in the matrix column represents the feature and each row represents a document/observation.

2.2.2 TF-IDF weighting

TF-IDF is a short for term frequency-inverse document frequency, it's another technique for term weighting, it differs from term frequency.

This technique tries to seek about the common terms which don't express the context of the document, high tf-idf weight represents the terms which is not common among documents, and low tf-idf represents the terms which is common between the documents. Tf-idf depending on the idea of if the terms is repeated and common between documents won't be efficient to express its context or classification process can based on it. Tf-idf combine the definitions of two concepts the first one is term frequency and the second one is inverse document frequency.

$$tf - idf(t,d) = tf(t,d) \times idf(t,D)$$

Equation 2.1: tf-idf equation

The value of Equation 2.1 will be in the range of [0,1] interval. We are familiar with tf(t,d) and how it can be calculated from Section 2.2.1 but we are not familiar with idf(t,D).

$$idf(t, D) = \log \frac{|D|}{n(t)}$$

Equation 2.2: inverse document frequency

Equation 2.2 states how inverse document frequency can be calculated, where D is the set of documents that the corpus contains and |D| the size of documents in the corpus(the number of documents in D), n(t) is the number of documents where t occurs within them.

Here's some explanation of tf-idf weights:

- highest when t occurs many times within a small number of documents(thus lending high discriminating power to those documents).
- Lower when the term occurs fewer times in a document, or occurs in many documents(thus offering a less pronounced relevance signal).
- Lowest when the term occurs in virtually all documents.

2.2.3 N-Gram

So far, we only deal with the vocabulary as a collection of terms each term contains only single word. In other representations we can represent terms in the vocabulary such that each term can contain more than one word. One of the most known representations techniques that can offer this idea is N-gram.

N-gram is a representation technique for vocabulary and documents that represents documents as a collection of terms where each term contains a fixed number n of words. We can think of N-gram as an extension of bag of word model, also it can be generalization for bag of word model where bag of word is 1-grams which also is called unigrams.

In N-gram representation, we can use unigrams, bigrams, trigrams or more than three. Here's an example for bigrams representation:

- Given a sentence : look like duck swim like duck quack like duck probably duck

Figure 2.4 shows how this sentence can be represented as a bigrams.

look_like	like_duck	duck_swim	swim_like	duck_quak	quack_like	duck_probabl	probabl_duck
1	3	1	1	1	1	1	1

Figure 2.4: bigrams representation

2.3 Stemming and lemmatization

Stemming is a pre-processing task in text mining in general and text classification in particular and it's very common task in natural language processing(NLP). Stemming is very useful in information retrieval systems. The purpose of stemming is to get the root form for different forms such as noun, adjective, verb, adverb and so on.

Typically, different forms for a word with the same root have similar semantic interpretation and might be replaced with each other. Since both forms of a word have the same meaning, we could reduce the two or more forms of a word to just one form, which is the root form. To achieve this goal, there are a lot of stemming algorithms for this purpose. Most of them are similar in many procedures with a different aspect for each stemmer that differs it from the other. Stemming is a powerful tool to reduce the feature space in text classification.

Word	Stemmer 1	Stemmer 2
cats	*cat*	*cat*
meekly	*meekli*	*meek*
pompous	*pompou*	*pompous*
thursday	*thursdai*	*thursday*
tied	*ti*	*tie*

Figure 2.5: Stemming process

Also there's another process called lemmatization which get the base form or the root form for different forms of a word with a little differences from stemming. The basic function of both is similar, but in stemming the root form of a word is obtained without consideration of the part of speech or the position of the word in the context. On the other hand lemmatization cares about the part of speech(POS). We can represent the stemming and lemmatization processes as black box functions with different arguments :

- stemming: stem(word);
- lemmatization: lemma(word,pos);

in the previous representation the difference is clear which is pos argument for lemmatization process.

The part of speech can be defined as an explanation of how a word is used in a sentence or the role of a word in a sentence. There are eight parts of speech, they are: nouns, pronouns, adjectives, verbs, adverbs, prepositions, conjunctions, interjections.

2.3.1 Stemming

In this study we only consider about Porter stemmer, it's an stemming algorithm was published in 1980. This stemmer is very widely used in information retrieval field. Porter stemmer uses a sequence of suffix-stripping rules. Figure 2.6 shows some of porter rules.

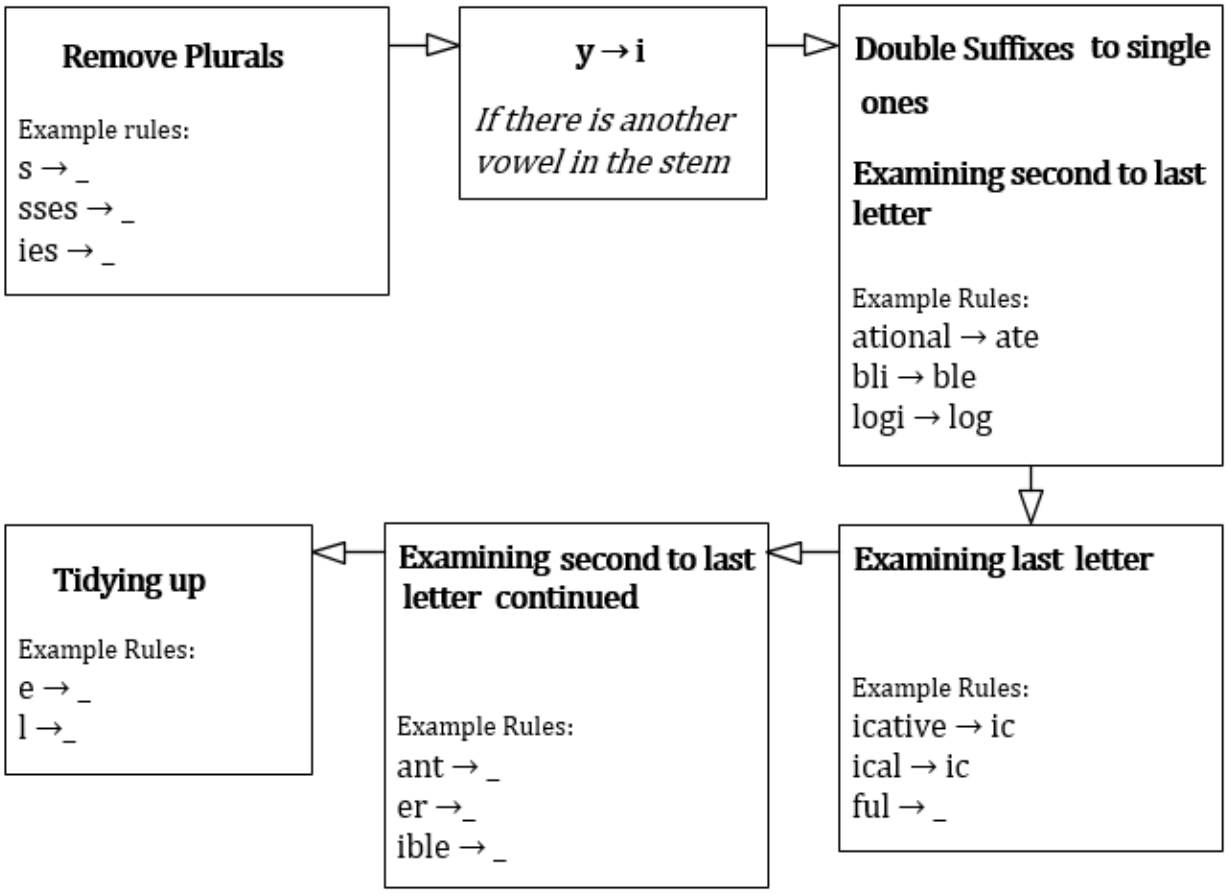

Figure 2.6: Porter algorithm rules

2.3.2 Lemmatization

Unlike stemming lemmatization need process need to define the POS of the word to get the root form of it. Also the lemmatization can mention to the same form of synonym words like adjectives with a similar meaning such as better, good, best. Lemmatization implementation will need to a dictionary. For sure, lemmatization process is more complex than stemming and typically is more efficient. In this study we will consider about the lemmatization tool for Stanford CoreNLP software for natural language processing. We won't dive into the details of lemmatization implementation, rather that we will use the lemmatization tool as a black box.

2.4 Feature Selection

Feature selection is the process of selecting a subset of the features/terms from the vocabulary according to some statistical measure. Figure 2.7 shows pseudo code for feature selection process.

The purpose of feature selection process is to reduce the features in the vocabulary and selecting only the most effective features by choosing the highest rank features. Each feature will be ranked using a function of

statistical measure, in this study will consider only to statistical measures, they are: Chi-square, and information gain.

```
SELECTFEATURES(𝔻, c, k)
1  V ← EXTRACTVOCABULARY(𝔻)
2  L ← []
3  for each t ∈ V
4  do A(t, c) ← COMPUTEFEATUREUTILITY(𝔻, t, c)
5     APPEND(L, ⟨A(t, c), t⟩)
6  return FEATURESWITHLARGESTVALUES(L, k)
```

Figure 2.7: Feature selection pseudo code

The process of feature selection usually done using selecting fixed number k that represents the number of selected features. To achieve the goal of selecting subset with size k from the vocabulary V with a set of documents(corpus) D, we will follow the pseudo code in Figure 2.7.

Firstly Vocabulary V should be extracted from the set of documents D, and then an empty list L should be created. For the term t in the vocabulary a statistical measure should be computed for it given a class to calculate how much does this term correlate to a class c and then add the term with its rank to the list L and then repeat the same procedure for each term in V. finally, the highest k features in L which they are ordered by the statistical rank will be selected.

2.4.1 Chi-Square

Chi-square is the first statistical ranking function we will deal with through the study.

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

Equation 2.3: Chi-square equation

In statistics, the chi-square test is applied to test the independence of two events, two events A and B are defined to be independent if P(AB) = P(A)P(B). In feature selection, the two events are occurrence of the term and occurrence of the class. We then rank terms with respect to Equation 2.3.

In equation 2.3, e_t define the existence of the term t within a document d by two binary values 0 and 1 if value 0 means that the term doesn't exist, value 1 means that the term exists within the document. e_c define the membership of a document d within a class c, value 1 means that the document d has a class label c and value 0 means that the class of document d isn't c. N is the observed frequency in D and e the expected frequency. For example, E_{11} is the expected frequency of t and c occurring together in a document.

Here's an example for applying chi-square test to calculate chi-square(D,t,C) ,where D is a set of documents t is a term and C is a set of classes, given the following:

- $D = \{d_1,d_2,d_3\}$;
- $t = t_1$;
- $C = \{c_1,c_2\}$;

Figure 2.8 shows the document term matrix for the D with respect to t.

	D	t1	C
1	d1	1	c1
2	d2	0	c1
3	d3	1	c2

Figure 2.8: Document Term Matrix for chi-square example

Firstly, we will calculate $P(t_1=0)$ 1/3 and $P(t_2=0)$ = 2/3 and then construct which is called contingency table that contains information for each class in the same table, but for simplicity we will construct a table for each class separately such as in Figure 2.9 that shows the table for class c_1 and Figure 2.10 that shows the table for class c_2.

C1	0	1
Observed	1	1
Expected	1/3	2/3

Figure 2.9: The table of the first class

C2	0	1
Observed	0	1
Expected	0	2/3

Figure 2.10: The table of the second class

The expectation for each value is calculated as in Equation 2.4.

$$E = O(a) * P(a)$$

Equation 2.4: The table of the second class

Where a in Equation 2.4 is the possible value of the term, in this case a could be only 1 or 0 as discussed.

$$x^2 = \sum \frac{(Observed - excpected)^2}{expected}$$

Equation 2.5: chi-square equation for the table version

The substitution of the example will be $x^2 = (1-1/3)^2 / (1/3) + (1-2/3)^2 / (2/3) + (0-0)^2 / 0 + (1-2/3)^2 / (2/3)$.

2.4.2 Information gain

$$IG(t) = -\sum_{i=1}^{m} P(c_i)\log P(c_i) + P(t)\sum_{i=1}^{m} P(c_i \mid t)\log P(c_i \mid t) + P(\bar{t})\sum_{i=1}^{m} P(c_i \mid \bar{t})\log P(c_i \mid \bar{t})$$

Equation 2.6: Information gain equation

We can apply information gain information on the previous example of chi-square, given D, C, t we can calculate the rank of t according to information gain using the formula in the Equation 2.6.

2.5 Learning Algorithms

In chapter 1, we discussed the concept of classifier and the purpose of using it and what inputs does it take but we didn't discuss how learning algorithm works or their mechanisms. Actually, there is a different models which was used to developing classification algorithms. Usually, they are categorized into parametric and nonparametric models. Parametric models include logistic regression, linear discriminant analysis, Perceptron, and simple neural networks.

On the other hand, nonparametric models include K-nearest neighbors, Decision tree, and Support vector machine. The main difference between two types of models is the parametric models summarize the data with a set of parameters of fixed size, while the nonparametric models don't make assumptions about the form of the mapping function. Nonparametric models are free to fit the data only using the training dataset.

In this study, only we will consider four classifiers. They are Naïve Bayes, Decision tree(ID3), Support vector machine, Random forest. It's known about these classifiers that they are efficient for text classification.

2.5.1 Naïve Bayes

Naïve Bayes is the best known probabilistic model, which is based on probability theory and conditional probability rules. The algorithm is based on applying Bayes' theorem with an assumption of the independency between the features. Historically, Naïve bayes is the first classifier is used for text classification purpose. The algorithm is very efficient in terms of complexity, since it takes linear time.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Equation 2.7: Bayes' rule

Everything starts with the bayes' rule, bayes' rule describes the probability of an event based on prior knowledge. This rule is very inspiring idea for future prediction tasks based on past events. Technically, the theorem is derived from total probability.

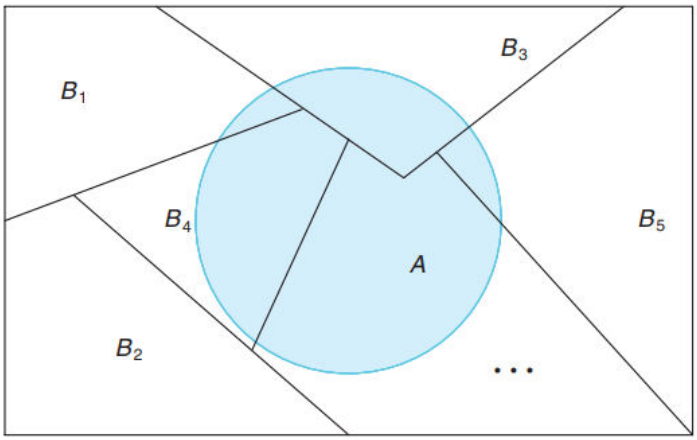

Figure 2.11: a portioning sample space

Total probability theorem states that if the set of events constitute a partition of the sample space such as what's Figure 2.11 shows. Then we can calculate the probability of any event A based on the provided information about the set of events that constitute a partition of the sample space.

$$P(A) = \sum_{i=1}^{k} P(B_i \cap A) = \sum_{i=1}^{k} P(B_i)P(A|B_i)$$

Equation 2.8: total probability theorem

Instead of asking about P(A), we asking about $P(B_r|A)$, simply this is the Bayes' rule.

$$P(B_r|A) = \frac{P(B_r \cap A)}{\sum_{i=1}^{k} P(B_i \cap A)} = \frac{P(B_r)P(A|B_r)}{\sum_{i=1}^{k} P(B_i)P(A|B_i)}$$

Equation 2.9: Bayes' rule derived from total probability

Here's an example of applying Naïve Bayes classifier on document term matrix. Figure 2.12 shows the document term matrix of our example.

D	term1	term2	class
d1	2	1	yes
d2	1	1	no
d3	1	0	yes

Figure 2.12: Document Term matrix for Bayes classifier example

Given a vector x = (1,1) we can predict the class of x by applying Bayes' formula such that P(class=yes|x), P(class=no|x). The class with highest probability should be the class of the observation. $P(class = yes|x) = P(class = yes) \times P(term1 = 1|class = yes) \times P(term2 = 1|class = yes) = \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2}$.

2.5.2 Decision Tree

Now, we will deal with Decision tree classifier with ID3 algorithm for constructing the decision tree. ID3 uses information gain and entropy as feature measure. Entropy is derived from information theory based on work by Claude Shannon. Firstly, we will discuss the structure of the Decision tree and then we will dive into the construction algorithm. Figure 2.13 shows an example of decision tree.

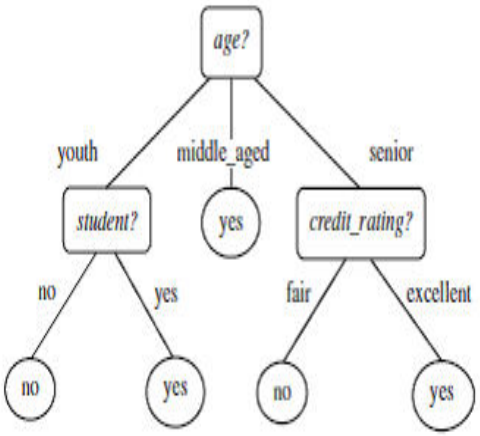

Figure 2.13: example of decision tree

First element in the decision tree is the root, root is a node. In decision tree, each node represent a feature. Each node in decision tree has branches, those branches represent the possible values for the feature. Each path should end with a leaf which a represent a class. In the Figure 2.13 yes and no represent two possible classes(leafs). Age, student, and

credit_rating represent features(Nodes). Youth, middle_aged, senior, no, yes, fair, and excellent represent branches.

Constructing decision tree according to ID3 should be done using entropy and information gain as measures for features and classes, firstly we should define what does entropy mean? Entropy is a measure of uncertainty, the value of the entropy is computed for a group of classes. If the sample which is computed completely homogeneous entropy will be zero. Homogenous means that all the sample elements have the same class. And if the sample is equally divided, the entropy value will be one. Information gain for a feature is the measure of how this feature affect in the class variation. The highest information gain value will be the feature which has the most effect on the class variation.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Equation 2.10

In equation 2.10, The expected information needed to classify a tuple in D is given by info(D), where D is a set of observations.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

Equation 2.11

In Equation 2.11, A is the feature which is tested given a set of observations D to examine the expected information required to classify a tuple from D based on the partitioning by feature A. The smaller the expected information required, the greater the purity of the partitions.

$$Gain(A) = Info(D) - Info_A(D).$$

Equation 2.12

Gain(A) tells us how much would be obtained by branching by feature A.

Decision tree construction steps will be as follows:

1. For each node selection, the selected node should be highest Gain(A) value.
2. After each node selection, the branching is done with all possible values of the selected feature in its partition D.
3. Repeat node selection for each branch in each path until the partition D for each path become labeled with the same class.
4. If the partition D for a path p labeled with the same class, then create a leaf node with the class of D.

2.5.3 Random Forest

Random forest is a combination of learning models consisting of a bagging of N decision tree models. We already discussed Decision Tree model so now we will discuss the idea of bagging that is considered the basic for random forest mode. The intuition behind Random forest can be described by bootstrap sampling which states that a given each training set, we can create new training sets by random sampling the original set $\overline{N} \leq N$ times with replacement. The concept of bagging can be explained as a set of classification models trained independently such that each model will be trained on unique sample from the original training set and then these models will be used in a combination for prediction. The predicted class of a random forest classification model is the class that has the maximum votes from the different decision trees. Simply, we can consider Random forest technique as an optimized model for decision tree.

2.5.4 Support vector machines

The intuition behind Support vector machine models can be visualized by Figure 2.14.

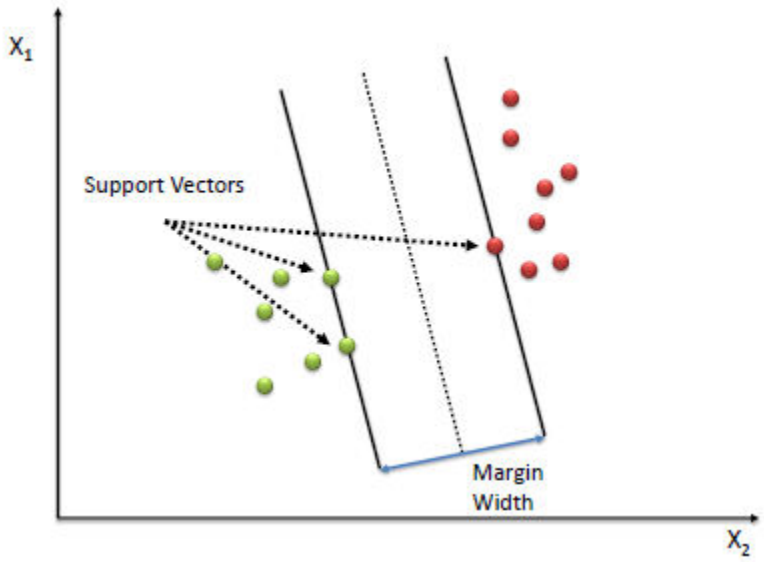

Figure 2.14: Applying SVM on two dimensional space

Firstly, we can define a Hyperplane as a subspace whose dimension is N-1, Where N is the number of dimensions in the feature space. This means that any line in two dimensional space is considered as a Hyperplane and any plane in 3-dimensional space is considered as a Hyperplane , for sure with respect to their spaces. The idea of SVM is to find the optimum Hyperplane that separates between two classes instances on the feature space. The core of SVM algorithm is to find the equation of this optimum Hyperplane, for this purpose many of optimization algorithms such as gradient descent are used to find the optimum Hyperplane. The optimum Hyperplane is a Hyperlane that separates between two classes instances with the maximum margin width between classes.

2.6 Proposed approaches

So far, we provided the theortical aspects behind these techniques. Figure 2.15, Figure 2.16, Figure 2.17, and Figure 2.18 summarize different techniques that we discussed for each phase.

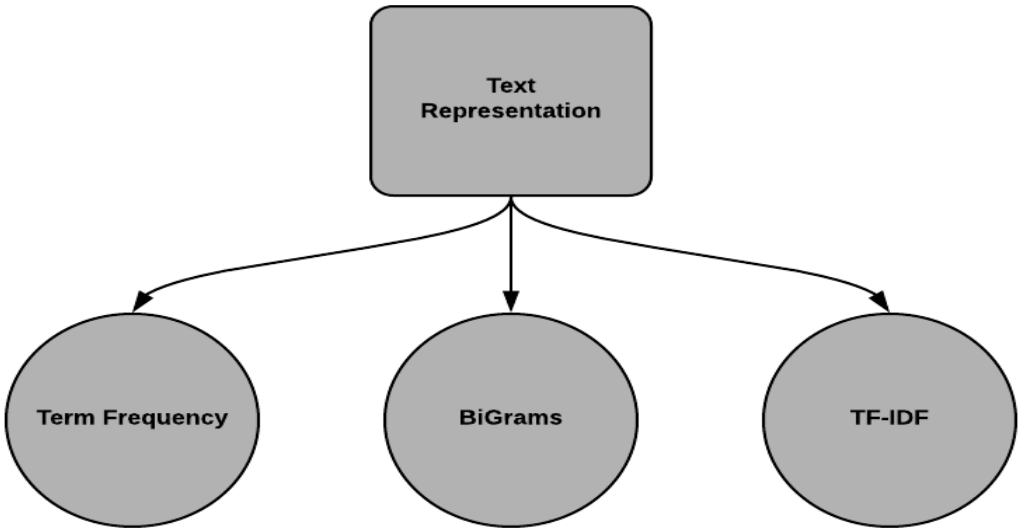

Figure 2.15: Text representation techniques

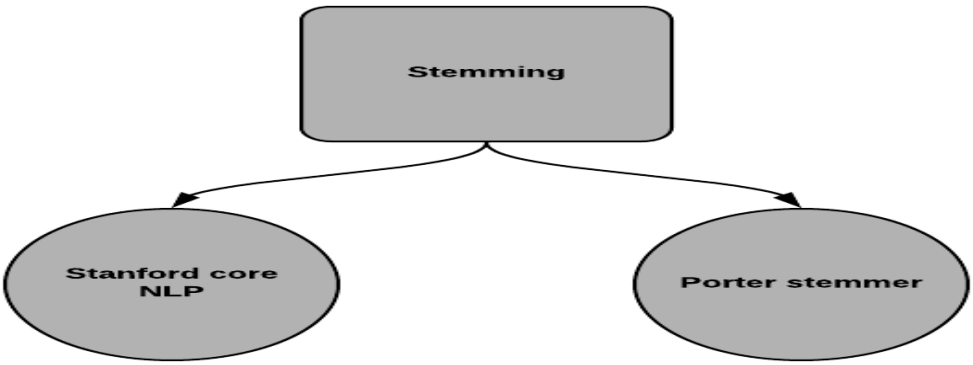

Figure 2.16: Stemming techniques

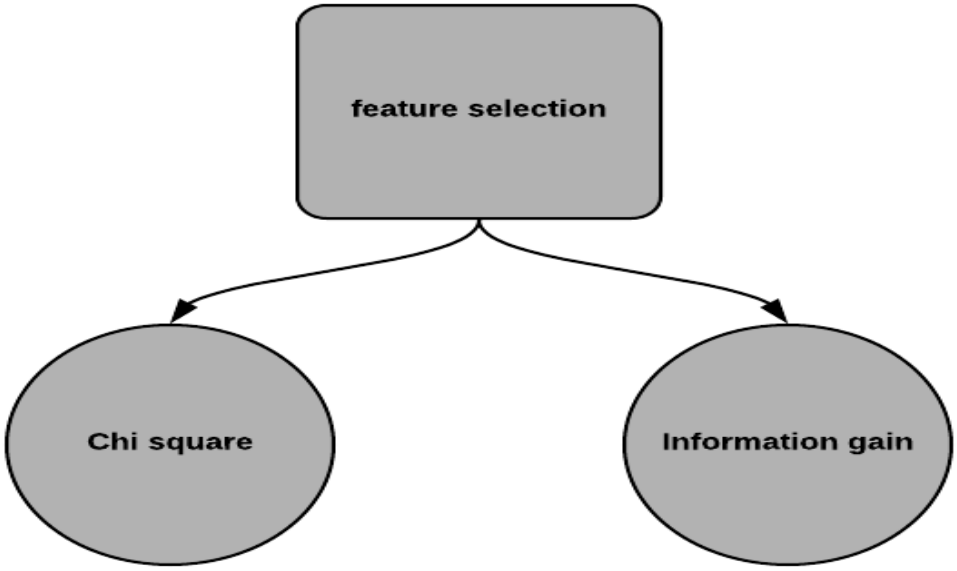

Figure 2.17: Feature selection techniques

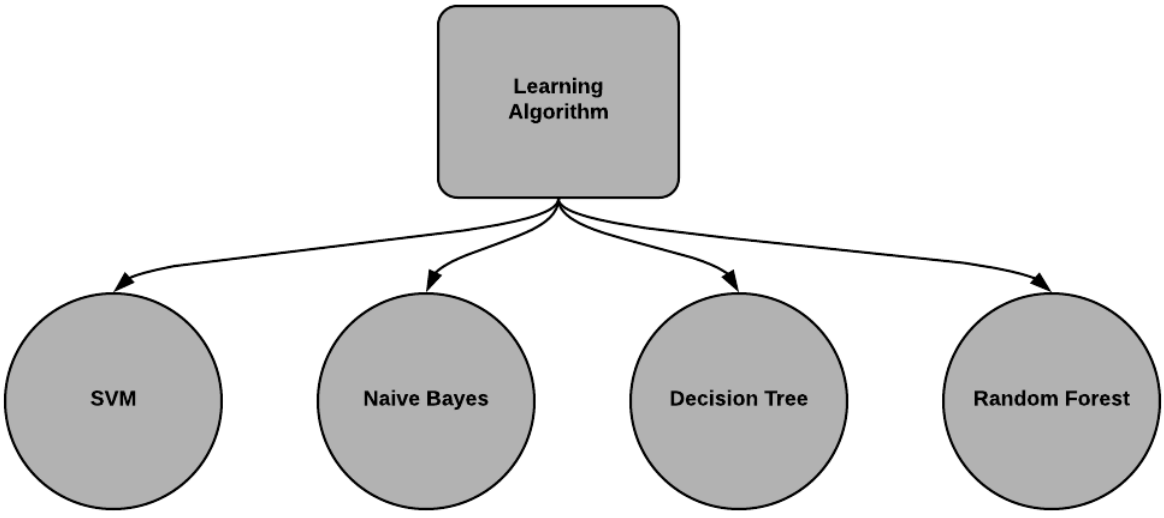

Figure 2.18: Classification algorithms

We aim to test various combinations of these techniques to examine which one with the highest accuracy with respect to our expirement. Each of our proposed aprroch which represent one combination of these techniques. Text representation, stemming, feature Selection, classifiers are the major factors of text classification accuracy, thus we propsed some approaches to be applied by controlled expirements on the selected dataset.

Following figures represent our proposed apporach that will be applied as expirements in the next chapter. In the following figures the name of classifiers are abbriviated such that Naïve Bayes(NB), Decision tree(DT), Support vector machines(SVM), Random forest(RF).

In each combination, we pay attention to make the results meaninful and make the impact of each factor is clear and can be observed to be interpreted seperately and with combinations. We will name each apporach by a letter, they will be ordered alphabetically.

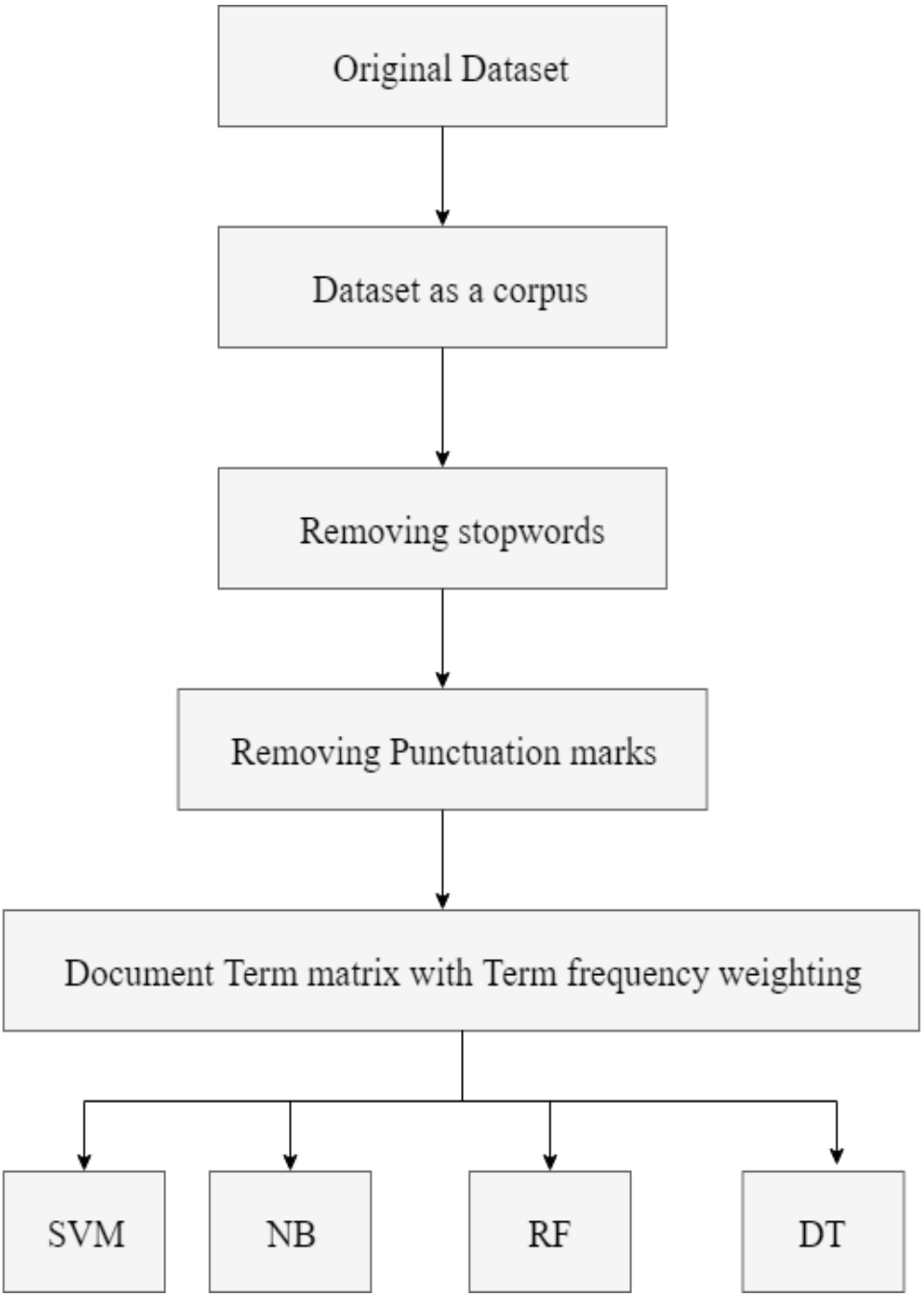

Figure 2.19: Approach

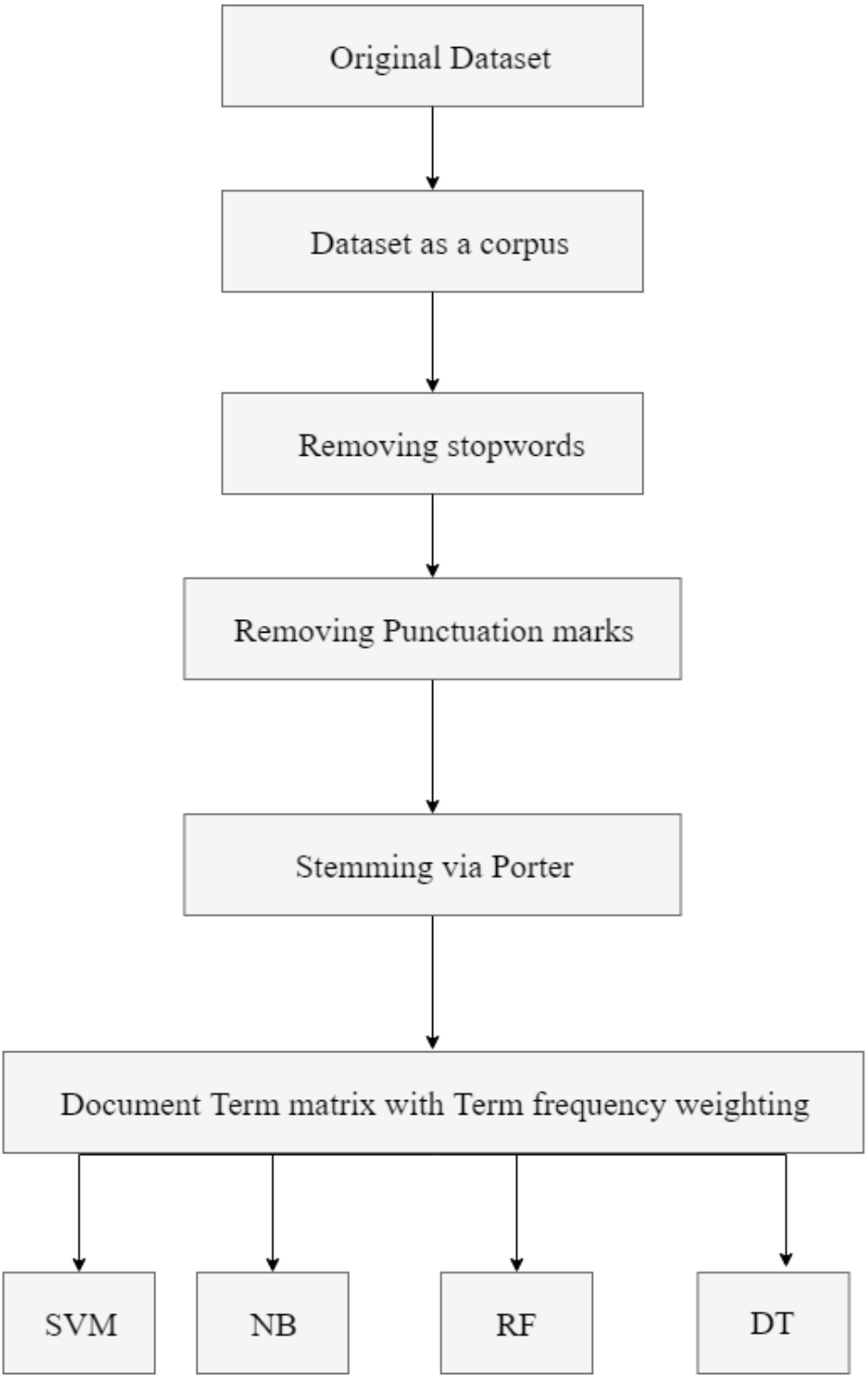

Figure 2.20: Approach B

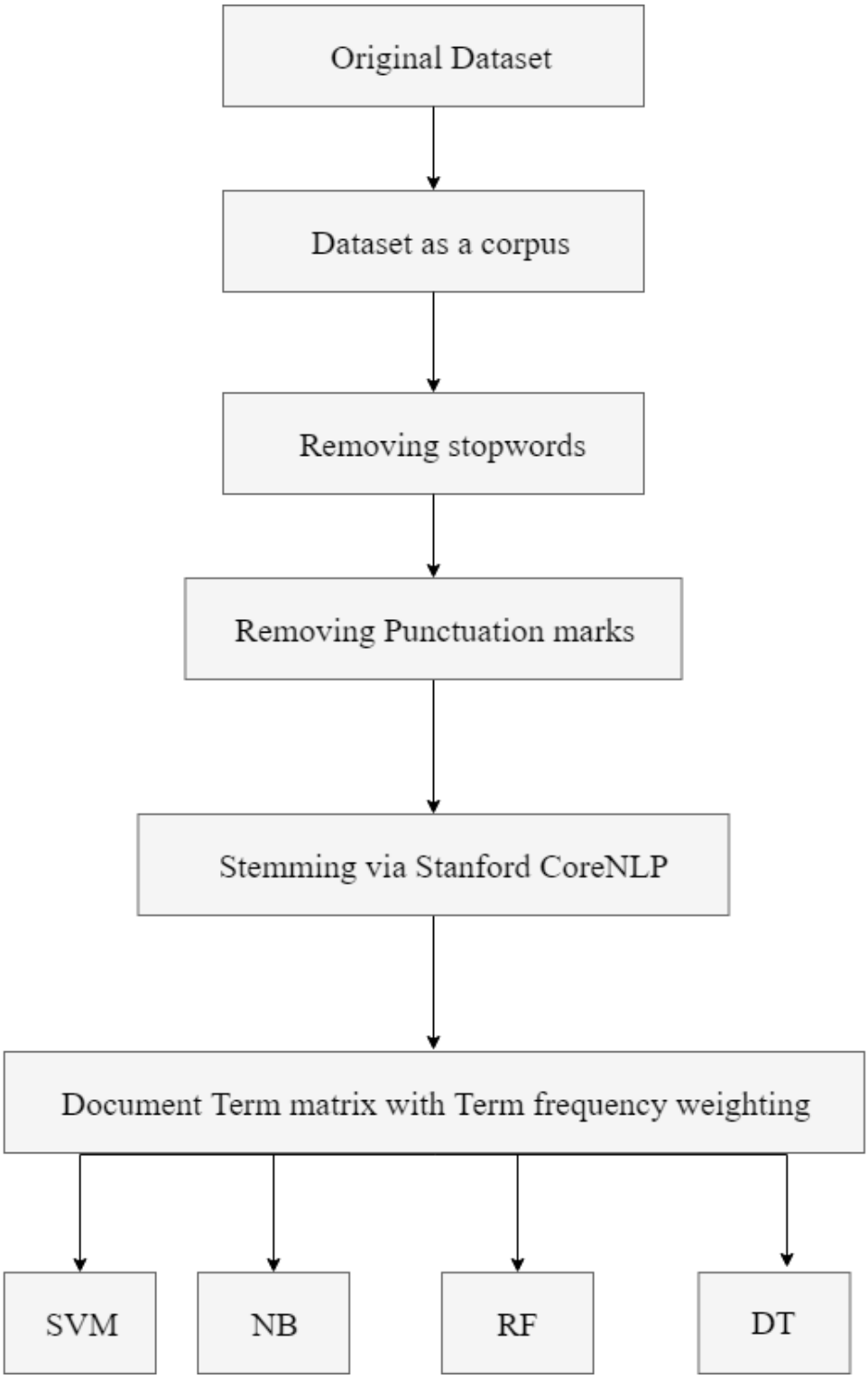

Figure 2.21: Approach C

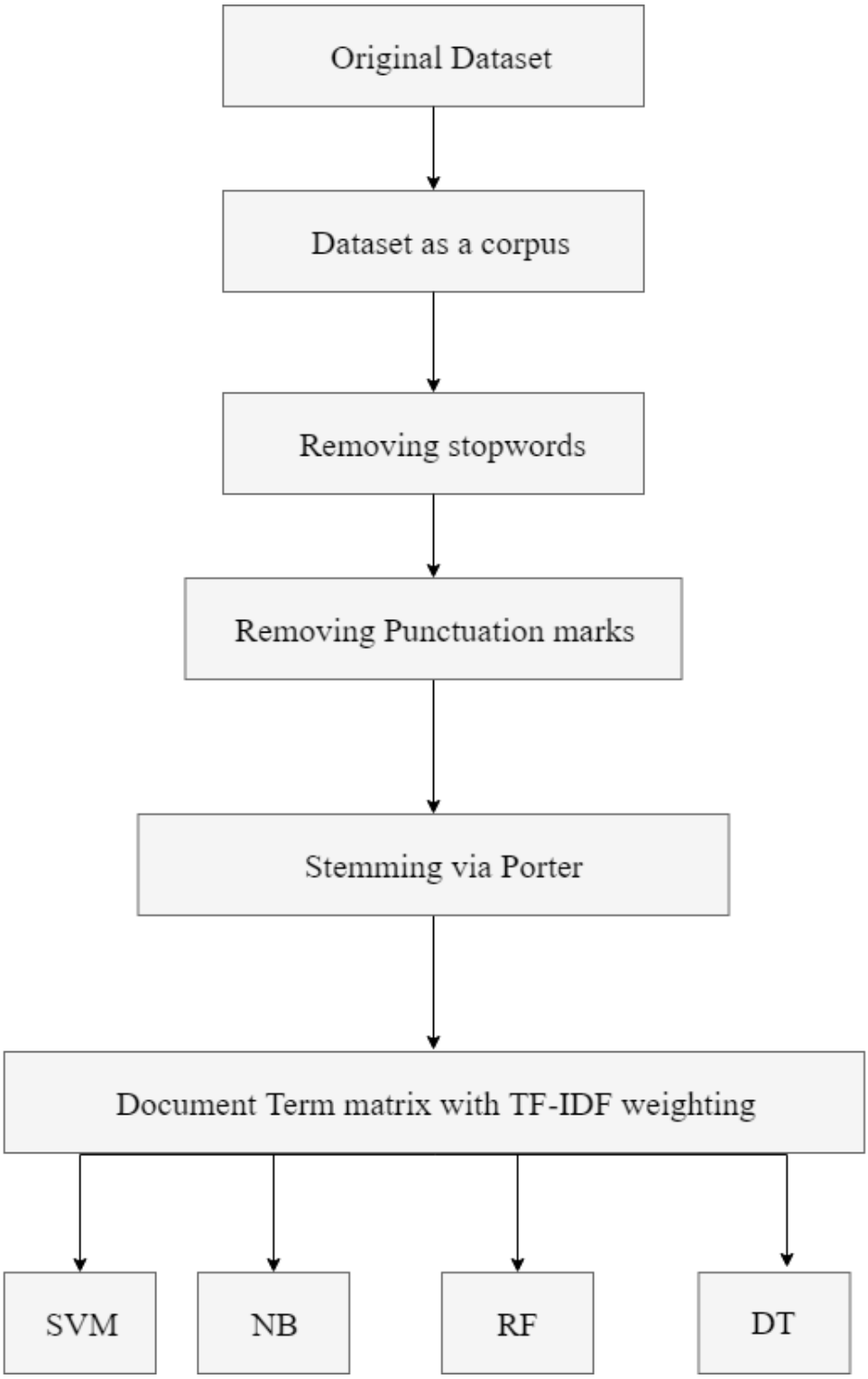

Figure 2.22: Approach D

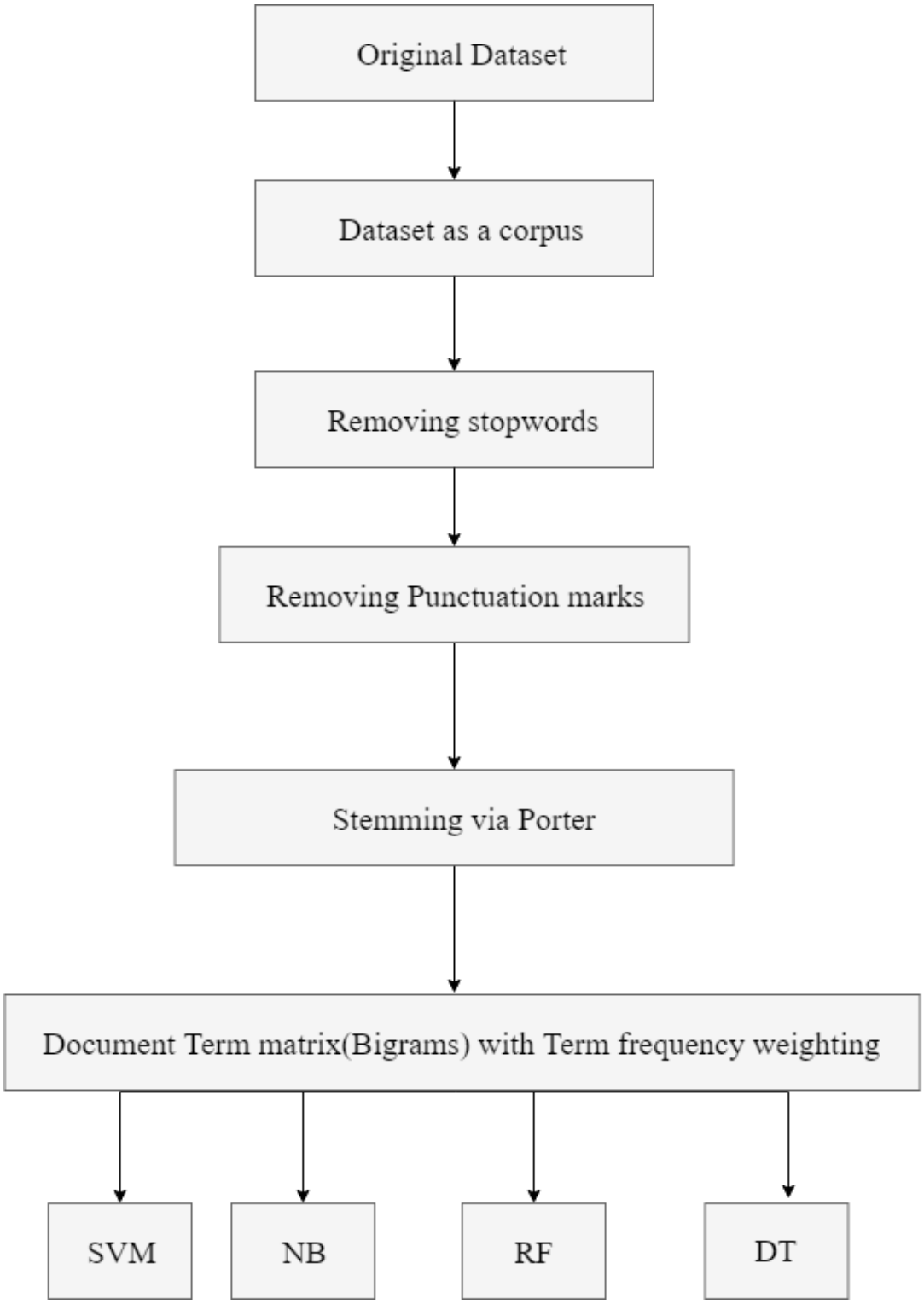

Figure 2.23: Approach E

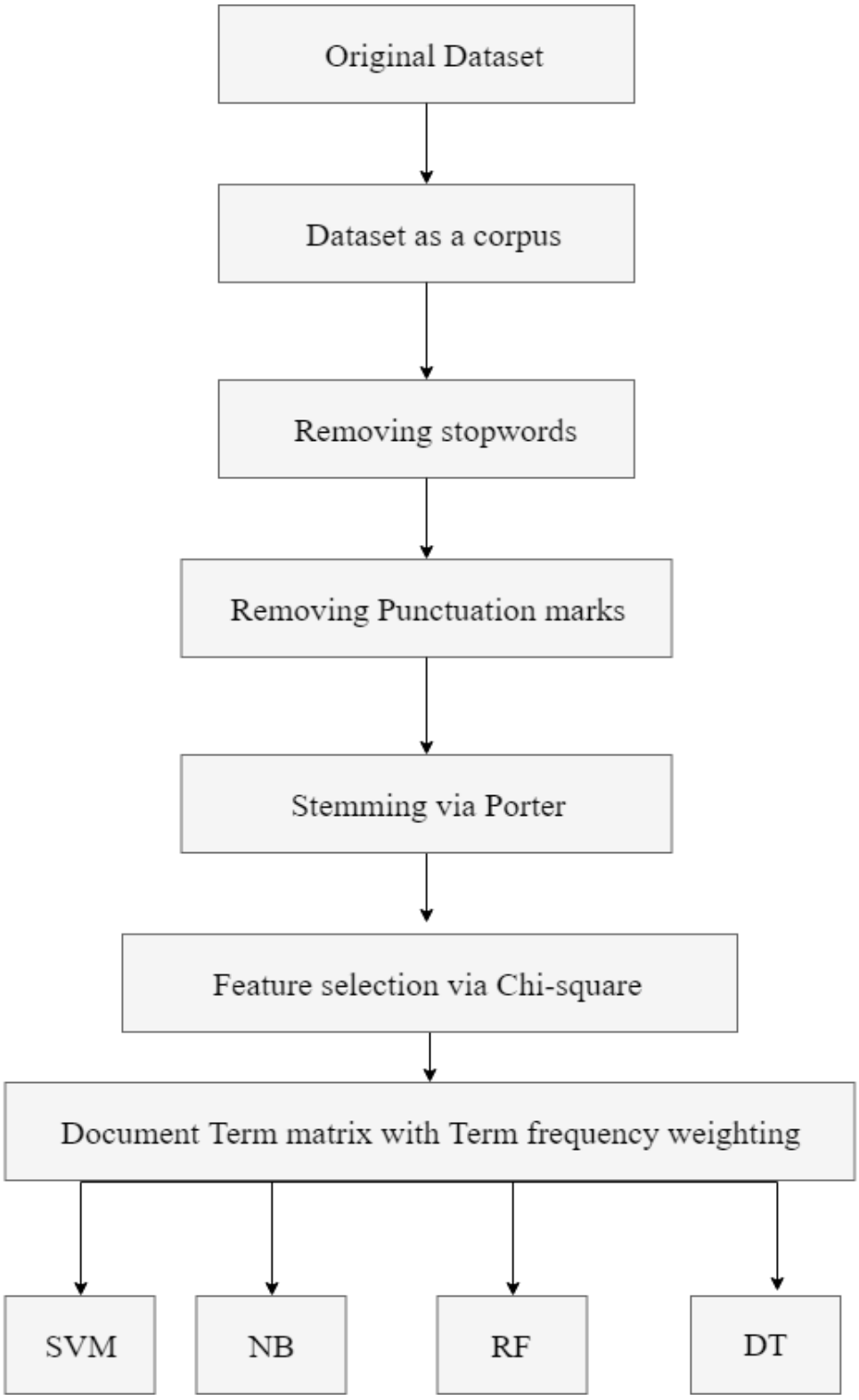

Figure 2.24: Approach F

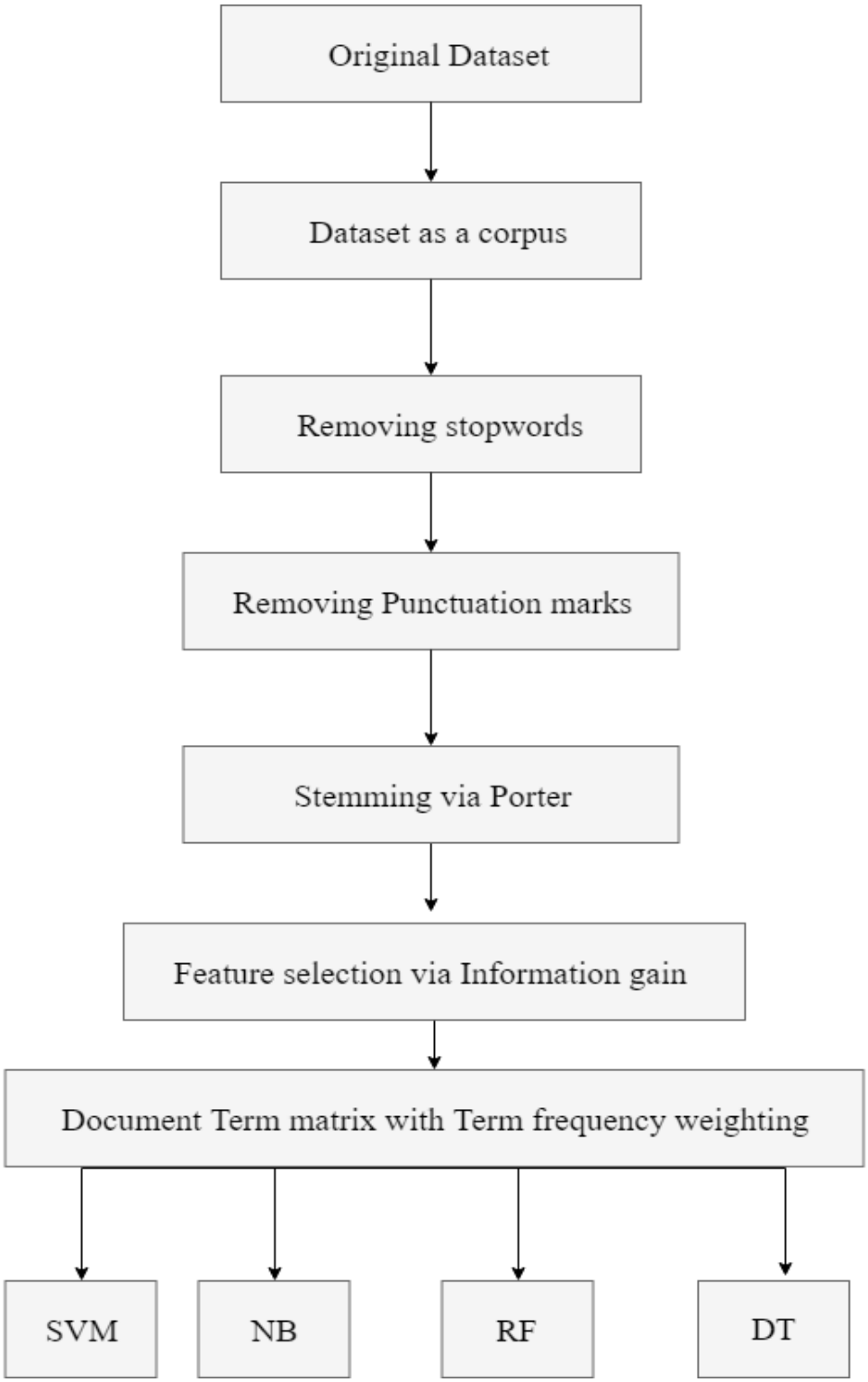

Figure 2.25: Approach

Chapter 3

Code and implementation

In this chapter, we illustrate how we applied the experiments on the selected dataset using the seven approaches that we stated at last section on the previous chapter.

All implementations were done using Java programming language and R programming language. We used Java mostly for text processing and stemming tasks and used R for classification purposes, since R provides a lot of packages that implement classification algorithms and many of machine learning techniques and data visualization for statistical purposes. Through this chapter we will show samples of our implementations and provide an explanation for each part.

3.1 Preprocessing

Tasks such as Removing stopwords, Removing punctuation marks, and dataset transformation are called preprocessing operations. We deal with a dataset organized as text files such that each class is folder contains a collection of text files. Figure 3.1 and Figure 3.2 show how the dataset is organized.

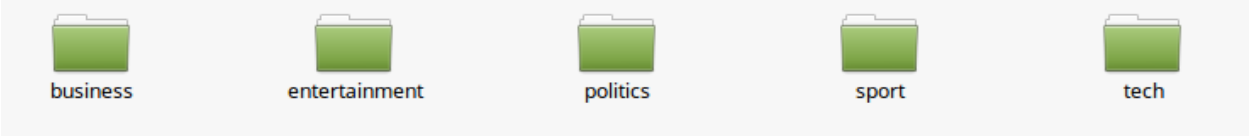

Figure 3.1: Classes organized as folders

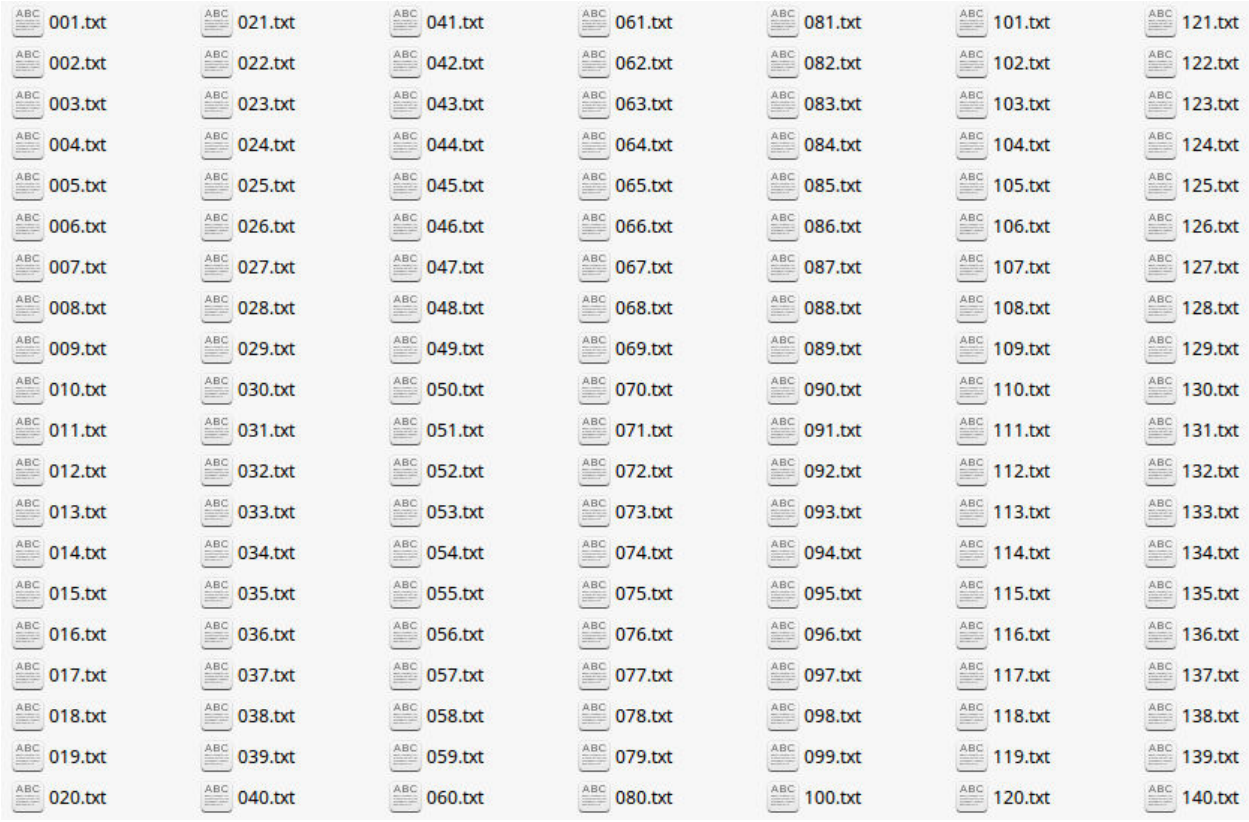

Figure 3.2: class instances organized as text files

For such a dataset, we need to use a framework that deals with files in appropriate form and facilitate required operations. We implement some functions to be our framework for this purpose. Figure 3.3 and Figure 3.4 show two basic function for reading and writing text files.

```
public static ArrayList<String> loadFile(String Dir) {

    ArrayList<String> lines = new ArrayList<String>();
    try {
        Scanner Reading = new Scanner(new File(Dir));
        while (Reading.hasNext()) {
            lines.add(Reading.nextLine());
        }
    } catch (FileNotFoundException e) {
        System.out.println("Error , Because of loadFile" + e.getMessage());
        System.exit(0);
    }
    return (lines);

}
```

Figure 3.3: Java implementation to read from file

```
private void saveFile(ArrayList<String> words, String dir) {
    try {
        FileWriter file = new FileWriter(new File(dir), true);
        for (String word : words) {
            file.write(word);
            file.write("\n");
        }
        file.flush();
        file.close();
    } catch (Exception e) {
        System.out.println("Error in saveFile" + e.getMessage());
    }
}
```

Figure 3.4: Java implementation to save file

loadFile facilitate dealing with files by returning the file content as an array list of strings where each line in the file is an element in the array. The function has one parameter which is the file directory.

In contrast, saveFile save array list of lines in a text file which is given by the parameter dir. loadFile and saveFile is the basic framework setup to deal with the dataset, they will be used mostly in all other functions.

```
private void getFilesList() {
    File folder = new File(Dir);
    ArrayList<String> All = new ArrayList<String>();
    for (final File fileEntry : folder.listFiles()) {
        if (fileEntry.isDirectory()) {
            continue;
        } else {
            All.add(fileEntry.getName());
        }
    }
    this.FilesList = All;
    String[] list = new String[FilesList.size()];
    list = FilesList.toArray(list);
    Arrays.sort(list);
    FilesList = new ArrayList<>(Arrays.asList(list));
}
```

Figure 3.5: function to load a folder's files names

In order to load the all dataset documents, we should load files names as a list to be read later by loadFile function. Figure 3.5 shows our implementation of a function that loads files names within a folder by its directory which is the class member Dir.

```
public void CollapseAsLines() {
    ArrayList<String> All = new ArrayList<String>();
    for (String file : this.FilesList) {
        ArrayList<String> F = Category.loadFile(Dir + "/" + file);
        String AllInLine = "";
        for (String line : F) {
            AllInLine = AllInLine + " " + line;
        }
        AllInLine = AllInLine.toLowerCase();
        AllInLine = Filtration.RemoveStopWords(AllInLine, StopWords);
        AllInLine = Filtration.Remove_punctuations(AllInLine);
        AllInLine = Filtration.RemoveStopWords(AllInLine, StopWords);
        All.add(AllInLine);
    }

    saveFile(All, Dir + "/" + "All.txt");
}
```

Figure 3.6: function to load a folder's files names

CollapseAsLines function, is the function that transforms all the dataset documents into a corpus, where each line represents a document. CollapseAsLines also removes stopwords and punctuation marks from the corpus by calling RemoveStopWords function and Remove_punctuations function. Figure 3.7 and Figure 3.8 show both of them.

Both functions take the document as string parameter and tokenize the document and start examine each word separately.

```
public static String Remove_punctuations(String Document) {
    String[] Splitted_Words = Document.split(" ");
    ArrayList<String> Clean_SplittedWords = new ArrayList<String>();
    for (String splitted_word : Splitted_Words) {
        String[] Words_temp = splitted_word.replaceAll("[^a-zA-Z ]", "").toLowerCase().split("\\s+");
        String CleanWord_temp = "";
        for (String CW : Words_temp) {
            if (CW.trim().length() == 0) {

                continue;
            }
            CleanWord_temp += CW.trim();
        }
        Clean_SplittedWords.add(CleanWord_temp.trim());
    }
    String Clean_Document = "";
    for (String cleanword : Clean_SplittedWords) {
        Clean_Document = Clean_Document.trim() + " " + cleanword;
    }
    return (Clean_Document);
}
```

Figure 3.7: function that removes punctuation marks and symbols

```
public static String RemoveStopWords(String Document, ArrayList<String> StopWords) {
    String[] Words_temp = Document.split(" ");
    ArrayList<String> Words = new ArrayList<>(Arrays.asList(Words_temp));
    int Size = Words.size();
    for (int i = 0; i < Size; i++) {
        for(String StopWord:StopWords){
          if(Words.get(i).trim().toLowerCase().equals(StopWord)){
              Words.remove(i);
              i--;
              Size = Words.size();
              break;
          }
        }
    }

    String Clean_Document = "";
    for(String word:Words){
        Clean_Document = Clean_Document.trim() + " " + word;
    }
    return(Clean_Document);
}
```

Figure 3.8: function that removes stopwords

Now, each class documents is stored as a text file which we call corpus. Dataset still need to some processing to be ready for classification. Now we will transform the corpus text into a CSV file such as in the Figure 3.9.

	Document	Class
2	glaxo aims high profit fall glaxosmithkline profits fall year bn bn europes	business
3	mcconnell drunk remark row scotlands minister told group high school pu	politics
4	lawyer attacks antiterror laws senior barrister resigned protest governmer	politics
5	sporting rivals extra time current slew sports games offers unparalleled c	tech
6	paraguay novel wins book prize novel set century paraguay won fiction p	entertainment
7	incredibles win animation awards incredibles movie beaten shrek main p	entertainment
8	oscars steer clear controversy oscars nominations list left controversial	entertainment
9	prince crowned top music earner prince earned pop star beating artists n	entertainment
10	bollywood dvd fraudster jailed major distributor pirated dvds bollywood fil	entertainment
11	rich pickings hitech thieves viruses trojans malicious programs net catch	tech
12	tories outlining policing plans local communities asked polls elect area p	politics
13	britons growing digitally obese gadget lovers hungry digital data carrying	tech
14	serena number serena williams moved places second rankings australian	sport
15	tories attack burglar uturns tory leader michael howard accused tony blai	politics
16	mansfield leyton orient secondhalf goal andy scott condemned mansfield	sport
17	cyber crime booms months dramatic growth security threat plague windo	tech
18	cabinet anger brown cash raid ministers unhappy plans whitehall cash co	politics
19	fears raised ballet future fewer children uk dainty footsteps dancers darc	entertainment
20	aids climate top davos agenda climate change fight aids leading list con	business
21	eastenders set remake plans create soap based bbcs eastenders reporte	entertainment
22	blair mayor apologise tony blair urged london mayor ken livingstone apol	politics
23	britannia members windfall britannia building society members receive pr	business
24	bmw drives record sales asia bmw forecast sales growth asia year regis	business
25	khodorkovsky ally denies charges close associate yukos boss mikhail k	business
26	mandelson warning bbc bbc steer demonising exdowning street media cl	politics

Figure 3.9: CSV form for the dataset

In CSV form, document is stored alongside its class. To transform corpus to this form we used R. Figure 2.10 shows the code that transforms the dataset as CSV form. This form facilitates classification process.

```
business_corpus = readLines("/home/fake/Desktop/Project/Dataset/bbc/business/All.txt")
entertainment_corpus = readLines("/home/fake/Desktop/Project/Dataset/bbc/entertainment/All.txt")
politics_corpus = readLines("/home/fake/Desktop/Project/Dataset/bbc/politics/All.txt")
sport_corpus = readLines("/home/fake/Desktop/Project/Dataset/bbc/sport/All.txt")
tech_corpus = readLines("/home/fake/Desktop/Project/Dataset/bbc/tech/All.txt")
business <- data.frame(business_corpus,rep("business",length(business_corpus)))
entertainment <- data.frame(entertainment_corpus,rep("entertainment",length(entertainment_corpus)))
politics <- data.frame(politics_corpus,rep("politics",length(politics_corpus)))
sport <- data.frame(sport_corpus,rep("sport",length(sport_corpus)))
tech <- data.frame(tech_corpus,rep("tech",length(tech_corpus)))
Columns <- c("Document","Class")
names(business)<- Columns
names(entertainment)<- Columns
names(politics)<- Columns
names(sport)<- Columns
names(tech)<- Columns
Dataset <- rbind(business,entertainment,politics,sport,tech)
set.seed(150)
sample <- sample.int(n=nrow(Dataset),size=floor(nrow(Dataset)))
Dataset <- Dataset[sample,]
write.csv(Dataset, file = "/home/fake/Desktop/Project/Dataset/Dataset_Orginal.csv")
```

Figure 3.10: R implementation to CSV form transformation

3.2 Stemming

As we discussed in the previous chapter, we will apply stemming process using Porter algorithm and will apply lemmatization process using Stanford CoreNLP software. For abbreviation we call both processes stemming since they are accomplish the same purpose.

We used Snowball's implementation for Porter algorithm, they provide an API for their libraries. We used this API to merge their functions with our implementation.

On the other hand, we used StanfordCoreNLP API to use their functions for lemmatization.

3.2.1 Porter Algorithm

```
private String CorpusDir;
private SnowballStemmer stemmer;
private String StemmedDir;

public Stemming(String Dir, String StemmedDir) throws Throwable {
    Class stemClass = Class.forName("org.tartarus.snowball.ext."
            + "porter" + "Stemmer");
    stemmer = (SnowballStemmer) stemClass.newInstance();
    CorpusDir = Dir;
    this.StemmedDir = StemmedDir;
}

public void StemmCorpus() throws Throwable {
    ArrayList<String> Lines = loadFile(CorpusDir);
    ArrayList<String> StemmedCorpus = new ArrayList<>();
    for (String Line : Lines) {
        ArrayList<String> StemmedWords = new ArrayList<>();
        String[] WordsTemp = Line.split(" ");
        for (String word : WordsTemp) {
            stemmer.setCurrent(word);
            stemmer.stem();
            StemmedWords.add(stemmer.getCurrent());
        }
        String All = "";
        for(String StemmedWord:StemmedWords){
            All = All.trim() + " " + StemmedWord;
        }
        StemmedCorpus.add(All);
    }
    saveFile(StemmedCorpus, StemmedDir);
}
```

Figure 3.11: Using Snowball API to apply Porter stemmer on each corpus

Figure 3.11 shows how we used Snowball API to use their functions to apply Porter stemmer on each corpus separately. Given a corpus directory, the function read the corpus as lines where each line is a document. Then separate among words by space and then using the Porter object to stem each word.

3.2.2 Stanford CoreNLP

```
public Lemmatizer() {
    Properties cfg;
    cfg = new Properties();
    cfg.put("annotators", "tokenize, ssplit, pos, lemma");
    this.pipeline = new StanfordCoreNLP(cfg);
}


public List<List<String>> lemmatize(String text) {
    List<List<String>> lines = new ArrayList<List<String>>();
    Annotation document = new Annotation(text);
    this.pipeline.annotate(document);

    List<CoreMap> sentences = document.get(SentencesAnnotation.class);
    for (CoreMap sentence : sentences) {
        List<String> words = new ArrayList<String>();
        for (CoreLabel token : sentence.get(TokensAnnotation.class)) {
            words.add(token.get(LemmaAnnotation.class));
        }
        lines.add(words);
    }
    return lines;
}
```

Figure 3.12: Using Stanford CoreNLP API to apply lemmatization on each document

Figure 3.12 shows how we used Stanford CoreNLP API to use their functions to apply lemmatization on each document. As we discussed in the previous chapter that the lemmatization process need to POS argument for each word. Thus it's important to keep the original document format since the Lemmatizer deals with sentences to identify the POS for each word, it doesn't deal with single words.

```
public void CollapseWithLemmatization() {
    ArrayList<String> Corpus = new ArrayList<String>();
    for (String file : FilesList) {
        ArrayList<String> Lines = loadFile(Dir + "/" + file);
        String all = "";
        for (String line : Lines) {
            if (line.length() > 0) {
                if (!line.substring(line.length() - 1).equals(".")) {
                    line = line + ".";
                }
            }
            all = all.trim() + " " + line;

        }
        Corpus.add(all);

    }

    Lemmatizer lemmatizer = new Lemmatizer();
    ArrayList<String> LemmatizedCorpus = new ArrayList<>();
    for (String C : Corpus) {
        List<List<String>> document = lemmatizer.lemmatize(C);
        String documentAsText = "";
        for (List<String> list : document) {
            for (String l : list) {
                documentAsText = documentAsText.trim() + " " + l;
            }
        }
        LemmatizedCorpus.add(documentAsText);
    }
    ArrayList<String> Clean_Corpus = new ArrayList<>();
    for (String Document : LemmatizedCorpus) {
        Document = Filtration.RemoveStopWords(Document, StopWords);
        Document = Filtration.Remove_punctuations(Document);
        Document = Filtration.RemoveStopWords(Document, StopWords);
        Clean_Corpus.add(Document);
    }

    saveFile(Clean_Corpus, Dir + "/" + "Lemmatized.txt");

}
```

Figure 3.13: our implementation to a function that lemmatize the whole corpus given a class folder

3.3 Text representation

So far we dealt theoretically with the idea of text representation for classification process. Now we will deal with the text representation in terms of programming. Here's how we used R programming language to implement text representation techniques.

The following figures illustrate the process.

```
4  Dataset <- read.csv("/home/fake/Desktop/Project/Dataset/Original CSV/Dataset_Orginal.csv")
```

Figure 3.14: loading the CSV file by the directory

```
6  set.seed(40)
7  sample <- sample.int(n=nrow(Dataset),size=floor(.7*nrow(Dataset)))
8  train <- Dataset[sample,]
9  test <- Dataset[-sample,]
```

Figure 3.15: Splitting the dataset into training set 70% and test one 30%

```
12  corpus <- Corpus(VectorSource(train$Document))
13  Document_Term_Matrix <- DocumentTermMatrix(corpus)
14  Document_Term_Matrix <- removeSparseTerms(Document_Term_Matrix,.99)
15  Trainset_Matrix <- as.matrix(Document_Term_Matrix)
```

Figure 3.16: constructing document term matrix with term frequency weighting by default

```
18  Trainset_Matrix <- as.data.frame(Trainset_Matrix)
19  Trainset_Matrix <- cbind(Trainset_Matrix,train$Class)
20  colnames(TRAINSET)[ncol(TRAINSET)] <- "y"
```

Figure 3.17: adding the class vector in the observation matrix

3.3.1 TF-IDF

We also dealt with the concept of TF-IDF theoretically in the previous chapter, Here's an our implementation to a function in R that construct Document term matrix with TF-IDF weighting.

```
preProcess_TFIDF <-
  function(row.data,
           stopword.dir,
           BagOfWord ,
           boolstemm) {
    row.data <- iconv(row.data, "WINDOWS-1252", "UTF-8")
    stopwordlist <- readLines(stopword.dir)
    train_corpus <- Corpus(VectorSource(row.data))
    train_corpus <- tm_map(train_corpus, content_transformer(tolower))
    train_corpus <- tm_map(train_corpus, removeNumbers)
    train_corpus <-
      tm_map(train_corpus, removeWords, stopwords("english"))
    train_corpus <- tm_map(train_corpus, removeWords, stopwordlist)
    train_corpus <- tm_map(train_corpus, removePunctuation)
    train_corpus <- tm_map(train_corpus, stripWhitespace)
    if (boolstemm) {
      train_corpus <- tm_map(train_corpus, stemDocument, language = "english")
    }

    DTM <- DocumentTermMatrix(
      train_corpus,
      control = list(
        tolower = T ,
        removeNumbers = T ,
        removePunctuation = T ,
        stopwords = T
        ,
        stripWhitespace = T ,
        dictionary = BagOfWord,
        weighting = weightTfIdf
      )
    )
    print("DTM DONE !!")
    print(DTM)
    return(DTM)
  }
```

Figure 3.18: Constructing DTM with TF-IDF weighting

3.3.2 N-grams

Here's our implementation of a function that accomplish N-grams representation of feature space. The function can apply unigrams, bigrams, or trigrams. But we apply only bigrams on our experiments.

```
preProcess.N_grams <-function(row.data,stopword.dir, BagOfWord, boolstemm  ,MIN_ngram , MAX_ngram) {
    row.data <- iconv(row.data, "WINDOWS-1252", "UTF-8")
    stopwordlist <- readLines(stopword.dir)
    train_corpus <- VCorpus(VectorSource(row.data))
    train_corpus <- tm_map(train_corpus, content_transformer(tolower))
    train_corpus <- tm_map(train_corpus, removeNumbers)
    train_corpus <-tm_map(train_corpus, removeWords, stopwords("english"))
    train_corpus <- tm_map(train_corpus, removeWords, stopwordlist)
    train_corpus <- tm_map(train_corpus, removePunctuation)
    train_corpus <- tm_map(train_corpus, stripWhitespace)
    if (boolstemm) {
      train_corpus <- tm_map(train_corpus, stemDocument, language = "english")
    }
    Tokenizer <-
      function(x)
        NGramTokenizer(x, Weka_control(min = MIN_ngram, max = MAX_ngram))
    DTM <- DocumentTermMatrix(
      train_corpus,
      control = list(
        tolower = T ,
        removeNumbers = T ,
        removePunctuation = T ,
        stopwords = T
        ,
        stripWhitespace = T ,
        dictionary = BagOfWord,
        weighting = weightTf,
        tokenize = Tokenizer
      )
    )
    print("DTM DONE !!")
    print(DTM)
    return(DTM)
  }
```

Figure 3.19: Constructing a bigrams feature space

3.4 Feature Selection

Now, We will show how we apply the two feature selection techniques : Chi-square, and Information gain in R.

3.4.1 Chi-square

```
train_chi <- as.data.frame(train_matrix)
train_chi$y <- train_dataset$class
chi_square <- chi.squared(y~.,data = train_chi  )
subset <- cutoff.k(chi_square , 300)
train_data_chi <- data.frame(y=train_chi$y , x = train_chi[subset])
test1_data_model <- as.data.frame(test1_matrix)
test1_data_model <-  data.frame(y=testdata$class , x = test1_data_model[subset])
```

Figure 3.20: selecting best 300 feature using Chi-square rank

3.4.2 Information gain

```
train_info <- as.data.frame(train_matrix)
train_info$y <- train_dataset$class
IG_weight <- information.gain(y~., data = train_info)

subset_info <- cutoff.k(IG_weight,800)
train_data_info <- data.frame(y =train_info$y, x= train_info[subset_info] )
test1_data_model <- as.data.frame(test1_matrix)
test1_data_model <-  data.frame(y=testdata$class , x = test1_data_model[subset_info])
```

Figure 3.21: selecting best 800 features using information gain ranking

3.5 Classifiers

R packages provide implementation of most known classification algorithms, so no need to implement algorithms from scratch. As we discussed before, classifier function should be trained first and after training phase will be ready for prediction. Also we discussed the idea of training set matrix in the previous chapter. Test set matrix is similar to training one but only without the class vector which should be the output of the classifier.

3.5.1 SVM

```
29 SVM_Model <- train(Y ~., data = Trainset, method = 'svmLinear3')
30 Testset<- as.matrix(Testset)
31 prediction <- predict(SVM_Model,newdata=Testset)
```

Figure 3.22: Training and predicting with SVM model

3.5.2 Random forest

```
17 RandomForest_Model <- randomForest(x = Trainset,y = Trainset$y , ntree = 60 )
18 prediction = predict(RandomForest_Model,newdata = Testset )
```

Figure 3.23: Training and predicting with Random forest model

3.5.3 Naïve Bayes

```
3  NaiveBayes_Model = naiveBayes( Y~.,data =Trainset)
4  prediction = predict(NaiveBayes_Model,Testset )
```

Figure 3.24: Training and predicting with Naïve Bayes model

3.5.3 Decision Tree

```
10 DecisionTree_Model <-rpart(Y~.,Trainset , method =  "class")
11 prediction = predict(DecisionTree_Model,newdata = Testset,type = "class" )
```

Figure 3.25: Training and prsedicting with Decision tree model

Chapter 4

Results

In this chapter, we display the results of classification process with respect to the seven approaches that we stated in 2.6. For each approach, we display its results in a separate section. We also define the concept of confusion matrix and how we can calculate the accuracy for given approach.

4.1 Accuracy and Confusion matrix

We calculate the accuracy for a given approach by calculating its success rate or perctenage of success for a testing set. We tested each approach by a fixed testing set which is a sample from the original dataset. At the beginning, we divided the original dataset into training set and testing set, each set is sampled from the original dataset. Training test is 70% and testing set is 30%.Confusion matrix is the main measure tool for classification process. Figure 4.1 shows a confusion matrix for a classification model with only two predefine class, yes and no. All test cases are 165 case. Now, we should compare the true results to the predicted result with the confusion matrix. Each row represents an true class instances which is compared to each column that represents the prediction of the model. The intersection cell between a row and a column with the same class name is the true prediction for the model with respect to some class instances.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Figure 4.1: example of confusion matrix

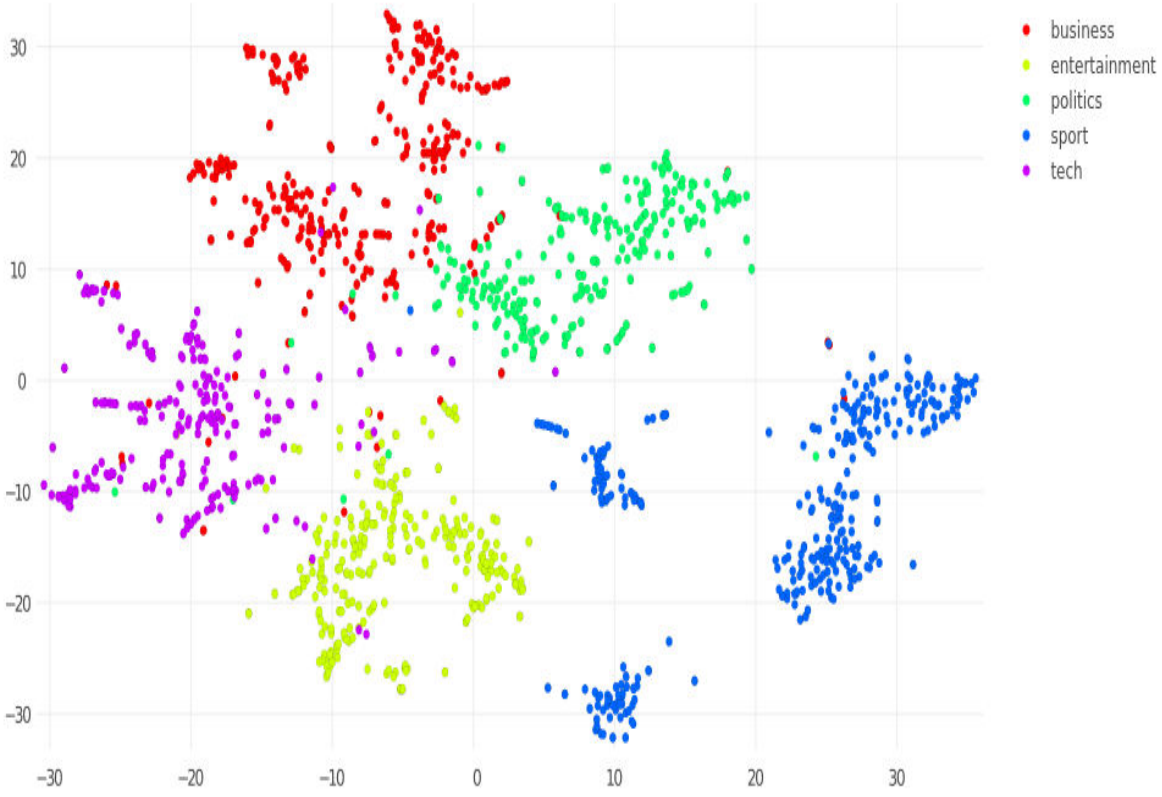

Figure 4.2: t-SNE dimensionality reduction

Figure 4.2 shows t-SNE dimensionality reduction for the dataset, which is used for data visualization for high-dimensional spaces. t-SNE constructs visualization for a high-dimensional space by projection into 2-dimensional space. The visualization shows that different article classes fall in different regions in the feature space which means that the classifcation process should have accurate classification performance.

4.2 Approach A

Figure 4.3: SVM model confusion matrix

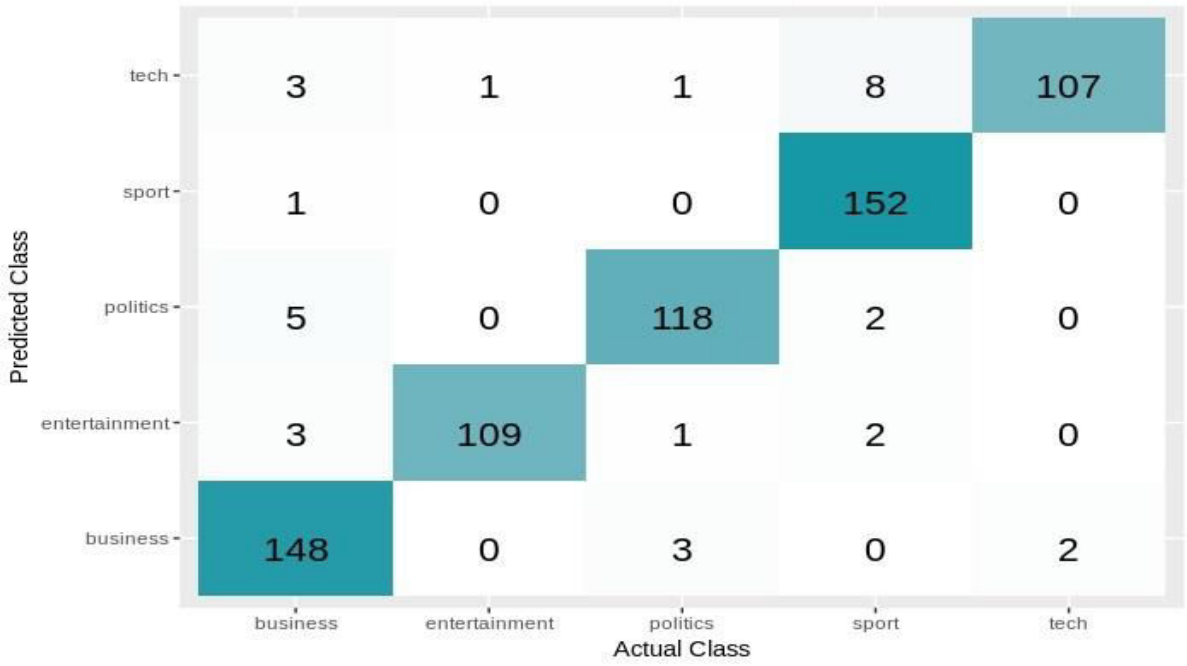

Figure 4.4: Random Forest Confusion matrix

Figure 4.5: Naïve Bayes Confusion matrix

Figure 4.6: Decision Tree Confusion matrix

	naive bayes	decision tree	random forest	support vector machine
1	0.7327327	0.7207207	0.951952	0.9534535

Figure 4.7: calculated Accuracy for each model

4.3 Approach B

Figure 4.8: SVM model confusion matrix

Figure 4.9: Random forest model confusion matrix

Figure 4.10: Naïve Bayes model confusion matrix

Figure 4.11: Decision Tree model confusion matrix

	naive bayes	decision tree	random forest	support vector machine
1	0.8528529	0.7612613	0.9429429	0.972973

Figure 4.12: calculated Accuracy for each model

4.4 Approach C

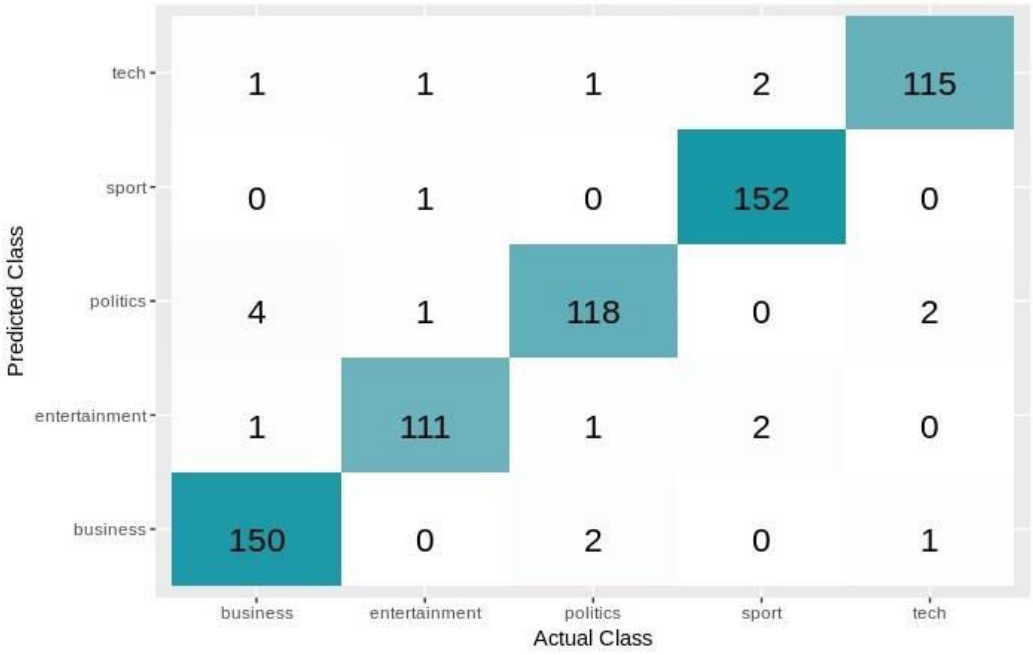

Figure 4.13: SVM model confusion matrix

Figure 4.14: Random Forest model confsusion matrix

Figure 4.15: Naïve Bayes model confusion matrix

Figure 4.16: Decision tree model confusion matri

	naive bayes	decision tree	random forest	support vector machine
1	0.8588589	0.7657658	0.960961	0.96997

Figure 4.17: calculated Accuracy for each model

4.5 Approach D

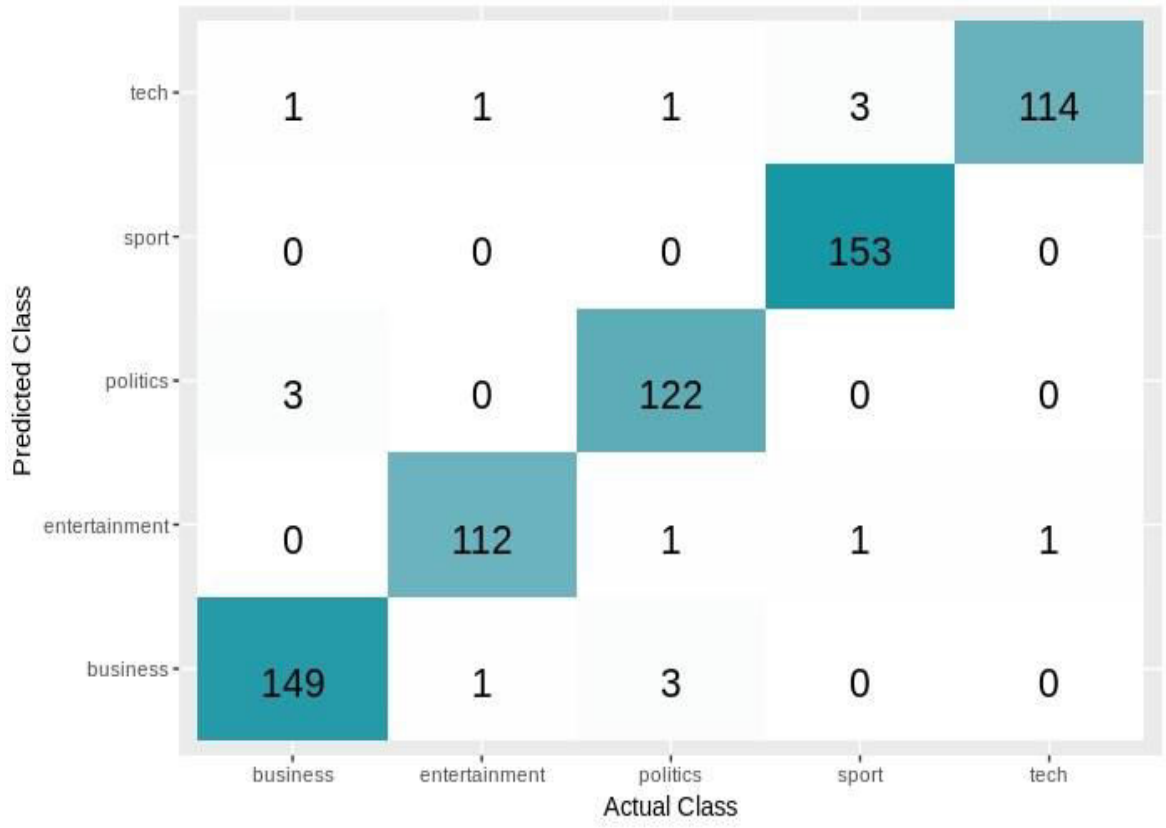

Figure 4.18: SVM model confusion matrix

Figure 4.19: Random forest model confusion matrix

Figure 4.20: Naïve Bayes model confusion matrix

Figure 4.21: Decision Tree model confusion matrix

	naive bayes	decision tree	random forest	support vector machine
1	0.8858859	0.7627628	0.951952	0.975976

Figure 4.22: calculated Accuracy for each model

4.6 Approach E

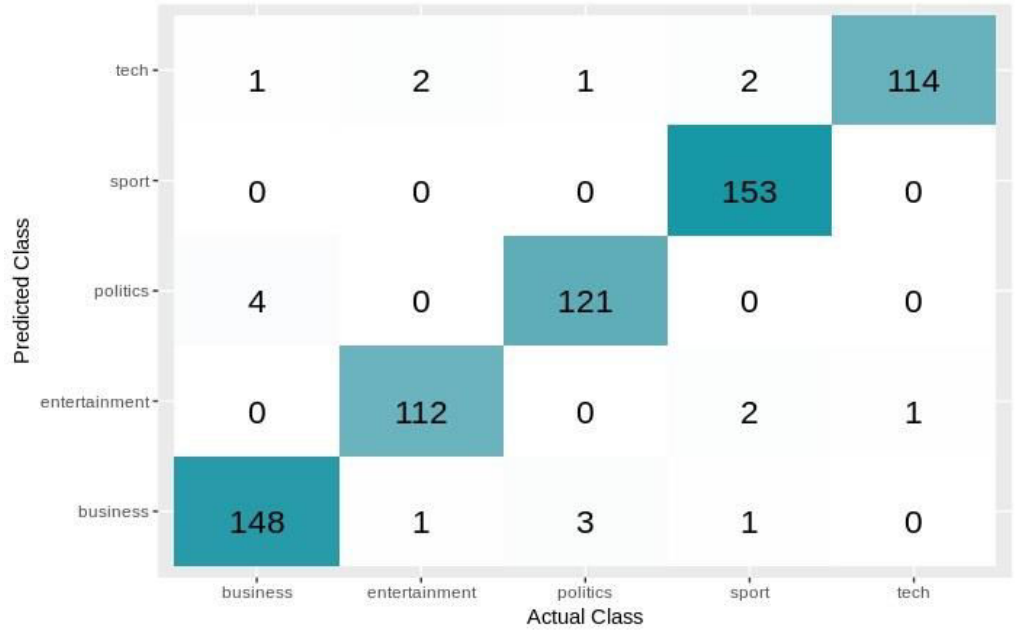

Figure 4.23: SVM model confusion matrix

Figure 4.24: Random forest model confusion matrix

Figure 4.25: Naïve Bayes model confusion matrix

Figure 4.26: Decision tree model confusion matrix

	naive bayes	decision tree	random forest	support vector machine
1	0.8273273	0.7612613	0.9504505	0.972973

Figure 4.27: calculated Accuracy for each model

4.7 Approach F

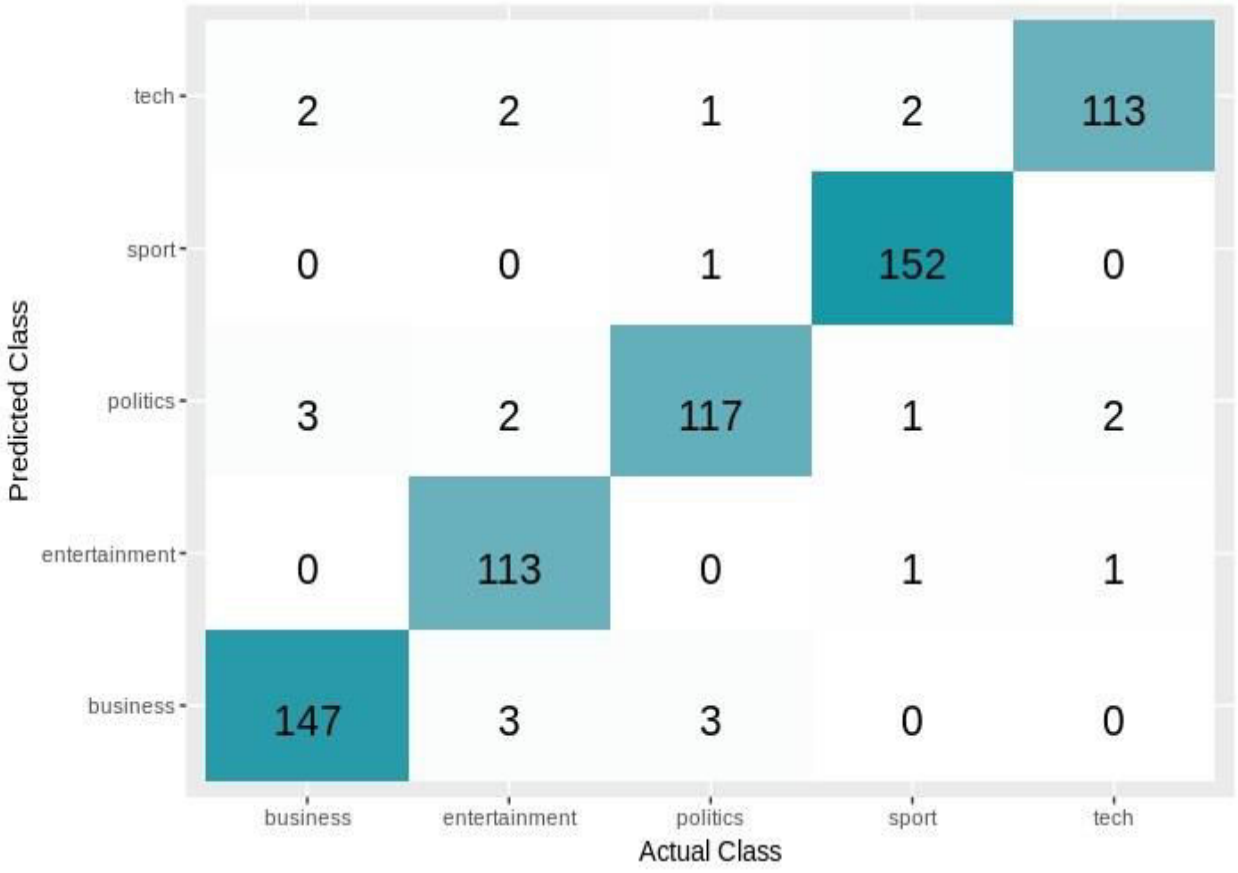

Figure 4.28: SVM model confusion matrix

Figure 4.29: Random forest model confusion matrix

Figure 4.30: Naïve Bayes model confusion matrix

Figure 4.31: Decision tree model confusion matrix

	naive bayes	decision tree	random forest	support vector machine
1	0.8603604	0.7612613	0.9429429	0.963964

Figure 4.32: calculated Accuracy for each model

4.8 Approach G

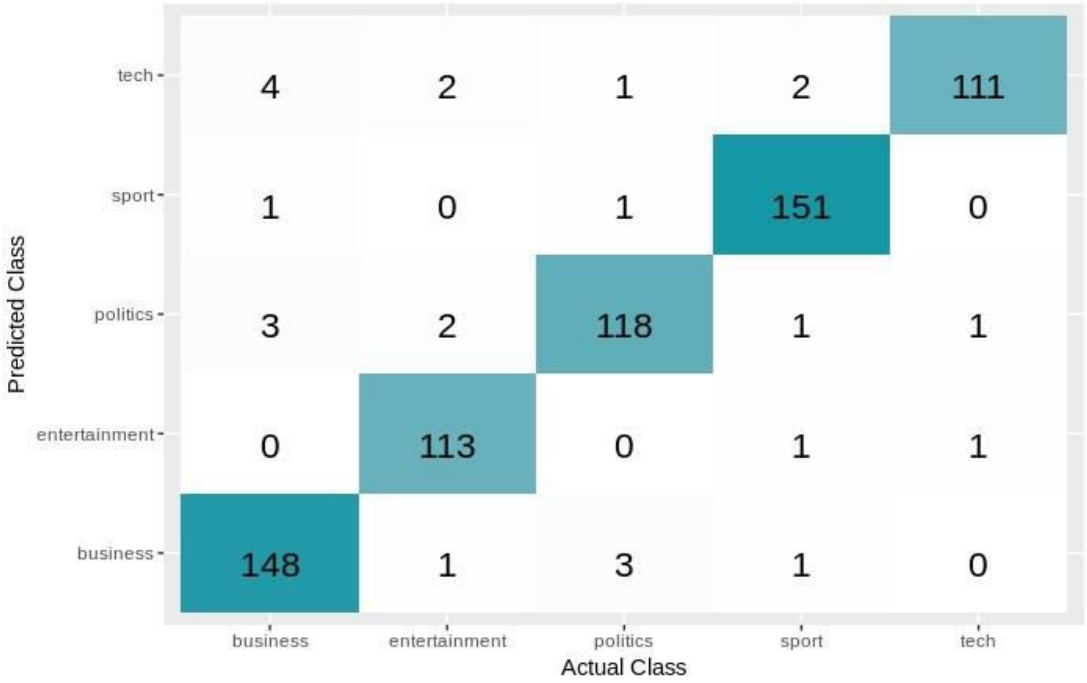

Figure 4.32: SVM model confusion matrix

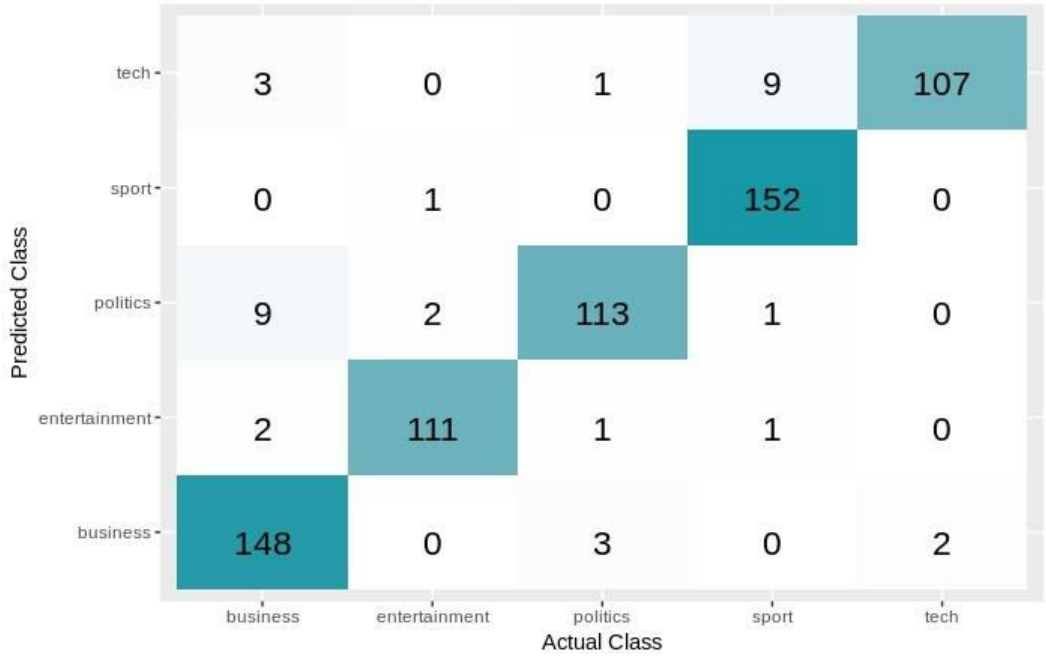

Figure 4.33: Random forest model confusion matrix

Figure 4.34: Naïve Bayes model confusion matrix

Figure 4.35: Decision tree model confusion matrix

	naive bayes	decision tree	random forest	support vector machine
1	0.8738739	0.7612613	0.9474474	0.9624625

Figure 4.36: Decision tree model confusion matrix

Chapter 5

Conclusion

In this chapter, we summarize what's done in this study and explain the results that we declared in the previous chapter. Quickly, we state pros and cons of each approach and for each single technique.

In this study, we applied different approaches on BBC's dataset for articles and news. Our purpose was to identify the promising factors that can increase classification accuracy. Those factors are text representation techniques, stemming techniques, and feature selection techniques.

First notable thing about the results is that Decision tree models accuracy is very limited with all approaches which means that Decision tree isn't a good choice for text classification. On the other hand, Support vector machine model was the best in the most approaches. This note lead us to infer that support vector machine is the best classifier for text classification purpose among the four models. Also Random forest model accuracy was very good, but its accuracy decreases when the dimensions were reduced. Random forest is better with higher dimensional feature space. Naïve Bayes model was moderate in the most approaches, but its accuracy enhanced with stemming either Stanford or Porter. The impact of stemming with porter and lemmatization with Stanford approximately is similar. About feature selection techniques, Chi-square is better than information gain with respect to classification accuracy. TF-IDF is the most efficient text representation technique. Bigrams representation didn't make a notable difference. The best approach that maximize the accuracy in our experiments was approach D. In approach D, we used Porter stemmer and TF-IDF weighting.

References

1. *Probability & Statistics for engineers & Scientists, Walpole, 9th Edition.*
2. *Mathematical statistics and data analysis, Rice.J.A, 3rd Edition.*
3. *Probability, Jim Pitman, Springer Texts in Statistics.*
4. *An introduction to Information Retrieval, Christopher, Prabhakar, Hinrich, 2009.*
5. *Learning from data, Yaser, Malik, Hsuan.*
6. *Machine Learning with R, Brett Lantz,2nd Edition.*
7. *Introduction to probability, Bertsekas, Tsitsiklis.*
8. *Elementary Statistics, Triola, 10th Edition.*
9. *Data Mining concepts and techniques, Jiawei, Kamber, Pei, 3rd Edition.*
10. *Representation and classification of text documents, Harish, Guru,Manjunath.*
11. *Text Classification using machine learning techniques, Ikonommakis, Kotsiantis, Tampakas.*
12. *N-gram-Based Text categorization, Cavnar, Trenkle.*
13. *Machine learning in automated text categorization, Sebastiani.*
14. *Text Categorization using machine learning, Hirotoshi Taira.*

15. *The Elements of statistical learning, Hasti, Tibshirani, Friedman.*
16. *Pattern recognition and machine learning, Bishop.*
17. *An introduction to statistical learning With Application in R, James, Witten, Hastie, Tibshirani.*
18. *Text Categorization with Support Vector machines: learning with many relevant features, Throsten Joachims.*
19. *Bayesian Reasoning and machine learning, David Barber.*
20. *Introduction to machine learning, Ethem Alpaydm, 2nd Edition.*

Appendix A: UML Modeling

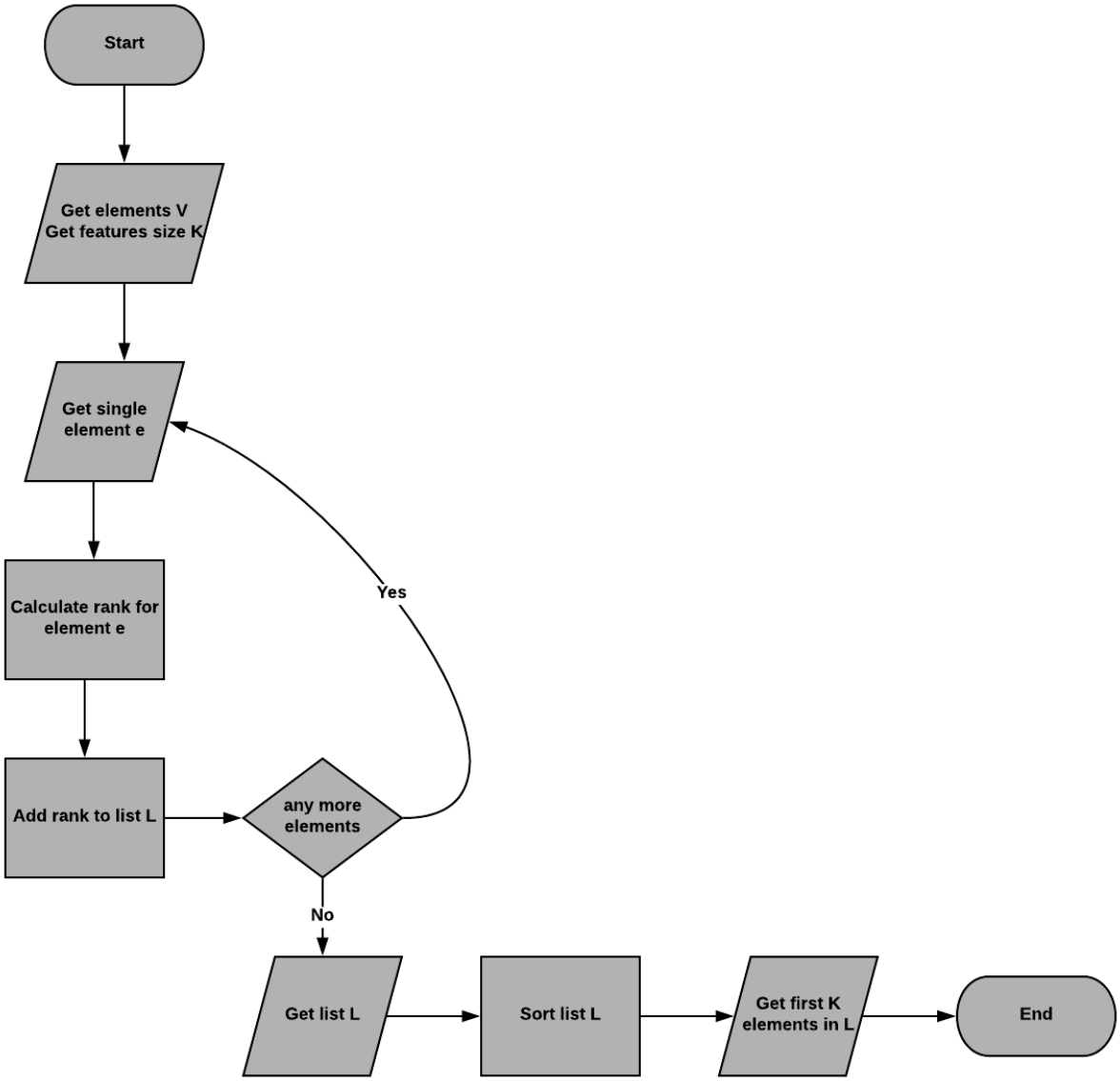

Diagram 1: Flowchart diagram describes feature selection process

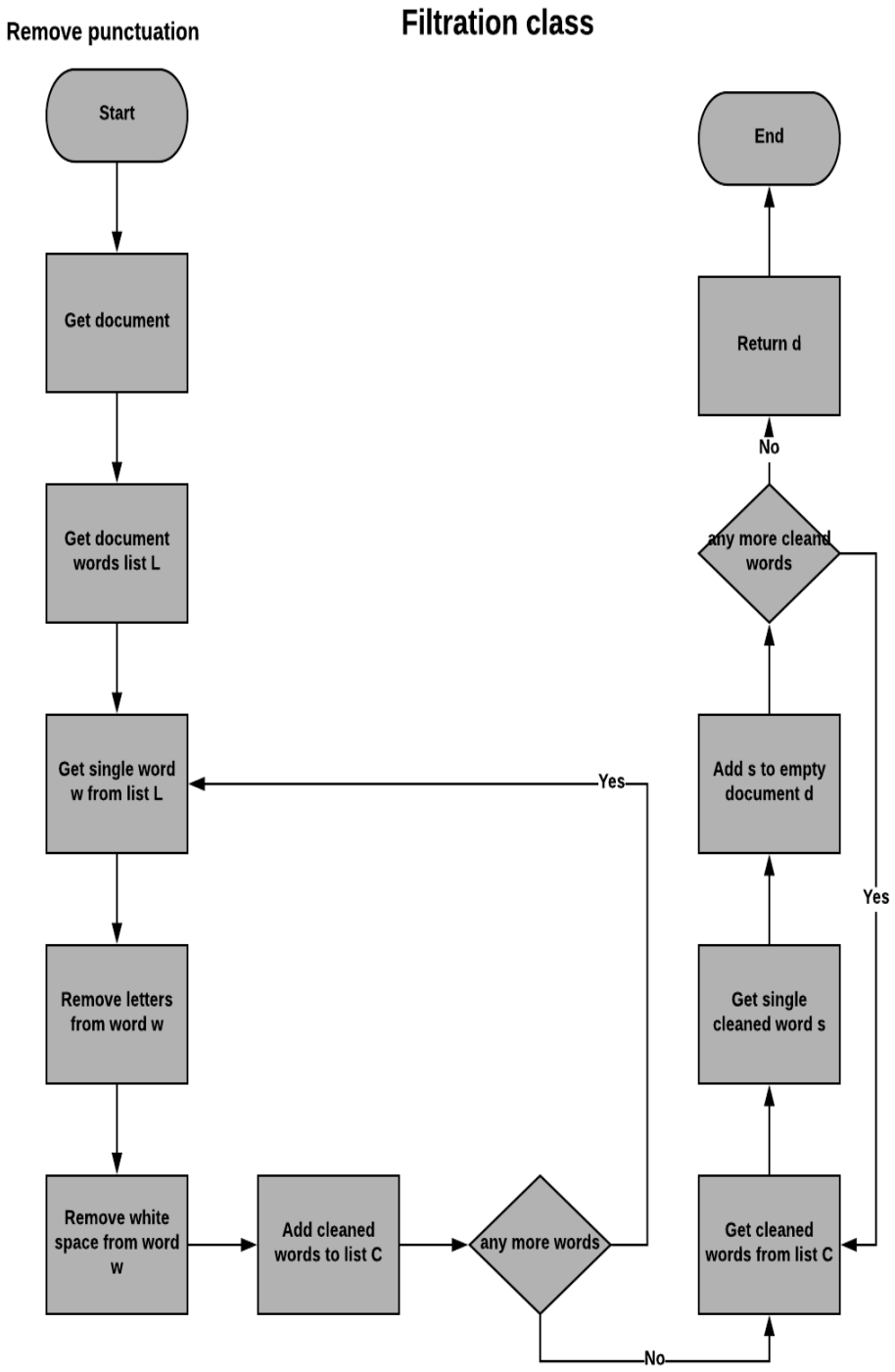

Diagram 2: Flowchart diagram that describes Removing punctuation marks process

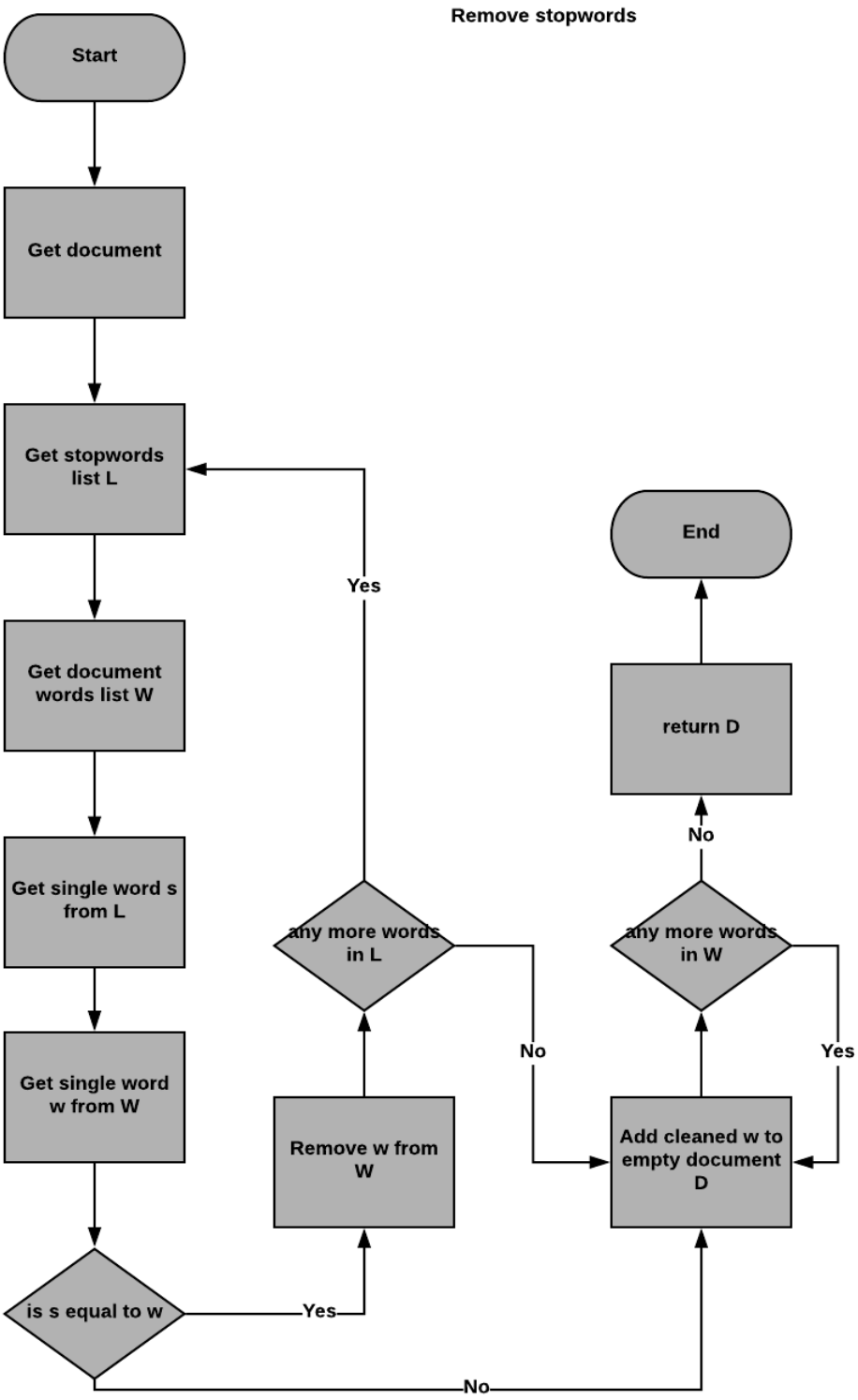

Diagram 3: Flowchart describes removing stopwords process

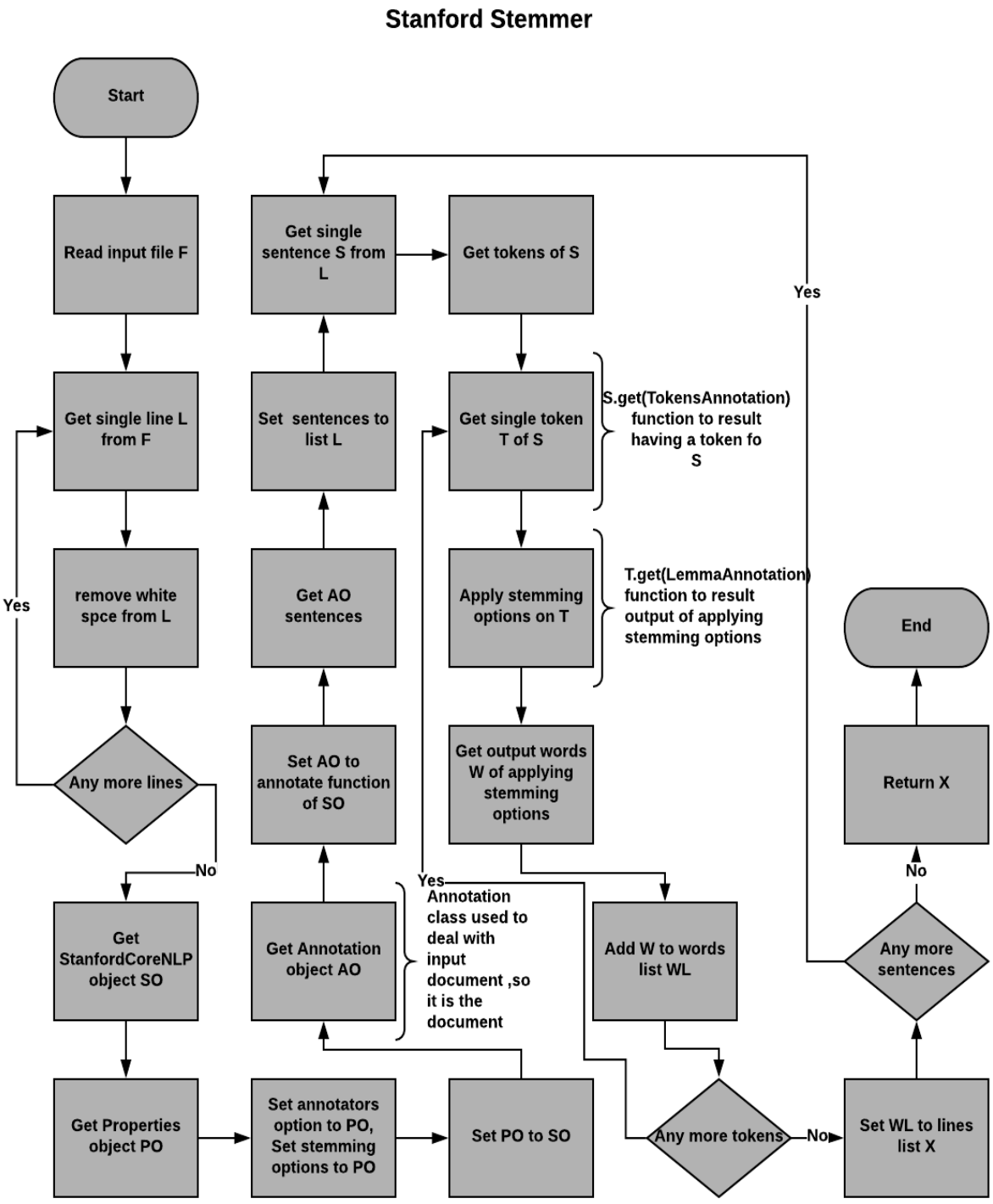

Diagram 4: Flowchart describes lemmatization process with Stanford CoreNLP

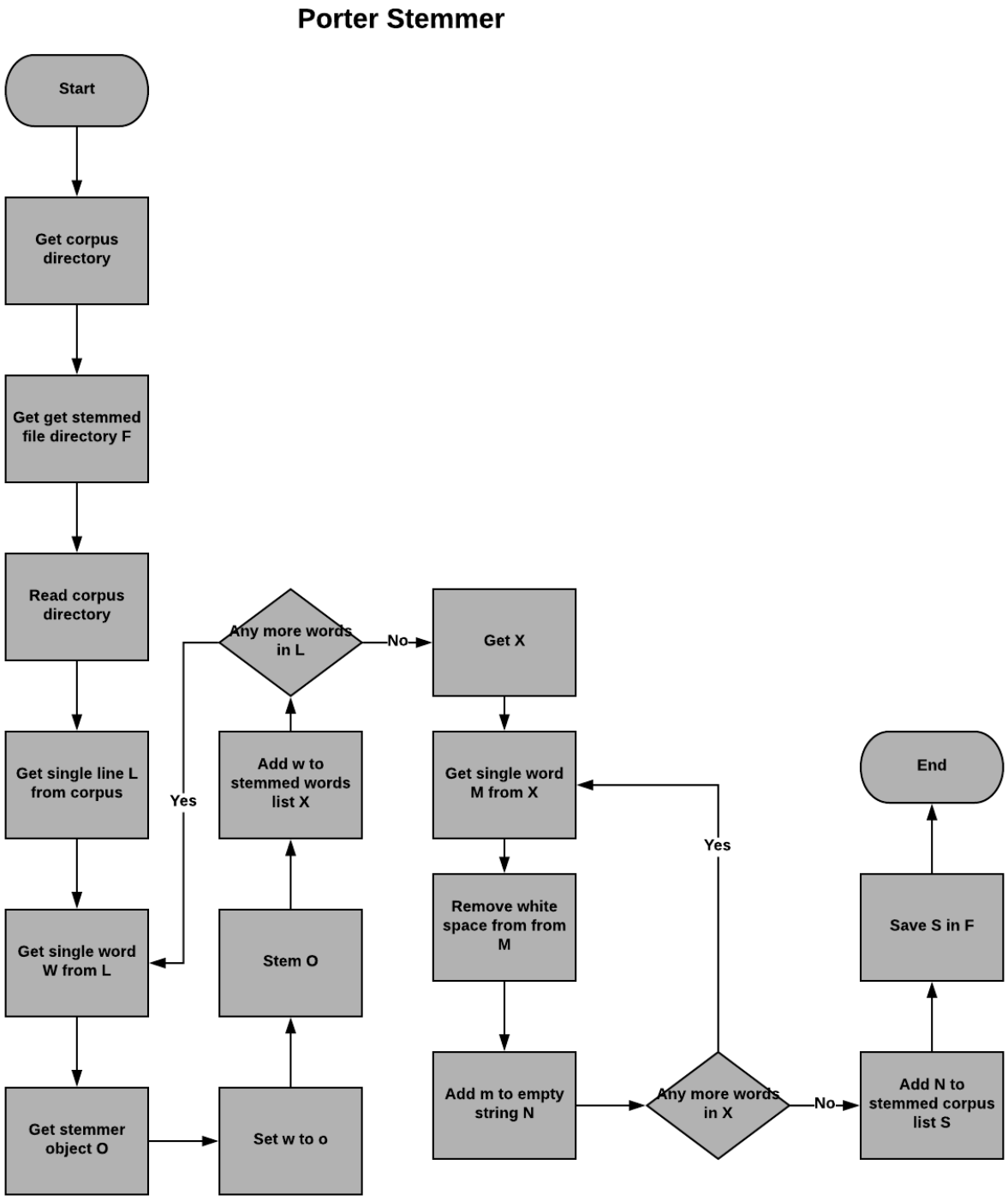

Diagram 5: Flowchart describes stemming process with Porter

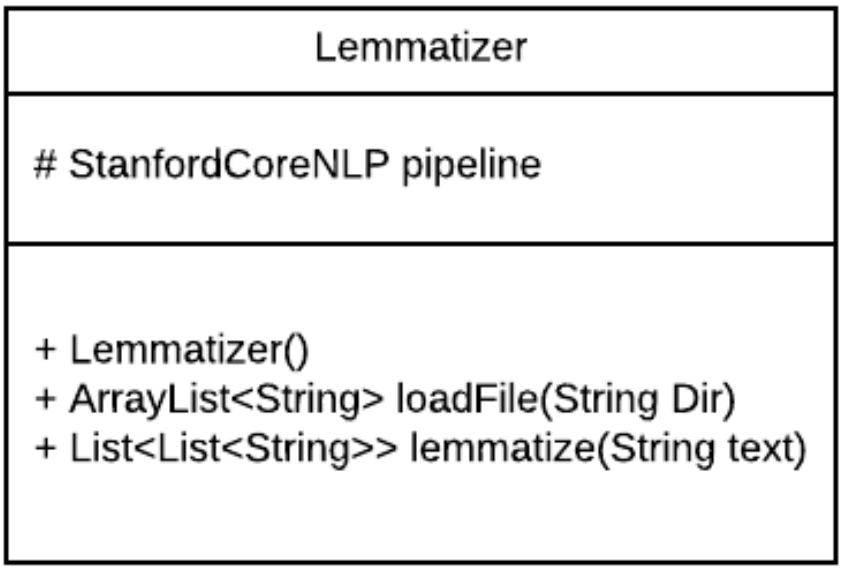

Diagram 6: Class Structure of lemmatizer class that uses Stanford CoreNLP API

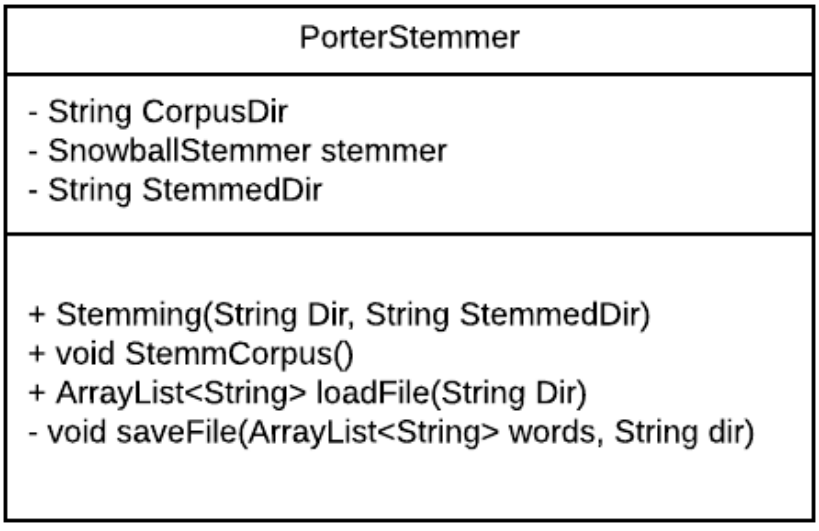

Diagram 7: Class Structure of PorterStemmer class that uses Snowball API

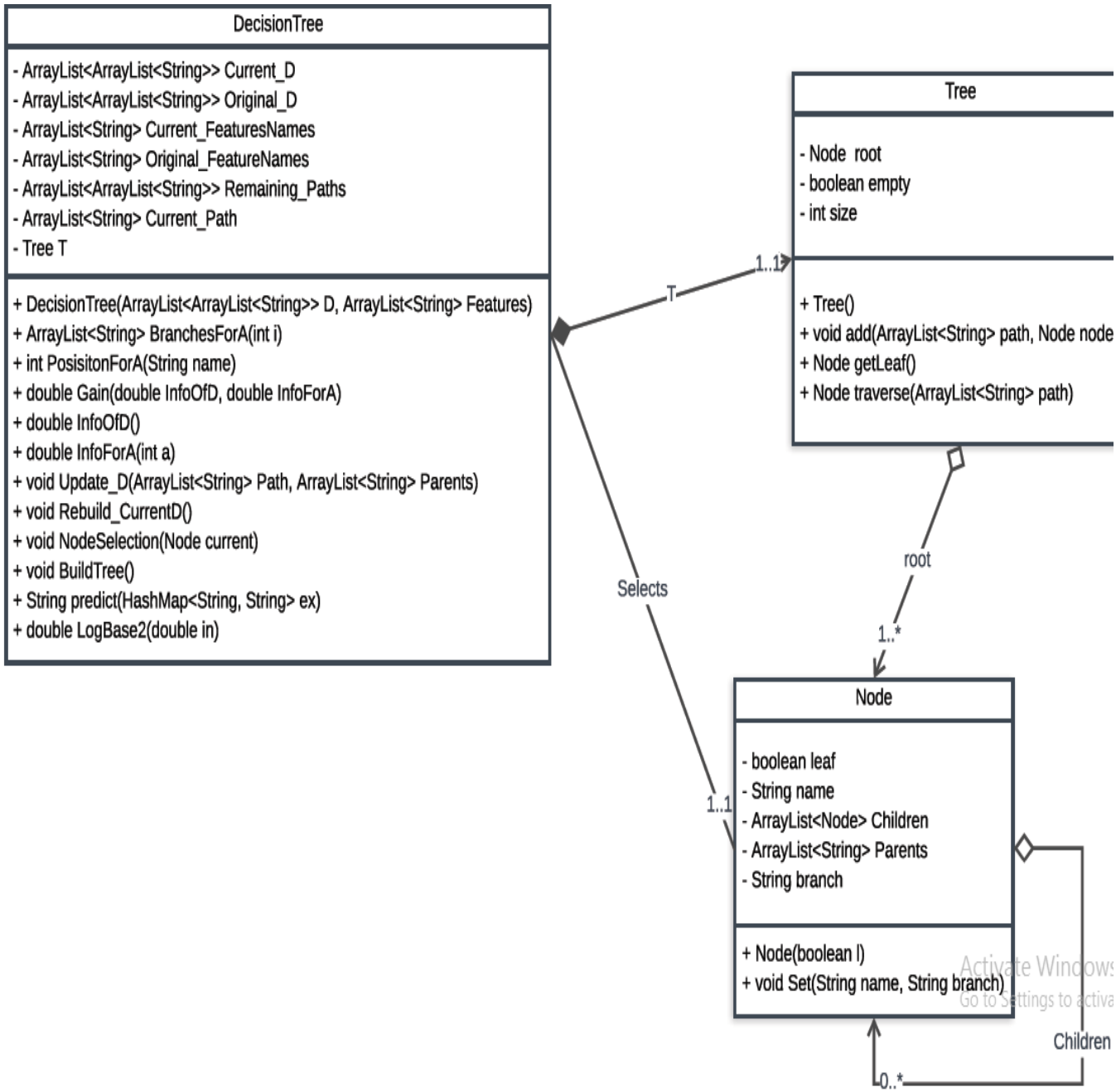

Diagram 8: Class diagram describes How decision tree model can be implemented

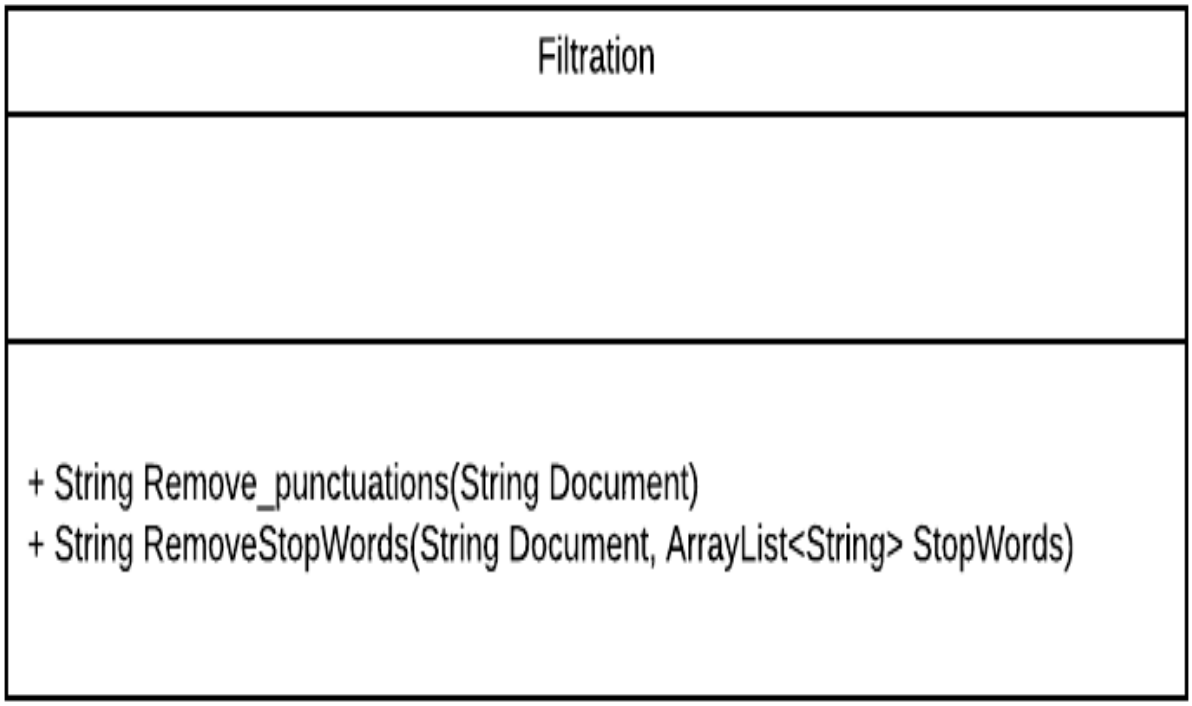

Diagram 9: Class Structure of class filteration that contains static functions for preprocessing