A Tool for Indexing and Classifying Unstructured Textual Documents Based on Product Family Algebra

A TOOL FOR INDEXING AND CLASSIFYING UNSTRUCTURED TEXTUAL DOCUMENTS BASED ON PRODUCT FAMILY ALGEBRA

BY

DEEMAH ALOMAIR, BSc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science in Computer Science

Master of Science (2020) McMaster University

(Computer Science) Hamilton, Ontario, Canada

TITLE:	A Tool for Indexing and Classifying Unstructured Textual Documents Based on Product Family Algebra
AUTHOR:	Deemah Alomair BSc. (Computer Information System) Imam Abdulrahman bin Faisal University, Saudi Arabia
SUPERVISOR:	Dr. Ridha Khedri
NUMBER OF PAGES:	x, 102

Abstract

Unstructured textual documents comprise the bulk of the data used and archived by organizations within all sectors of the economy. The need to index and classify these documents became an interesting topic that gained more attention in the field of data analytic. Different approaches are used to perform indexing and classification of textual documents. They range from supervised Machine Learning (ML) approaches to rule-based ones. There is a need for exploring novel classification approaches that exhibit better effectiveness and performance in classifying the increasing volume of this kind of data.

In this thesis, we propose a novel approach to index and classify unstructured textual documents based on Product Family Algebra (PFA) and implemented using Binary Decision Diagram (BDD). In the proposed approach, a signature is first constructed for a document or a family of documents. The signature is relative to a dictionary of the typical words used in the category under consideration. Then, using operations on product family implemented using BDDs, we carry the classification of a document or families of documents using their signatures. Since ML methods are considered to be the de facto standard in document classification and to compare our method performance to their, we implement four ML classification methods: Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (K-NN), and Decision Tree

(DT). After that, we merge these modules into one software system called Smart Document Classification System (SDCS).

The assessment of our approach to the classification of textual documents shows its flexibility in indexing and classifying families of textual documents. The classification is deterministic and on a single document (not families of documents), it compares very well with the SVM ML-classifier. Using rules articulated in the language of PFA, It offers a variety of ways for classifying families of documents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Document classification and indexing is a widely-used process for a long time. Even before having electronic data available through the Internet, public libraries were using classification and indexing systems to organize books based on some criteria. The most popular classification systems used by public libraries are Library of Congress (LC) and Dewey Decimal Classification (DDC) systems. Both are subject-based classification systems. LC is a widely-used classification system in most of the academic libraries around the world [CIW16]. Each book is assigned to a Recall Number which consists of three main parts. The first part is alphabetic. It reflects the class of the main study subject to which the book belongs. Hence, there are many classes in this classification method where each class contains a special study subject. The second part is numeric which represents the specific area of study. The third part called the Cutter Number. It is an indication of the author's name. It consists of a number and letters starting with a decimal dot. The date of publication and the number of the copies could be added as the fourth and fifth optional parts. DDC [Kap01][MBMN09] has ten broad areas of study ranging from "000" to "999" used to classify the book

in the library. Each number reflects a main subject study area. For instance, "000" is for *computer science, information, and general works* subject area. Every broad area or class is divided into divisions. Each division is further divided into sections. Hence, every book is labeled with this Recall Number that consists of three digits that represent a class, division, and section. In the case that there are extra subsections, their numbers can be added after the decimal dot. For example, the recall number of a book that belongs to the *Ice skating* subsection of the section *Athletic and outdoor sports and games* under the division of *Recreational and performing arts* of the main class *Arts & recreation* is 796.91. For more details about the classification using DDC system, we refer the reader to [MBMN09].

All previous classification methods have a limitation and not suitable for the huge number of data available today. Classifications that involve the subject of study would require either human interaction to identify the subject, or a tool like the one we are processing in this thesis.

Recently, electronic data has been growing rapidly. It is becoming essential for many basic transactions in all the sectors of the economy. This form of data can vary in characteristics and usage. The main important characteristics for any digital data are its volume, variety, and velocity, which are known as the 3V's [SC17]. Volume refers to the quantity of data usage. variety is related to the difference in the types of data and its structure: structured, unstructured, or semi-structured. Velocity is defined as the speed in which the data are processed. Electronic data that has these three characteristics are commonly referred to as big data [DMGG16]. It is a collection of data sets that regular tools fail to handle. It therefore, needs new approaches and tools to exploit the knowledge it contains. Big data has many applications, such as

indexing and searching, Machine Learning (ML) [Jat18], and security.

The increase of big data usage in different fields makes analyzing big data an important topic to get valuable knowledge. Big data analytic is about finding covered patterns and hidden relationships between data regardless of its structure. Analyzing big data involves a broad range of activities that aim at enabling organizations to reduce cost and enhance performance in daily processes and in better decision-making [VG17].

On the other hand, using big data in daily processes gives rise to many challenges. One of the most known challenges is dealing with and classifying unstructured data. Unstructured data refers to data that is gathered from the World Wide Web as text, picture, or video and not formed in database management system format (i.e., Unstructured document) [KBLK10]. One of the users of unstructured documents are organizations within all the sectors of the economy. Accessing and operating these documents is essential for achieving organizations' goals. This opens the doors to the importance of the document classification and indexing. It becomes one of the highest priorities for any organization. However, this process must be done efficiently to gain more advantage.

Unstructured document classification using ML is the process of assigning a document to the most appropriate class from a predefined training set. It is considered as a supervised learning approach. Supervised learning is a way to classify a new document based on previous training data or experience [KBLK10]. Classification is divided into two steps. The first step is forming the training set. It involves labeling all the classes of the training set. The second step is classifying a new document based on the predefined classes in the training set [SJD13]. On the other hand, clustering is considered

unsupervised learning approach. Unsupervised learning is classifying a document in a new way where no previous knowledge or training data involved [KBLK10]. Clustering is the process of assigning a document to the right category. However, it does not need a training set to do so. It usually depends on measuring the similarities between documents. Distance measuring is one of the methods used to achieve the clustering goal [SJD13]. Many types of research have been made to implement an effective automatic way to classify or cluster documents.

There are many document classification methods. However, we find in [BG12], that these methods have disadvantages in different classification aspects that affect the accuracy, speed, or performance of the classification result. Some of the well-known methods to classify documents automatically are Support Vector Machine (SVM) [MM09], Naive Bayes (NB) [BK19], and K-Nearest Neighbor (K-NN) [MS17]. Although they work perfectly in some situations they have different disadvantages. For instance, despite SVM is a very popular technique, it is considered complex and consumes time and memory to train the data and make the classification decision [KBLK10]. Moreover, each feature is measured independently even if it is relevant to other features. This might lead that a single document may belong to more than one category at the same time, which leads to confusion. NB is considered to have lower performance than other classification methods like SVM [KBLK10]. On the other hand, the K-NN algorithm [MS17] is known as a Lazy Learner. It needs more time to test new documents in comparison to other methods [BK19]. Besides, it could involve a high amount of calculations to compute distances between documents [KBLK10].

Unstructured document indexing is another important topic under the umbrella of

big data analytic [MMM11]. Indexing is the process that precedes classification or information retrieval processes. It represents the document in a format that is understandable by the classifier. It is usually done by assigning weight to each term in a document, then it represents the document as a vector of terms along with their weights. Indexing is not an easy task as it appears because of the synonyms of words. Moreover, the place in which the term appears in the document and its frequency play an important role in its importance and its meaning. Recently, many automated indexing techniques are available in [TAAEG10] [BA13] [KL05]. Ongoing research in this area is getting more interest.

A conventional indexing schema is a method of indexing documents based on the number of occurrences of a term in the document. It ignores where the word or term occurs. This might reduce the accuracy and performance of the technique. Many approaches tried to improve the conventional method in a way that the place of the term is considered.

Sentence Ranking [MMM11] is one of the methods to improve the conventional indexing schema. It considers the number of occurrences of the term and where it appears to give the final weight for each word. However, this method has been implemented in a small data set and it is not proved to work well in a big data context. On the other hand, several stemming algorithms to indexing unstructured documents have been proposed like Stemmer Algorithm to Indexing Unstructured Document (SAID) [BO15]. The aim of stemming is to improve the indexing process by removing the prefix, suffix from the term. This reduces the number of terms in a single document and gives more accurate indexing results.

1.1 Problem

Three important factors affect indexing and classifying unstructured documents: Stemmer approaches, weighting schemes, and classification methods [BEMS17]. Approximately all approaches in the big data field need some improvement in terms of accuracy, recall, precision, and speed. All recent algorithms have some disadvantages in a different aspect. Indexing and classifying a large number of documents to get the desired output became a challenge faced by many organizations and it will be further increased with the growing number of electronic data produced every day. This thesis aims to address the problem of indexing and classifying unstructured documents. Our main objective is to design and implement a tool based on Product Family Algebra (PFA) [HKM11] implemented using Binary Decision Diagram (BDD) [TLH09] for indexing and classifying unstructured documents with a limited cost on machines and users.

1.2 Motivation

Document indexing and classification is not a new topic in the world of computer science, especially in data analytic and ML fields. Researches allocated huge efforts to get the most suitable way to classify and index documents automatically with the least time and memory. The rapid growth of electronic documents made this process more complicated.

By surveying the literature as briefly presented in the previous section, we find that in document classification and indexing techniques, there is no single approach without a significant weakness. Most of the approaches are non-deterministic, where erroneous

results are likely. In the end, this motivates us to explore a new deterministic tool that helps in classifies and index a huge number of documents automatically with minimum efforts.

1.3 Contribution

The main contributions of our thesis include:

(i) Propose signature as a mean for indexing unstructured documents based on BDD. Then, perform operations on signatures to get the signature of a family of documents or the commonalities between documents. This allows us to reason on signatures rather on documents.

(ii) Construct a module called rule-based approach applied to signatures that classifies documents based on signatures.

(iii) The implementation of four ML classifiers: SVM, NB, K-NN, and Decision Tree (DT).

(iv) Present a system called SDCS that indexes and classifies documents using two approaches: The rule-based approach applied to signatures and the ML-based approaches.

(v) Compare the performance of the rule-based approach applied to signatures and ML-based approaches: SVM, NB, K-NN, and DT.

1.4 Structure of the Thesis

In Chapter 2, we give a comprehensive view on the literature review related to document indexing and classification. In Chapter 3, we explain the mathematical background of the proposed rule-based approach applied to signatures. In Chapter 4, we show the system design and its main components. In Chapter 5, we show the assessment of the tool. In Chapter 6, we conclude and point for future work.

Chapter 2

Stemming, Indexing, and Classification

The number of unstructured documents gathered from different resources has been increasing significantly. About 80% of an organization's documents considered as unstructured documents [KBLK10]. Thus, the number of research on exploring possible classifying and indexing techniques for unstructured documents had gained more attention. In this chapter, I present research work that has been carried in this area. When exploring indexing or classification, we found that the stemming process always precedes these processes. Stemming is an important step to give the most efficient and accurate indexing and classification results [Har18]. I briefly present the main techniques for stemming, indexing, and classifying unstructured documents.

2.1 Stemming Approaches

Many of the indexing and classifying techniques presented in the literature utilize stemming for the preparation of documents prior to their usage. I present below some of the well-known stemming approaches.

Stemming is the process of getting the root of a word by removing any suffixation. The stemming process aims to reduce the length of the word and cut down the number of the words in the document to be classified or indexed [BO15]. There are many stemming algorithms that are language-specific [Har18]; each language has its own rules for structuring a word. Also, there are many types of stemmers: Rule-based stemmers [Har18], statistical-based stemmers [Har18], corpus-based stemmers [Har18], and context-sensitive stemmers [Har18].

One of the rule-based stemming algorithms is Porter stemmer [Har18]. There are approximately 62 different rules in this algorithm. It is considered fast and produces a good stemming result. Most of its rules are for removing the affixes from words. For instance, one of the common suffix-based removal rules is used when the word ends with "SSES". It then converts it to "SS". For instance, the word "Processes" is reduced to "Process" and "Caresses" is reduced to "Caress". This method is commonly used in stemming English unstructured documents.

Lovins stemmer is one of the widely used stemming algorithms [BEMS17] [Har18]. It is a rule-based stemming method. All suffixes are saved in a table called endings. It checks if the rightmost side of the word appears in the ending list, then it removes it. For instance, "Suffocation" is turned into "Suffoc". The second step deals with spelling exceptions. After removing the suffix, some words need to be adjusted to

turn into valid words. This process called Partial Matching or Recording and it uses 29 conditions and 35 transformation rules.

We find in [BO15] Stemmer Algorithm to Indexing Unstructured Document (SAID), which is a modified version of Porter and Lovins algorithms. The main contribution of SAID is adding more than 100 suffixes for single words and over 40 suffixes for compound words. The first step of the algorithm is to search for compound words, then removes the composition. For example, "Chairman" or "Relationship" is considered as compound words. Then, it searches for a suffix in each word and applies the correct rule for it like in Porter, or Lovins methods. Next, it introduces new rules for suffix elimination to get more accurate results. These are considered new rules that are not mentioned in Lovins method. For example, "Greater" is converted to "Great". Finally, it applies the transformation rule to transform the suffix to another suffix. For instance, eliminating the doubling from a word after removing its suffix is considered changing a suffix to another suffix. For instance, "Preferred" turns into "Preferr" by applying the rule of removing "ed", then "Preferr" becomes "Prefer" by applying the transformation rule. The experimental test showed that this method delivers more accurate results than Porter and Lovins algorithms.

2.2 Indexing Techniques

Indexing is the process that serves as a median between the stemming, or document preprocessing phase, and the classification phase. After filtering the document by removing all unwanted words and stemming, one needs to represent the document in a way readable by a classifier. Indexing a document consists of two steps: document representation and word weighting. Most of the document representation techniques

are taken from the information retrieval field [Seb02]. Document representation can be categorized into two main groups: The Boolean model and the vector model. As mentioned in [LMG09], the Boolean model is based on set theory and Boolean algebra. The basic idea behind it is to treat a document as an index term based on a given query. The weight of the term is 1, if the word in the query appears in the document and 0 otherwise. For instance, if the query contains the word "Data" and we have three documents then we can ends with $\{(doc_1,0)$, $(doc_2,0)$, $(doc_3,1)\}$, which means the word "Data" only appears in the third document. The vector model is an algebraic approach in which the document represented as a vector of terms. For example, $d_1 = \{t_1, t_2, t_3\}$. Where d_1 is document number 1 and t_1,t_2, and t_3 are terms. We represent a class of documents with a matrix where each row represents a document of the class and each column represents a term. One example of this category is representing a document as a Vector Space Model (VSM)[Mun07].

Representing a document by one of the above categories could be modified by one of the weighting schemes available. This is done by giving each word a special weight of importance. One of the well-known schemes for weighting is Term Frequency–Inverse Document Frequency (TF-IDF) [MV15]. Below is a summary of some of the weighting techniques known in this field.

We found in [MV15], that TF-IDF is an unsupervised learning scheme. It is used to give each word in a document a specific weight. This weight is based on the importance of the word in the document and on the importance of the word in a set of documents within a class. The method involves the following steps. First, it calculates the Term Frequency (TF) for each word, which is the number of occurrences of a word in a single document. The more the word appears in the document the more

it is important. Then, it calculates the Inverse Document Frequency (IDF), which is the importance of a word in a set of documents within one class. The IDF for a term t is calculated using the formula: $\texttt{IDF}(t, C) = \log \frac{N}{D}$, where N is the number of documents in the class C and D is the number of documents that contain the word in the same class C. If a given class does not contain the word, this leads to division by 0. To go around this problem, we add 1 to the denominator. If a word has a high number of occurrences in the document, it gets a high TF value. In contracts, if a word appears in many documents, it gets a lower IDF value. Finally, TF-IDF is calculated using this formula: $\texttt{TF-IDF} = \texttt{TF} \times \texttt{IDF}$. This weighting scheme can be used further in many information retrieval applications or document classification. However, the main drawback of this method is not considering the semantic or the location of the word in the document, which reduces the performance radically. Many researches have been done to improve this conventional method in different ways as I present below.

The paper [SWLH16] discusses a supervised weighting scheme known as Supervised Graph-based. This method could be used for text classification. To illustrate the idea of this method, let us consider that we have a document that is stemmed and all its stopping words removed. Then, we represent the document as an undirected co-occurrence graph. In the graph, each node is a word and each edge links the co-occurring terms. After that, we use Degree Centrality Measures (DCM) [Gol15], which is the number of connective edges to a node. For instance, if a node is connected to eight edges, then its DCM value is 8. The weight of each term is calculated by multiplying the Supervised Relevance Weight (SRW) with its DCM. After that, we

use this weight to convert the graph into a vector-based representation. To get the SRW of a word we need to go through several steps. As SRW takes into consideration three important factors: The concentration of a term in class as compared to other classes, the number of documents that contain this term in the class, and the number of documents in different classes that contain the term. First, we need to calculate the class relevance probability P of a term t in class C_i using the following function:

$$P(t, C_i) = \log_2(2 + \frac{\texttt{nbd}(C_i, t)}{max(1, \texttt{nbdN}(C_i, t))}) \times \log_2(2 + \frac{\texttt{nbd}(C_i, t)}{max(1, \texttt{nbnd}(C_i, t))}), where \tag{2.1}$$

$\texttt{nbd}(C_i, t)$ is the function that returns the number of documents in class C_i that do contain the word t. $\texttt{nbnd}(C_i, t)$ is the function that returns the number of documents in class C_i that do not contain the word t. $\texttt{nbdN}(C_i, t)$ is the function that returns the number of documents not in class C_i that do contain the word t. The above function P is computed for a term t in each class, then the class with the maximum relevance probability value of the term t is taken to be used later. Next, the density of each class for a term t is computed as follows:

$$\texttt{Class-Density}(t) = \frac{\texttt{nbd}(C_i, t)}{N_i}, where \tag{2.2}$$

C_i is a class, $\texttt{nbd}(C_i, t)$ is the function that returns the number of documents in C_i that do contain the word t and N_i is the total number of documents in class C_i. The average density of a term t is computed by dividing the sum of its densities in the

classes by the number of classes using this equation:

$$\texttt{Avg-Density}(t) = \sum_{i=1}^{C} \frac{(\frac{\texttt{nbd}(C_i, t)}{N_i})}{C}, where \tag{2.3}$$

C is the total number of classes.

Lastly, the SRW is calculated to determine the importance of the term in the classes by giving more weight to terms that facilitate the process of document discrimination from several classes.

$$\texttt{srw} = max(P(t, C_i)) \times \log_{10}(\frac{1}{\texttt{Avg-Density}(t)}) \tag{2.4}$$

The weight of each node is computed by multiplying the SRW with Term Weight (TW). Where TW is equal to the centrality score (DCM) for the term t in the graph. This method was used with the SVM classifier. It is found to be more accurate and outperforms the traditional TF-IDF weighting scheme and the unsupervised graph-based term weighting [SWLH16].

As discussed in [DA16], document presentation is an important step to be done before any classification. Many methods represent a document as a bag of words. Then, they assign a weight for each word based on the number of its occurrences in the document. Some modified methods join the number of occurrences with the location of occurrences of the term in the weighting scheme. Document classification with a weighted Frequency Pattern (FP) tree algorithm is one of them. The idea behind it is to treat a document as item sets. Each sentence constructs a single item set. In the regular FP tree algorithm, each item in the item set gets the unit value regardless if the sentence appears in the title or a paragraph. However, in the modified version

of the FP tree algorithm, each sentence's weight varies depending on the location of the sentence. Hence, to calculate the total weight for each word in the document we do the following. First, we multiply the number of occurrences of the term in a single section with the weight of a term in this section. Next, we repeat this process for all sections where the word appears. Then, we take the summation. This method proved to have more accuracy than conventional weighting schemes. Moreover, it considers the semantics of the word along with the number of occurrences.

Class-indexing-based term weighting for an automatic text classification scheme is a weighting method that could be used later with any classifier [RS13]. It is based on the conventional indexing scheme TF-IDF. However, some adjustment has been made as an improvement. The presentation of the data is divided into three main sections: Term index, document index, and class index. The VSM for each section contains both data and link subsections. In the term index, the data is the term itself. Also, it contains three links: one point to the next term, the second points to the class index, and the third points to the document index. The purpose of the second and the third links is to get the most appropriate class and document for the current term. In the document index, the data represent the document itself. Where the link points to the next document. In the class index, the data contain the class information. Moreover, it has two links: one point to the next class and the other points to the Class space density frequency (CSDF). The main differences between the traditional TF-IDF approach and this method are the addition of a class index and CSDF. CSDF is the main change that overcomes the drawback of giving a higher weight to the rare terms and assigning a lower one of the repeated terms in the traditional method. Instead, it gives equal weight for both. The main idea of

CSDF is to check if the class contains a specific term or not. If the answer yes, it gives the class weight 1 otherwise it gives 0. This process is repeated for each class in the training set. Next, it calculates the Class density (CD) for the class with weight 1. Which is equal to the number of documents in the class that contain the term (t) using this formula: $C_{cd}(t_i) = \frac{D_t}{N}$ where, $C_{cd}(t_i)$ is the CD of the term t in the class C, D_t is the number of documents in the class that contains the term t and N is the total number of documents in the class. Then, it computes the CSDF by summing the CDs for a specific term. Next, it gets the Inverse Class Space Density Frequency (ICSF) by using this equation: $\texttt{icsf} = \frac{1}{\texttt{csdf}}$. The total weight of each term is computed by multiplying TF with IDF and ICSF. The resulted weight could be used with any classifier like NB, centroid classifier, or SVM.

Per [MMM11], sentence ranking has been used as an effective technique for unstructured document indexing. The proposed indexing scheme depends on the weight of a sentence in which a term occurs. It could be calculated in two steps: In step one, we calculate the importance weight for each sentence in the document based on the position of the sentence. This can be calculated using a text summarization method called sentence ranking. Sentence ranking consists of four steps:

1. We enumerate the probability of a word in the sentence using the following equation:

$$P(w|S) = \frac{\texttt{TF}(w, S)}{|S|}, where \tag{2.5}$$

   $\texttt{TF}(w, S)$ is the function that returns frequency of a term in a sentence and $|S|$ is the number of words in a sentence.

2. We get the probability of the word in the document obtained as follows:

$$P(w|D) = \frac{\texttt{TF}(w, D)}{|D|}, where \tag{2.6}$$

   $\texttt{TF}(w, D)$ is the function that returns frequency of a term in a document and $|D|$ is the number of words in the document.

3. We get the comparison of the sentence to a document using relative entropy equation (KL-divergence) with the help of Equation 2.7:

$$KL_{(S)} = \sum_{w} P(w|S) \log \frac{P(w|S)}{P(w|D)}, where \tag{2.7}$$

   w is a word in a sentence (i.e., $w \in S$).

4. We get the importance of the sentence $I_{(s)}$, by taking the inverse of $KL - divergence$:

$$I_{(S)} = \frac{1}{KL_S} \tag{2.8}$$

In step two, if all terms in a single sentence have the same weight, we can get the weight for a term in document $W(t, D)$ by aggregating the weights of the term in all its occurrences in the document using this equation:

$$W(t, D) = \sum_{S_i}^{\text{if } t \in S_i} I_{(S_i)} \tag{2.9}$$

This method gets the weight for a specific term not just by considering the number of

occurrences, but it considers the semantics of the term in the document. The experimental test shows that this method gets a more accurate results than the conventional document indexing method.

The inverted index is one of the most well-known indexing techniques for unstructured documents [SSR12]. The main elements of this technique are two hash-indexed tables. The first table called the document table. It consists of two columns: *doc_id* and *posting_list*. *Doc_id* is the unique identity number for each document. *Posting_list* is the list of all terms (words) that occur in the document. The second table is the term table. As the previous table, it consists of two attributes: *term_id* and *posting_list*. *Term_id* gives a unique number for each term. The *posting_list* is the list of documents ids where the term occurs. Scanning these tables are fair enough to find where the term appears or which document contains certain words. The main disadvantage of this technique is missing the semantics of the word. The paper [SSR12] discusses another indexing technique known as a signature file. It gives each document a signature record in the database, where this record contains a fixed number of bits. All the bits are initially set to 0. If the document contains a term, it switches the bit for that term to 1. Usually, the number of terms exceeds the number of bits given. In that case, more than one word might map to the same bit, which is known as multiple-to-one mapping. This is considered one of the main drawbacks of this technique because it makes the query search more expensive.

The cube index for unstructured text has been introduced in [JR11]. The basic idea of traditional indexing is applying direct indexing. Direct indexing is divided into two parts: *Document_ID* and a list of all the words in that document. This method is not convenient when the number of documents is huge. Thus, it is not a

good idea to implement this methodology in big data applications. In contrast, Data cube [JR11] allows the document to be viewed in multiple dimensions (word, next word, document). It deals with the unstructured documents as a hierarchy, where the words are located in the lowest level of the hierarchy and the whole document is located at the highest level of the cube. In this way, it provides a mapping between words and documents. Users can fetch the relationship between the words easily. Also, the location of the word can be detected for semantic analysis. Moreover, the user can see the word from a different levels of abstraction. One strength of this approach is the ability to use data mining basic operations on the data cube like drill up/down, dice, and slice. These operations provide a variety of indexing services to the user to serve different demands.

2.3 Classification Approaches

Text classification of unstructured documents is the last step after stemming and indexing the document. A classifier is needed now to start the classification phase. Some literature review about well-known classifiers is described below.

SVM is one of the most known supervised ML methods used for the text classification purpose [KBLK10]. Mainly it is a binary classifier with two classes in the training set. However, it could be used with multi-classes training set. To start the SVM classifier, training documents should be represented as VSM and weighted using the TF-IDF scheme. SVM creates a hyper-plane that separates the classes of the training set and ensures the largest margin between them. Every document in a class is represented as a point in a multi-dimensional space. The main strength of SVM is its outstanding performance in comparison to other ML classifiers. However, it is considered a

complex classifier and consumes time and memory from the machine [MM09] [SJV17].

DT is an ML classifier that helps in classifying a document automatically [BK19]. The structure of this methodology is to build a hierarchical decomposition of training set's classes using tree structure. Each branch reflects a class with a class name labeled in the leaf of that branch. Each internal node in the tree represents a term or a feature. The base node or the root of each branch is the feature with the highest weight of that branch's category. When testing a document, the method starts at the root of the tree. Then, it moves down in the tree based on the matching feature until it reaches the most suitable category or leaf. Decision trees are considered very simple to implement and understand which is one of the main advantages of this tool in comparison to other decision support tools.

Rocchio Algorithm is a supervised ML classifier [KBLK10]. The algorithm calculates the centroid for each class in the training data and makes it the key for this class. When testing a new document, it computes the similarity measure between a given document and each centroid for each class in the training set. Finally, it assigns the document to the class with the minimum distance. One strength of this algorithm is its ease to implement. However, the classification accuracy result is relatively low [BK19].

Artificial Neural Network (ANN) is a supervised ML classifier [BK19]. It mainly consists of neurons. Neurons are compilation nodes which are considered as the main engine of this classifier. These neurons are placed in three layers: an input layer, an output layer, and hidden layers. The number of terms in training data makes the number of neurons in the input layer, where the number of categories

makes the ones in the output layer. Each node from the input layer is linked to its appropriate category node with a specific weight. This link goes through several hidden layers. The activation function is used to get the right linkage between these nodes. Many activation functions could be used for this purpose like binary step and sigmoid functions [Sha17]. As the number of hidden layers increases the efficiency of this classifier increases too. One advantage of ANN is its ability to handle effectively very large documents. However, it consumes a large space of memory and CPU. Besides, it is considered one of the complex classifiers [KBLK10].

K-NN is one of the widely used supervised ML algorithms for text classification [KBLK10]. The main idea behind this algorithm is to find the most similar category to the document based on the closest distance. The decision about the value of K for K-NN is crucial because it affects the accuracy of the classification result. The number K is the number of classes in the training data. The research done in [RS17] found that when the number of K increases the accuracy increases too. Classes of the training set are taking place in multi-dimensional feature space and conduct a region. The similarity between the test document and a class is measured using Euclidean Distance, which computes the distance between the vectors of the new document and each class of the training set. The nearest distance is selected as the category of the new document. Many challenges are associated with K-NN like the suitable number of K and the distance measures. However, the algorithm is still doing well with a scalable number of documents.

NB is a document classifier considered as a supervised ML algorithm [KBLK10]. In this classifier, each feature assumed to be independent even if there are similar features. NB classifies a new document based on Bayes Theorem [KBLK10]. We

apply the Theorem for each class, then the class with the highest probability value is selected as the category of the document. NB has been used widely because it can handle complex situation and it works well even with a small training set. On the other hand, classification performance could be inefficient if the features are relevant which commonly occurs in a real situation. To overcome this drawback many types of research have been carried. For instance, combining NB with the SVM approach leads to better performance. The major use of NB is in spam filtering and web content classification.

A comparison of text classification methods is a widely-discussed topic. One comparison has been made to see the most accurate tool [BEMS17]. The experiment carried considering the three important factors that affect any classification result: Stemming, document indexing, and classification techniques. The paper combines different methods to get the best indexing and classification result. The paper adopts the following steps: First, it extracts terms which include taking words one by one, then remove the stopping words. Second, it performs the stemming of words if necessary. Many stemming algorithms were tested like Lovins, iterated Lovins, snowball, and Null. Next, it represents a document and weights its terms. The representation method used is VSM, then TF-IDF weighting scheme was implemented. Finally, it classifies documents by using a classifier. Several classifiers have been tested like Bayesian Network, SVM, Random Forest (RF), Simple Logistic Function (SLogiF), Bernoulli NB and, Vote. The latter classifier combines Naive Bayes Multivariate Bernoulli (NBMU), SVM, RF, SLogiF, and NB to improve the accuracy result. The experimental study [BEMS17] carried into two data sets. The first data set is founded by the set of British Broadcasting Corporation (BBC) news articles divided into five

categories with a total of 2225 articles. The second data set is BBC sports articles divided into five categories with a total of 737 articles. In both data sets, they found that the Vote classifier gets the most accurate results. Then they compare the best matching stemming approach with Vote. They found that the Lovins stemmer is the best for the first data set. However, with the second data set Snowball stemmer performed better.

Chapter 3

Theoretical Background

In Chapters 1 and 2, I introduced the problem of indexing and classifying unstructured documents. I listed the main factors that affect classification results. I explored some recent research papers related to this area. In this Chapter, I introduce the theoretical background of our work consisting of PFA and BDDs. The structure of our system depends mainly on PFA and its main operations implemented using BDD. In the first stage of our classification system, we convert a given dictionary document into a BDD. With the help of this BDD, we build the BDD corresponding to the document to be classified. The classified document's BDD is transformed into a vector that serves as a signature for this document. As I discuss in Section 3.3, the operations on products and families are performed through Boolean operations on BDDs or their signatures. I use Boolean operations on different document's signatures to get the commonality between documents or to get a family of documents. Later, a signature for a document or a family of documents is used to get the final decision of a classification.

3.1 Binary Decision Diagram

A BDD is a rooted directed acyclic graph used to represent Boolean functions [TLH09]. It consists of two terminal nodes 0 and 1. Decision nodes are considered variables with two possible edges: high and low. A low edge represents the absence of the variable and encoded as 0. However, a high edge represented as 1 and indicates the presence of the variable [TLH09]. Many Boolean operations could be performed on variables inside one BDD and then extended to BDDs. Some of the main Boolean operations that might be implemented on BDD are logical *and* and logical *or*. BDD is known as Reduced Ordered Binary Decision Diagram (ROBDD) if two conditions hold. The first condition is when no decision node points to the same node from its high and low edges. The second condition is when any redundancy from sub-graphs is removed by merging nodes with the same label that points to the same node in their high and low edges. The ROBDD of the function $f(x_1, x_2) = x_1 \wedge x_2$ is illustrated in Figure 3.1. Due to the implementation efficiency of ROBDD, we adopt ROBDD throughout this work.

BDD is used to represent a set. Each element in the universal set under consideration is taken as a decision node. In a set, each element should appear only once. No duplication is allowed. However, as I discuss in Section 3.3.2, BDDs could represent a bag of elements where each element can occur more than one time.

3.2 Product Family Algebra

In this section, I introduce a general PFA. I explore some of the popular PFA models and how we implement them using BDD. Then, we get some insight about PFA

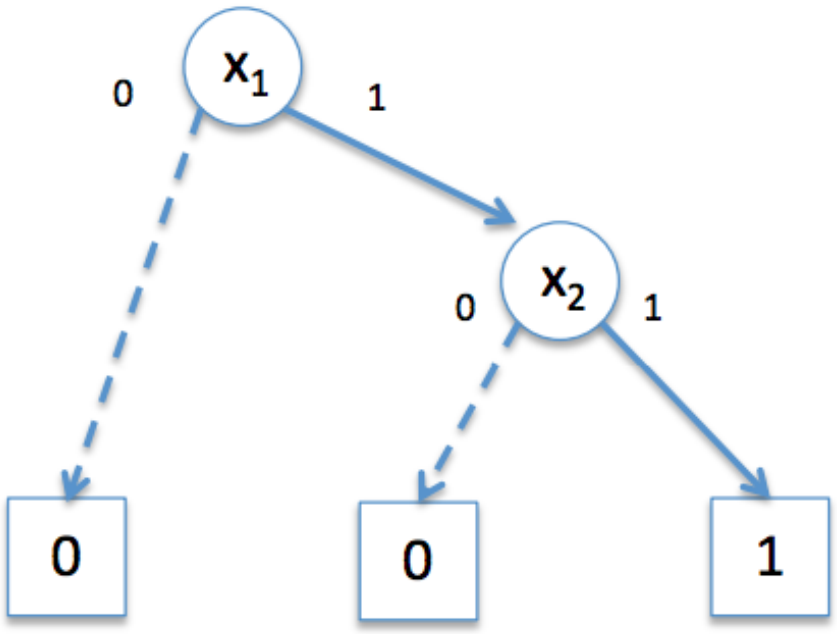

Figure 3.1: ROBDD of the function $f(x_1, x_2) = x_1 \wedge x_2$

relations and their main results. Product and feature are essential components in the construction of PFA, I need to define them prior to tackling PFA. Below are definitions of semiring, product, and feature, receptively.

Definition 3.2.1 (e.g., [HKM11]). *A semiring is a quintuple* $(S, +, 0, \cdot, 1)$ *such that* $(S, +, 0)$ *is a commutative monoid, and* $(S, \cdot, 1)$ *is a monoid such that* $\cdot$ *distributes over* $+$ *and* 0 *is an inhalator. The semiring is commutative if* $\cdot$ *is commutative and it is idempotent if* $+$ *is idempotent.*

Definition 3.2.2 (e.g., [HKM11]). *Assume a commutative idempotent semiring* A. *An element* $a \in A$ *is called a product if it satisfies the following laws:*

1. $\forall(b \mid b \in A : (b \leq a) \Rightarrow (b = 0 \vee b = a))$,

2. $\forall(b, c \mid b, c \in A : (a \leq b + c) \Rightarrow ((a \leq b) \vee (a \leq c)))$.

In particular, 0 is a product. A product a is proper if $a \neq 0$.

Definition 3.2.2.(1) indicates that a product could not be divided into subfamilies. It is always an atomic element. Definition 3.2.2.(2) indicates that if a is a sub-family of a family consisting of both families b and c, then it should be a sub-family for at least one of them. There is a special product which is a family consisting of only one product with no feature denoted by $1 = \{\emptyset\}$. On the other hand, 0 is an empty family with no products.

Definition 3.2.3 (e.g., [HKM11])**.** *An element a is called a feature if it is a proper product different from 1 satisfying the following laws:*

1. $\forall(b \mid b \in A : (b \mid a) \Rightarrow (b = 1 \vee b = a))$,

2. $\forall(b, c \mid b, c \in A : (a \mid (b \cdot c)) \Rightarrow ((a \mid b) \vee (a \mid c)))$,

where the divisibility relation $\mid$ is given by $x \mid y \iff \exists(z \mid z \in A : y = x \cdot z)$

The above conditions indicate that a feature could not be divided into sub-features. However, Definition 3.2.3.(1) says that if b is essential for constructing a then, b is either the special product 1, or a itself. Definition 3.2.3.(2) states that if a is a mandatory feature for product families b and c together, then it should be a mandatory feature to at least one of them.

Definition 3.2.4 (e.g., [HKM11])**.** *A product family algebra $(S , + , 0 , \cdot , 1)$ is a commutative and idempotent semiring in which 1 is a product. Its elements are called product families or briefly families.*

The carrier set S is the set of families of products. The operation $+$ is a binary operation on S and it is interpreted as a choice between its operands. The constant 0 is the identity element of $+$. The operation $\cdot$ is a binary operation on S and it is interpreted as a mandatory presence of its operands in the family corresponding to the application of $\cdot$ on its operands. The constant 1 is the identity element of $\cdot$ operation. As I discussed earlier, the classification system that I propose requests a dictionary and testing document each time it performs a classification. Each document can be treated as a family of products, where each section in the document is considered as a product. From a different perspective, a document could be viewed as a product, where a group of documents build up a family. On the other hand, a dictionary is a family with one product.

In [HKM11], we found that PFA has a set model and a bag model. In our system, we are interested in implementing PFA as set within the set based model (PFS), or within the bag based model (PFB). Hence, I need to show how we implement the operations $+$ and $\cdot$ in the two models using BDD.

In the following, we construct concrete models from a given set $\mathbb{F}$ of features. In our work the dictionary of words gives us the set $\mathbb{F}$.

3.2.1 Set Model Implementation Using BDD

In the set model, a product family is a set of features. A family is a set of sets; a set of products. A family of all possible products is considered as a set of all subsets or the power set denoted as $\mathbb{P} = \mathcal{P}(\mathbb{F})$. The representation of a family in the set model could be translated into BDD easily [Alt10]. Each product of the family transformed into a single BDD with several variables reflect the product's features. Each element

of $\mathbb{F}$ is considered as a variable (i.e., every word in the dictionary is considered as a variable). The implementation of the operation $\cdot$ is applying a set union between two families. We need to apply + between all products (BDDs) of these families. For example, if we have two families V and H, each of which has only one of products p_1 and p_2, respectively. Hence, $\{p_1\} \cdot \{p_2\} = \{(p_1 \cup p_2)\}$. Therefore, the operation $\cdot$ is feature merging operator on products that could be defined as follow:

$\cdot : \mathcal{P}(\mathbb{P}) \times \mathcal{P}(\mathbb{P}) \to \mathcal{P}(\mathbb{P})$

$P \cdot Q = \{p \cup q \mid p \in P, q \in Q\}$

The operator + is a union between product families. For instance, if we apply + between previous families, we get $\{p1, p2\}$. The + on set model is choice among families defined below.

$+ : \mathcal{P}(\mathbb{P}) \times \mathcal{P}(\mathbb{P}) \to \mathcal{P}(\mathbb{P})$

$P + Q = P \cup Q$

After defining the operations of feature algebra, we can construct the structure of the set model as follows: given a set of features $\mathbb{F}$. Let $\mathbb{P} = \mathcal{P}(\mathbb{F})$, then $\mathbb{P}FS =_{df} (\mathcal{P}(\mathbb{P}), +, \emptyset, \cdot, \{\emptyset\})$.

Examples 3.2.1, 3.2.2, and 3.2.3 illustrate the idea of representing a set on BDD more clearly.

Example 3.2.1. *Let A be a family with one product p_1. Let $\mathbb{F}$ be a set of features. Hence, a feature here represents a word. For example, $\mathbb{F} = \{a, b, c, d, e, f\}$. Let p_1 is a product on $\mathbb{F}$ (i.e., $p_1 \in \mathcal{P}(\mathbb{F})$) and $p_1 = \{a, b\}$. In this case, p_1 is a document formed by two words a and b. The product p_1 is represented by the Singleton set using the BDD of Figure 3.2.*

Example 3.2.2. *With the same setting of $\mathbb{F}$ as in Example 3.2.1, we use BDD to*

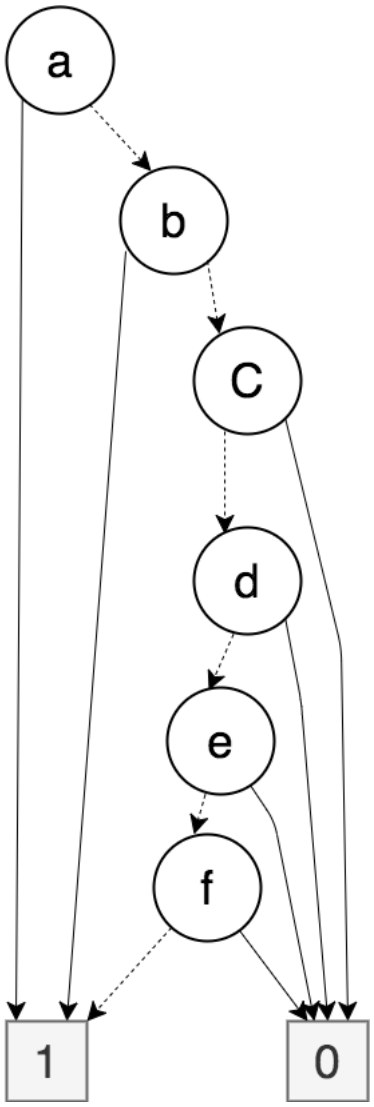

Figure 3.2: The BDD representing p_1

represent the product p_2 that is a product on $\mathbb{F}$ and $p_2 = \{c, d, f\}$. In this case, p_2 is a document formed by three words c,d, and f. The representation of p_2 as set using BDD is shown in Figure 3.3.

Example 3.2.3. *Let Z be a family with one product p_3. Z is a document resulting from merging the previous documents $Z = A \cdot B$. To Perform the operation on*

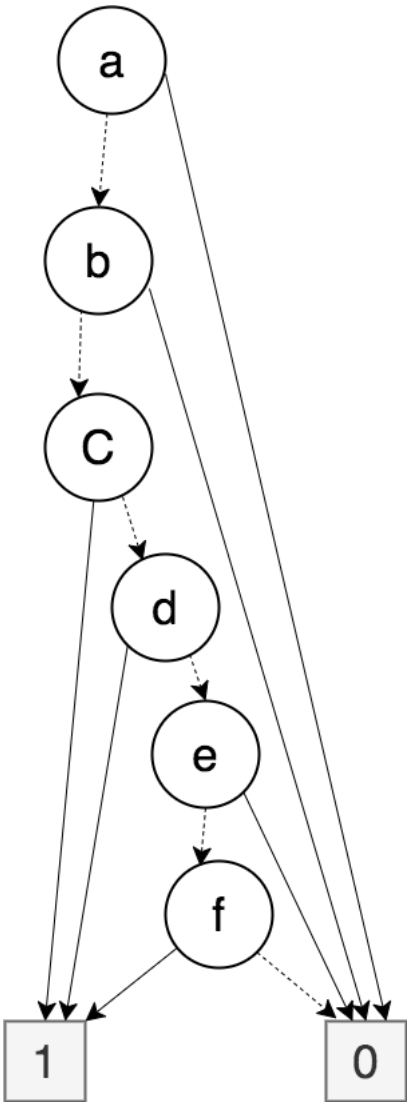

Figure 3.3: The BDD corresponding to p_2

these families, we conduct a union between the products of the families. We get $Z = \{(p_1 \cup p_2)\}$. *The family* Z *is the singleton set formed by the product* $p_3 = \{a, b, c, d, f\}$. *We can represent* p_3 *as set using the BDD given in Figure 3.4.*

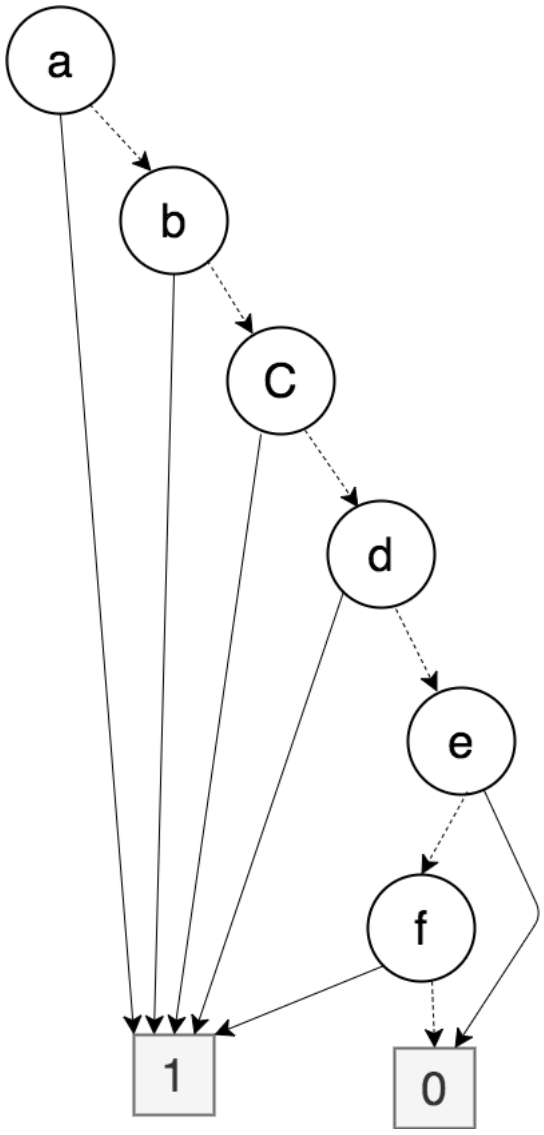

Figure 3.4: The BDD corresponding to product p_3

3.2.2 Bag Model Implementation Using BDD

As mentioned earlier in this Chapter, the bag model allows representing the re-occurrence of elements. In this case, a product is a bag of elements of $\mathbb{F}$ (Rather

than a set of elements of $\mathbb{F}$). The regular implementation of ROBDD does not directly support the bag model. To transform the bag model into real implementation, we added the occurrence level in BDDs as given in [Alt10]. In this case, a BDD saves each occurrence of an element by encoding it as a bit string. A maximum number of occurrences of elements in a BDD needs to be defined before. For example, let us assume that the maximum number of occurrences is five. Then we need three bits to encode it. If the element x appears in the product p_1 four times, this is encoded as 100 in the BDD. However, if the element y appears in the product p_1 six times, it is encoded as 101. The + operation is performed as the same as in the set model. However, operation $\cdot$ is performed slightly differently. We need to sum the number of occurrences of an element that shows up in both BDDs and encode the total in the new BDD. Examples 3.2.4, 3.2.5, and 3.2.6 illustrate the idea more clearly.

Example 3.2.4. *Let X be a family with one product p_1. Let $\mathbb{M}(\mathbb{F})$ be the set of bags (multi-sets) formed from the features of $\mathbb{F}$. Hence, a feature here represents a word. Let us take $\mathbb{F} = \{a, b, c\}$. Let p_1 be a product on $\mathbb{F}$ (i.e., $p_1 \in \mathbb{M}(\mathbb{F})$) and $p_1 = \{(a, 1), (b, 2), (c, 6)\}$. In this case, p_1 is a document formed by three words a,b, and c, where a occurs one time, b occurs two times and c occurs six times. Let us assume that the maximum number of occurrences of any element of $\mathbb{F}$ is seven. Then we need three bits to encode each occurrence. Each bit makes a level in the BDD. The occurrence of feature a is 1 and encoded as 001. To encode this occurrence, we need to select $l1$, but not $l2$ and $l3$. The occurrence of feature b is 2 and encoded as 010. So, we need to select $l2$, but not $l1$ and $l3$. The occurrence of feature c is 6 and encoded as 110. To encode this occurrence, we need to select $l2$ and $l3$, but not $l1$. Representation of p_1 as a bag using BDD is given in Figure 3.5.*

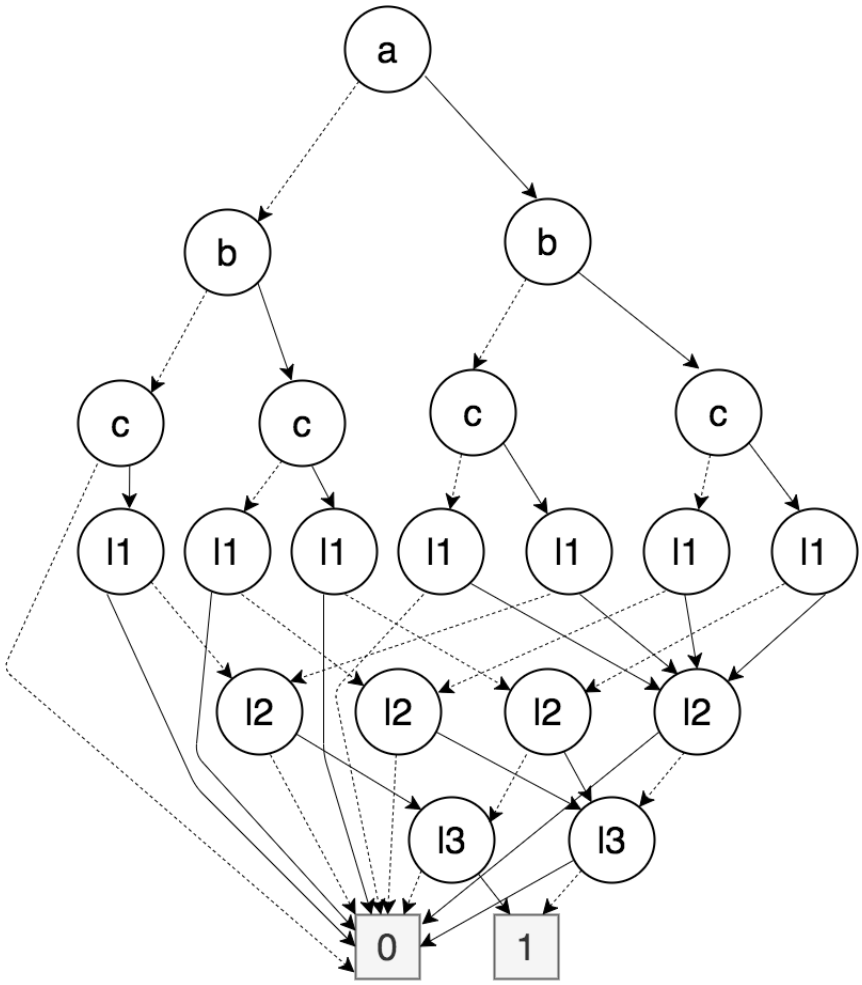

Figure 3.5: Bag representation of p_1 of family X using BDD

Example 3.2.5. *Similarly, as in Example 3.2.4, we use BDD to represent a family Y as a bag with one product p_2. Let $p_2 = \{(a, 2), (b, 3)\}$. In this case, p_2 is a document formed by two words a and b. However, the occurrence of each feature needs to be encoded. Let us assume that the maximum number of occurrences is seven. The occurrence of feature a is 2 and encoded as 010. To encode this occurrence, we need to select only $l2$. The occurrence of element b is 3 and encoded as 011. To encode this occurrence, we need to select $l1$ and $l2$, but not $l3$. Absence of feature c is encoded*

as 000 in the BDD. Representation of the product p_2 as a bag using BDD is given in Figure 3.6.

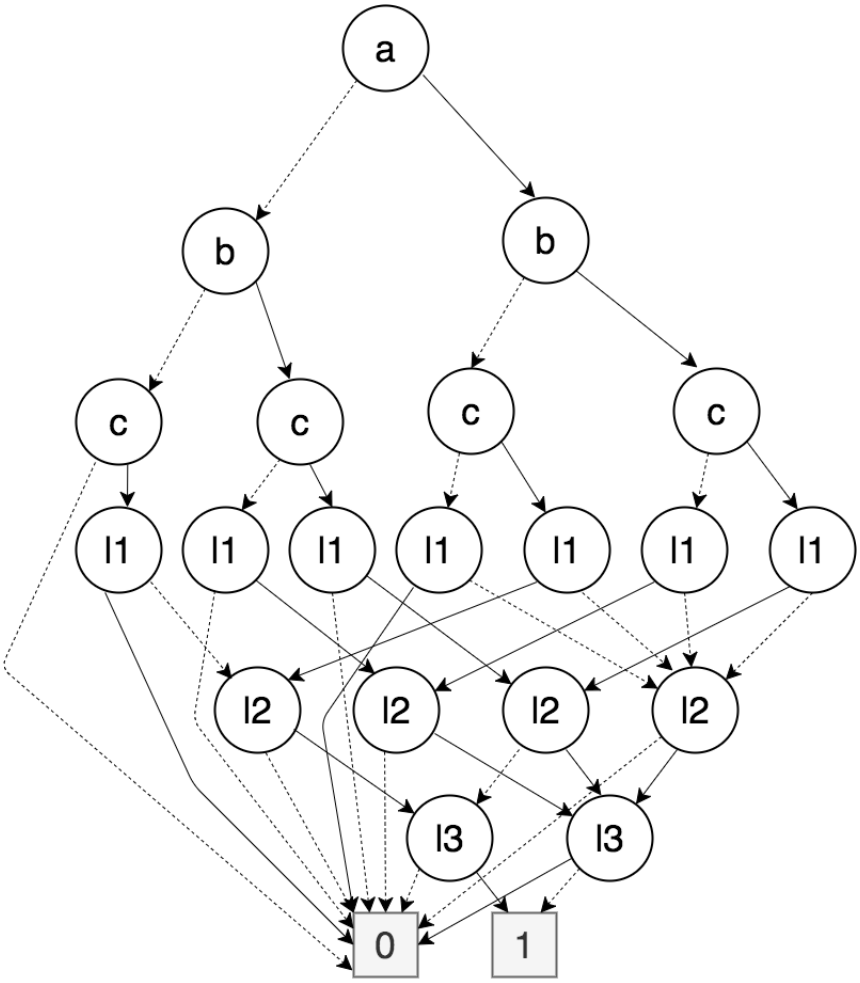

Figure 3.6: Bag representation of p_2 of family Y using BDD

Example 3.2.6. *Let Z be a family with one product p_3. Z is a document resulting from merging the previous documents $Z = X \cdot Y$. Thus, we need to sum the number of occurrences for each feature. Then encode the new total in p_3. Family $Z = \{\{(a,3),(b,5),(c,6)\}\}$. The BDD representing the product p_3 of the family Z as a bag is given in Figure 3.7.*

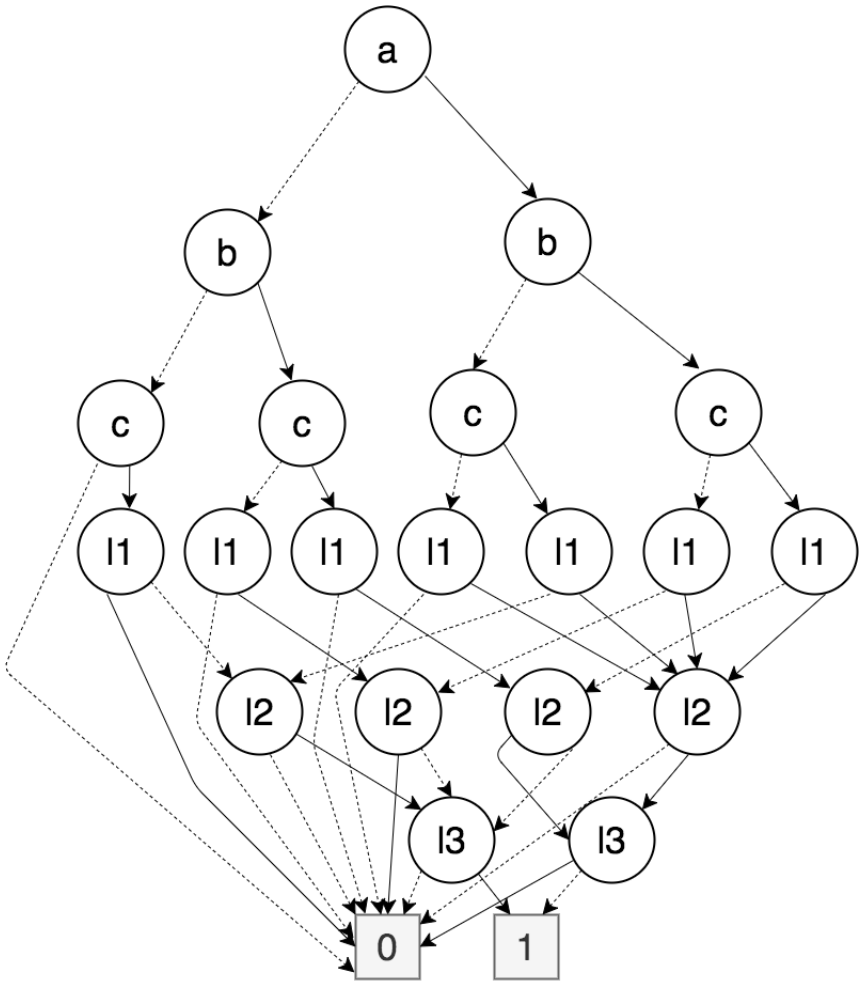

Figure 3.7: Bag representation of p_3 of family Z using BDD

3.3 Elements of PFA Theory

3.3.1 Relations Defined on PFA Theory

In this sub-section, I explain three relations on product families: Sub-family, refinement, and implication.

3.3.1.1 PFA Sub-Family Relation

As found in [HKM11], the sub-family is a relation on families. A family of products can be partitioned into a number of smaller families. These families include part of the products. The formal definition of sub-family is shown in Definition 3.3.1:

Definition 3.3.1 (e.g., [HKM11]). *Let $(S\ ,\ +\ ,\ 0\ ,\ \cdot\ ,\ 1)$ be a PFA. Let a and $b \in S$, we say that a is a sub-family of b and denoted by $a \leq b$, if and only if $a+b=b$. where $\leq$ is partial order (i.e., a reflexive, antisymmetric, and transitive relation) called the natural order on S. It has 0 as its least element.*

The sub-family relation is commonly used in our classification system. A document could be considered as a sub-family of another document if it contains part of its contents. In addition, each section in the document could be treated as a sub-family of the main document.

3.3.1.2 PFA refinement relation

One of the important relations on PFA is the refinement relation [HKM11]. For example, in a phone company, they released a new phone called $phone_2$ that has the exact features of $phone_1$, but with additional features. $phone_2$ is a refinement of $phone_1$. Translating this to PFA we can get Definition 3.3.2:

Definition 3.3.2 (e.g., [HKM11]). *Let $(S\ ,\ +\ ,\ 0\ ,\ \cdot\ ,\ 1)$ be a PFA. Let a and $b \in S$, we say that a refines b denoted by $a \sqsubseteq b$ and defined as:*

$a \sqsubseteq b \iff \exists(c \mid c \in S : a \leq b \cdot c)$;

it is a preorder (i.e., reflexive and transitive).

The above definition illustrates that family a is a refinement of family b if and only if every product of a has all the features of some product of b plus additional ones. There are many results related to this relation.

Proposition 3.3.1 (e.g., [HKM11]). *If a, b, c are elements of product family algebra and p is product then:*

1. $b|a \Rightarrow a \sqsubseteq b$
2. $a \leq b \iff a \sqsubseteq b$.
3. $a \cdot b \sqsubseteq b$.
4. $a \sqsubseteq a + b$.
5. $a \sqsubseteq b \Rightarrow a + c \sqsubseteq b + c$.
6. $a \sqsubseteq b \Rightarrow a \cdot c \sqsubseteq b \cdot c$.
7. $a \sqsubseteq 0 \iff a \leq 0$.
8. $0 \sqsubseteq a \sqsubseteq 1$.
9. $a + b \sqsubseteq c \iff a \sqsubseteq c \wedge b \sqsubseteq c$.
10. $p \sqsubseteq a + b \iff p \sqsubseteq a \vee p \sqsubseteq b$.

The proof of the proposition can be found in [HKM11].

This proposition shows ten properties that hold for the refinement relation in PFA. Proposition 3.3.1.(1) shows that if family b is essential to construct family a, then a is a refinement of b. It is like extending family b to a. A reverse implication could be right if all products of b are used to build a and a has more features. Proposition 3.3.1.(2) shows that if family a is sub-family of family b by then it should be a refinement of b. Proposition 3.3.1.(3) says that merging family a and b is a refinement of family b. Proposition 3.3.1.(4) states that family a is a refinement of $a + b$. Proposition 3.3.1.(5,6) say that if family a is a refinement of family b then combining family c to family a and b using $\cdot$ or $+$ operations will not change the situation. Hence, $a + c$, or $a \cdot c$ is a refinement of $b + c$ or $b \cdot c$ respectively. Proposition 3.3.1.(7) says that family a is a refinement of the empty family if it

is less than or equal to it. Proposition 3.3.1.(8) shows that an empty family of products refines family a and family a refines the empty product represented as 1. Proposition 3.3.1.(9) states that family $a + b$ refines family c if and only if family a refines c and family b refines c. Proposition 3.3.1.(10) says that product p refines family $a + b$ if and only if it is a refinement of one of them. For more details on the results of Proposition 3.3.1, I refer the reader to [HKM11].

Proposition 3.3.2 (e.g., [HKM11])**.** *If a product family algebra contains a $\leq$-greatest element T, then:*

$a \sqsubseteq b \iff a \leq b \cdot T \iff a \cdot T \leq b \cdot T.$

The proof of the proposition can be found in [HKM11].

Proposition 3.3.2 brings the property of the refinement and subfamily relationships when we involve the $\leq$-greatest element. At PFS, the $\leq$-greatest element T is the singleton set that contains $\mathbb{F}$ (i.e., $T = \{\mathbb{F}\}$). In our case, the dictionary contains the set $\mathbb{F}$ of the words (i.e., features) that could be used to form documents. Therefore, T is the family formed by the singleton set of the document that contains all the words. Of course, we can have several singleton families of documents that can be represented by T. This is due to the fact that we abstract (do not consider) the order in which the words appear in the document.

In the bag model, if we fix the maximum number of occurrences to a number k, then instead of taking $\mathbb{F}$ as the product to be in the set that forms T, we will take the set $\mathbb{B} \stackrel{\text{def}}{=} \mathbb{F} \times \{k\} = \{(x, k) \mid x \in \mathbb{F}\}$. In other terms, $\mathbb{B}$ is the set of the elements of $\mathbb{F}$ each occurring k times.

Proposition 3.3.2 says that if family a refines family b, then appending the greatest element T to both preserves the subfamily order but not necessarily the refinement.

The implementation of the refinement relation in our proposed system is straightforward. A document refines another document if it contains all its words and more. For instance, if we treat each section in a document a family with one product then a document is a refinement of its sections.

3.3.1.3 PFA Implication Relation

One of the relations that need to be fetched when discussing PFA is the implication relation [HKM11]. The following definition introduces it.

Definition 3.3.3 (e.g., [HKM11])**.** *Let $(S\ ,\ +\ ,\ 0\ ,\ \cdot\ ,\ 1)$ be a PFA. Let $a, b, c, d \in S$ with a finite number of products and p is a product. We define the implication relation in a family-induction style:*

1. $a \xrightarrow{p} b \iff (p \sqsubseteq a \to p \sqsubseteq b)$.
2. $a \xrightarrow{c+d} b \iff a \xrightarrow{c} b \wedge a \xrightarrow{d} b$.

Definition 3.3.3.(1) shows that if product p requires a then it requires b as well. This is equal to say if product p is a refinement of a then it implies that it is a refinement of b as well. Definition 3.3.3.(2) shows that if product $c + d$ together requires a then b this equal to say that c needs a and b and d requires them as well.
Implication relation is very useful to put some restrictions on the family. Example 3.3.1 explains the relation more clearly.

Example 3.3.1. *Let us assume that we have a family A, which consists of all Business documents. We would like to add a restriction to classify a document as belonging to family A. We can say, if the document contains the word "Business", then it needs*

to have the word "Organization" as well. We can specify this requirement using the implication relation by writing Business $\xrightarrow{A}$ *Organization.*

There are results derived from the connection between PFA relations shown below:

Proposition 3.3.3 (e.g., [HKM11])**.** *Let (S , + , 0 , · , 1) be a PFA. Let* $a, b, c \in S$ *with a finite number of products then,*

1. $\xrightarrow{a}$ *is preorder.*
2. *Let* $b \sqsubseteq c$*, then* $(c \xrightarrow{a} d) \Rightarrow (b \xrightarrow{a} d)$ *and* $(d \xrightarrow{a} b) \Rightarrow (d \xrightarrow{a} c)$*. In particular,* $b \sqsubseteq c \Rightarrow (b \xrightarrow{a} c)$.
3. *Let* $b \leq c$*, then* $(c \xrightarrow{a} d) \Rightarrow (b \xrightarrow{a} d)$ *and* $(d \xrightarrow{a} b) \Rightarrow (d \xrightarrow{a} c)$.

The proof of the proposition can be found in [HKM11].

Proposition 3.3.4 (e.g., [HKM11])**.** *Let (S , + , 0 , · , 1) be a PFA. Let* $a, b, c, d, e \in S$ *with a finite number of products then,*

1. $b \xrightarrow{a} b + c$.
2. $b.c \xrightarrow{a} b$.
3. $(b \xrightarrow{a} c) \Rightarrow (b \xrightarrow{a} c + d)$.
4. $(b \xrightarrow{a} d) \Rightarrow (b.c \xrightarrow{a} d)$.
5. *If p is product, then* $(b \xrightarrow{p} c) \Rightarrow (b + d \xrightarrow{p} c + d)$.
6. $((a \xrightarrow{e} b) \wedge (c \xrightarrow{e} d)) \Rightarrow ((a.c \xrightarrow{e} b) \wedge (a.c \xrightarrow{e} d))$
7. $(a + b \xrightarrow{e} c) \iff ((a \xrightarrow{e} c) \wedge (b \xrightarrow{e} c))$.

The proof of the proposition can be found in [HKM11].

Proposition 3.3.4.(1) shows that if family a requires b then it could be extended to require $b + c$. Proposition 3.3.4.(2) indicates that if the construction of family a

requires $b \cdot c$ then by default it requires b alone. This applied for c also. Proposition 3.3.4.(3,4 and 5) indicate that requirement relation could be extended on one side without affecting the other side. Proposition 3.3.4.(6) says that pushing all the requirements of the family into one relation should include all implication members. Proposition 3.3.4.(7) says that if family e requires $a + b$ then it requires c that means both a and b require c in construction of family e. For more details on the results in Proposition 3.3.4, I refer the reader to [HKM11].

The implication relation is essential for the construction of our classification decision rules. Decision rules help in deciding if a given document belongs to a certain category or not. Also, we can request that more than one decision rule should be met to classify a given document.

Chapter 4

Methodology and Tool Design

In this Chapter, I explore our approach to document classification. The proposed approach is used to build a tool for document classification that we named SDCS. I present the architectural and detailed design of the tool. The main goal of our tool is to give the opportunity for the user to index and classify a given document using two independent modules. The first option, indexing and classifying using rule-based approach applied to the signature. The second option, using ML classifiers. The design and the main functions of each module are discussed in detail throughout this Chapter.

4.1 Architectural Design

SDCS can be viewed as a layered architecture as in Figure 4.1. Mainly it consists of three layers. Modification or changes in one layer could be done independently without the need to go through other layers. In the end, these layers were connected to get our full application.

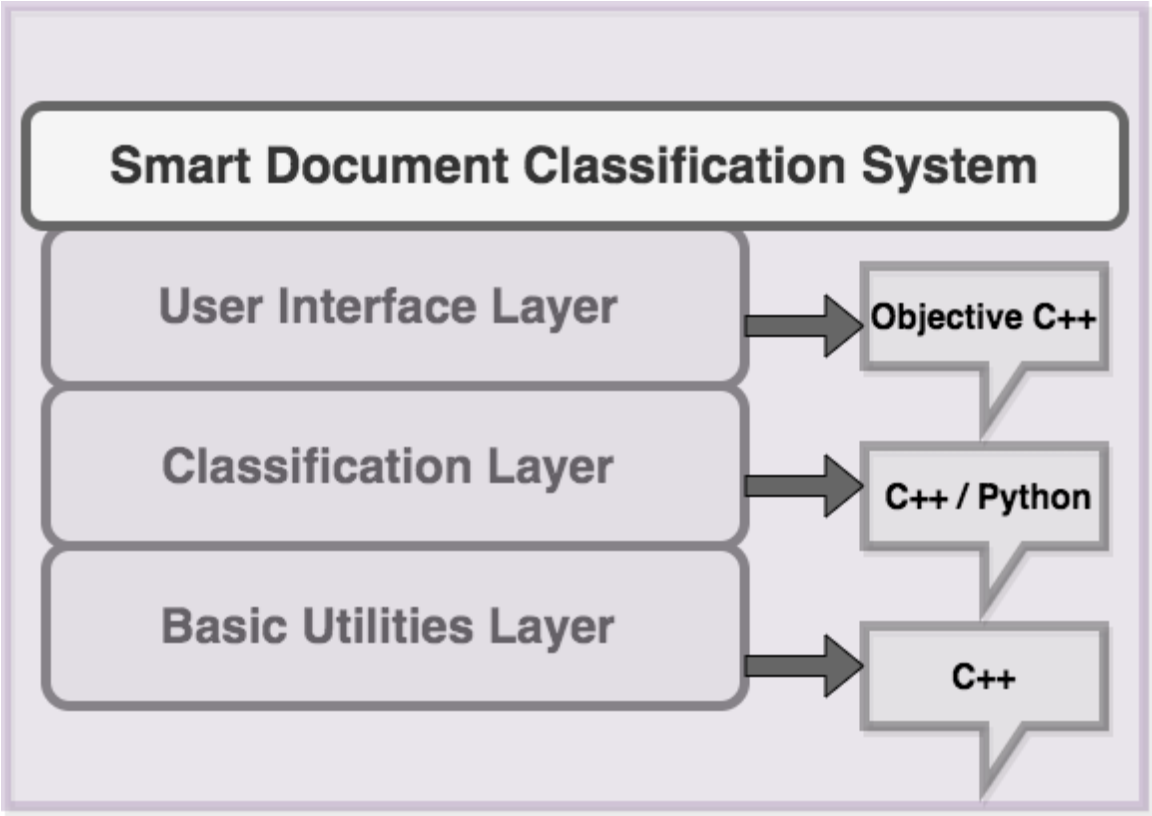

Figure 4.1: Architectural Design

The user interface layer is the top layer in our model. It allows users to interact with our system and provides a variety of services to users. It enables system users to classify new documents using our proposed approach or using a given ML classifier among SVM, NB, K-NN, and DT. In addition, all the required directories and files are created and set up within this layer.

The classification layer is the pulp of our tool. Two core independent approaches make up this layer. First, ML-based classification approach. Second, the rule-based approach applied to the signature. Each of which built up from several modules.

Basic utilities layer is the lowest layer in our tool, where all the necessary preprocessing

functions on the document to be classified are performed. This layer consists of the functions that the program needs prior to the classification of a given document.

4.2 Detailed Design

In this section, I give a detailed explanation of each layer. Starting from the user interface layer down to the basic utilities layer. A detailed view of the system design is illustrated in Figure 4.2.

4.2.1 User Interface Layer

The user interface layer is consisting of two main modules: the user interface module and setup module. The user interface module is serving as a median between our underlying structure and users. This module provides a variety of options to the users. We can present this module using system windows. The main interface of our system is shown in Figure 4.3. As clearly shown in the previous Figure, one can say we divide our main interface window into two sections: *Build Signature* and *Classifying*. The *Build Signature* section gives the user two options to choose from. First, indexing new document by building new signature. Second, indexing by using existing signatures. This option uses Boolean operations (Union, Intersection) between existing documents signatures to create a new signature. On the other hand, *classifying* section enables the user to classify a new document using two options. Option one, rule-based approach applied to the signature. Which is depends on BDDs. Option two, classify a document using one of the given ML classifiers. In our system, we provide four ML classifiers: SVM, NB, K-NN, and DT. This module has

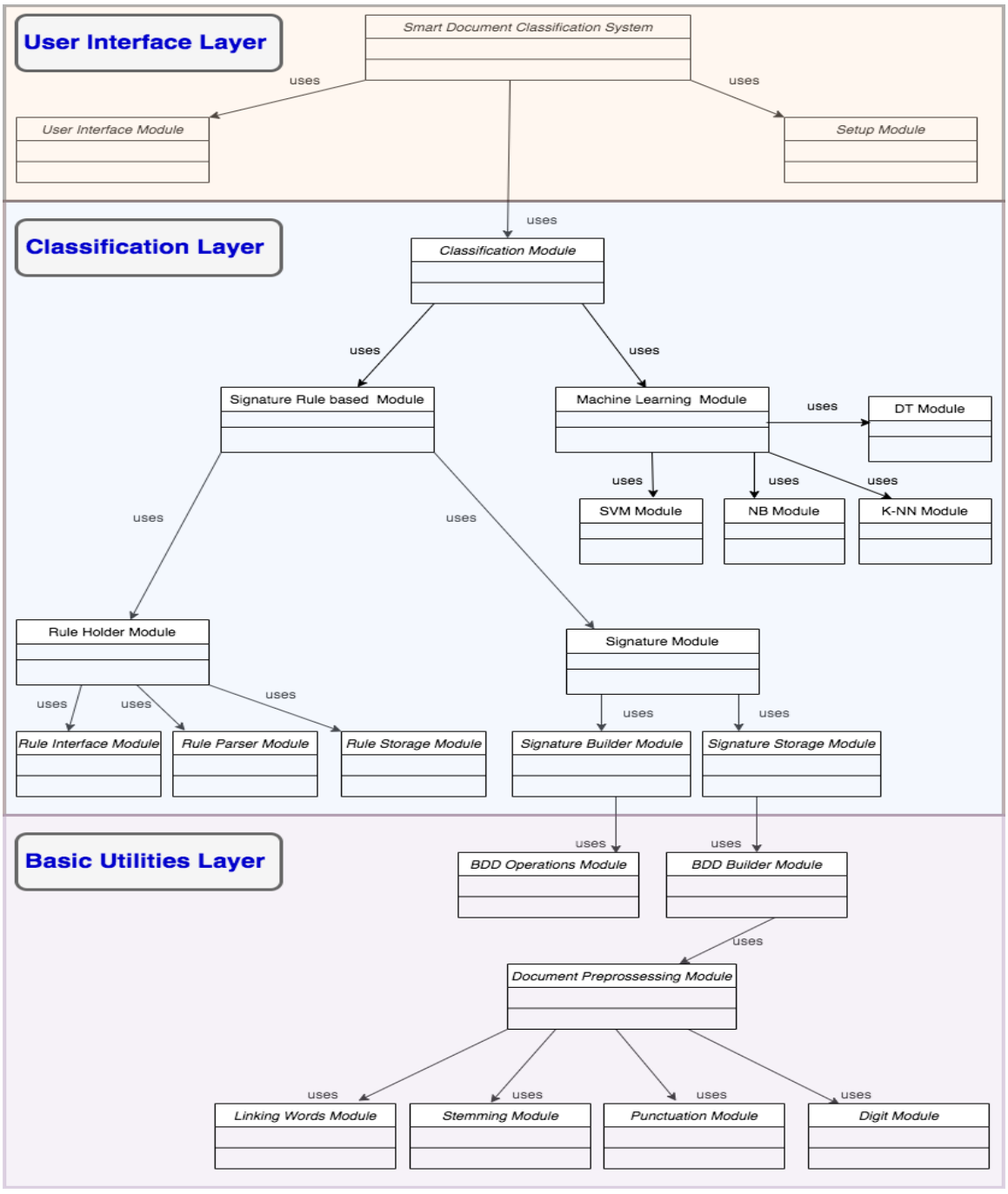

Figure 4.2: System Design

been written using objective C++ language and linked to other layers. It is consisting of twelve windows, each of which has a special purpose. While the setup module contains all the required documents and folders. For the rule-based approach applied to the signature, we need two essential files to run the program: The dictionary and the document to be classified. The dictionary file does not need to be static. It can be changeable based on the needs of the classification. Moreover, all required folders need to be created. We need a folder to save all created signatures. This folder enables us to call any signature back for further uses. For the ML-based approach, training set's files need to be set up before calling any classifier.

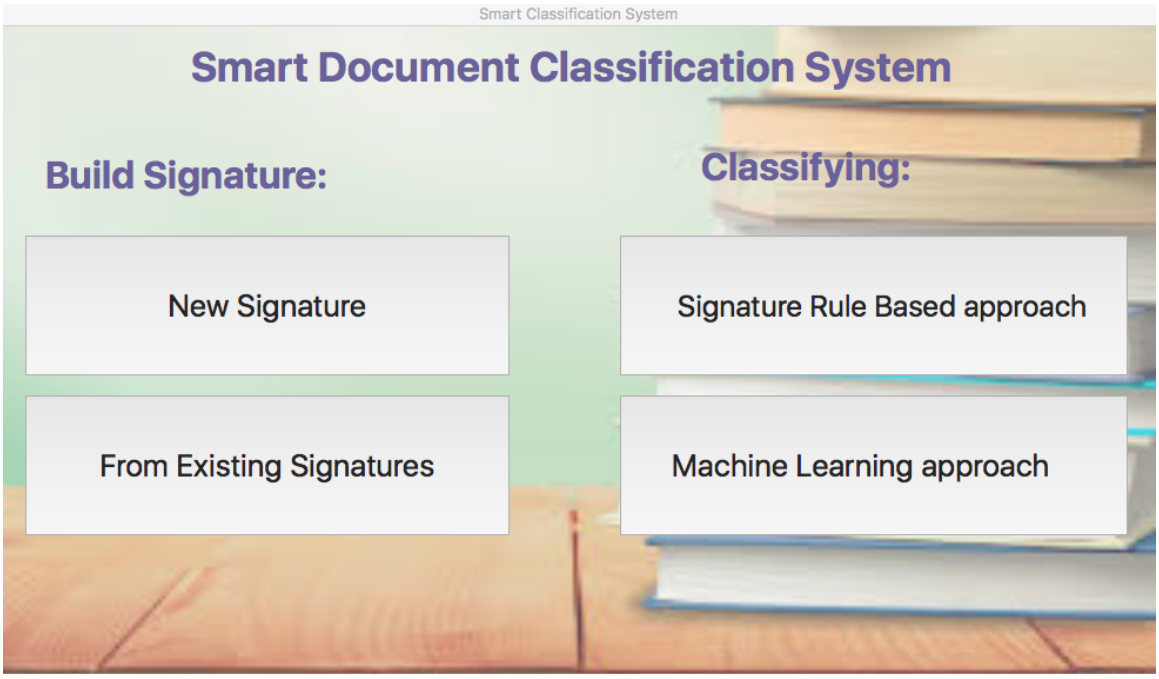

Figure 4.3: SDCS Interface

4.2.2 Classification Layer

The classification layer is the layer that provides the essential functionalities of our tool. In general, the classification layer consists of two main independent classification methods. The rule-based approach applied to the signature and ML-based classification approach. Rule-based approach applied to the signature is divided into two main modules: The signature builder module and the parsing rule module. Each of these consists of a number of functionalities. This approach was written in C++ language. Its implementation depends mainly on BDDs. To activate BDD on the C++ environment, the *BuDDy* library was used [LN10]. On the other hand, the ML-based classification approach consists of four classifiers modules: SVM, NB, K-NN, and DT. These modules were written in Python language. The user has the option to classify a document based on the rule-based approach applied to the signature or the ML-based approach. An abstract view of this layer is shown in Figure 4.4. A concrete view of the classification layer is shown in Figure 4.5. A detailed explanation of each approach along with their modules is given below.

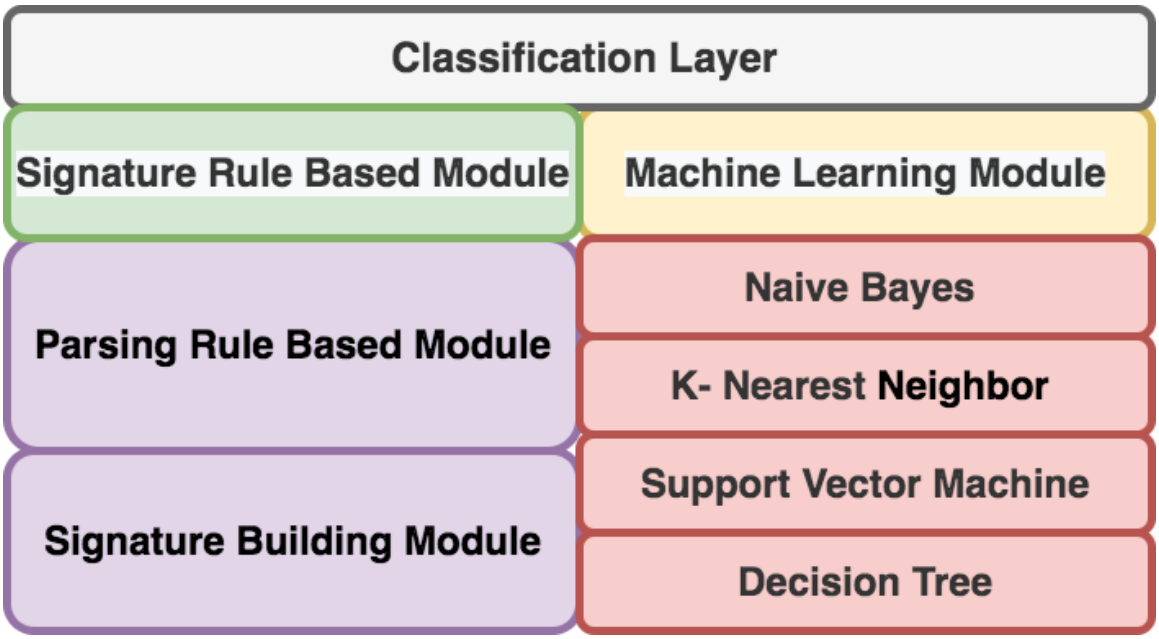

Figure 4.4: Abstract View of Classification Layer

4.2.2.1 Rule-based Approach Applied to the Signature

In this sub-section, I explain in detail how is the rule-based approach applied to the signature implemented. It consists of two connected modules. Signature building module and parsing rule module.

- Module: DocumentSignature.

  Service: This module contains the functions required to build a signature for the document to be classified.

- Module: ParsingRulesFunctions.

  Service: This module contains the rules that used for classifying the document's signature.

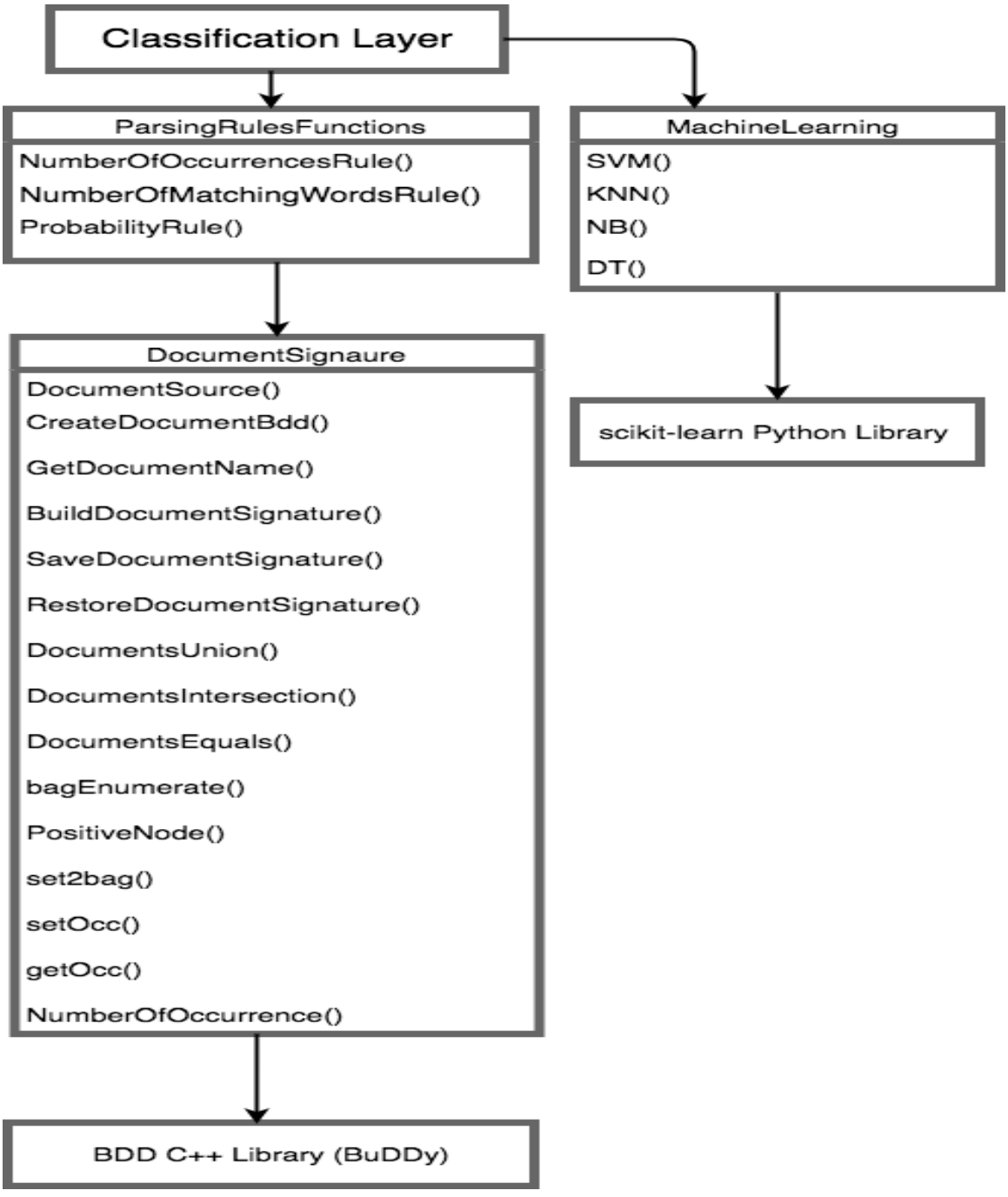

Figure 4.5: Classification Layer's Functions

4.2.2.1.1 Signature Building Module

A signature building module consists of a number of functionalities working together to get a unique signature for each document. This module depends mainly on BDD implemented as a bag model. To get access to BDD in the C++ environment, the *BuDDy* library was linked as a prior step [LN10]. The input of this module is dictionary and filtered document files. These files are taken from the basic utilities layer. The output is the document's signature. This signature is used later in the parsing rules module to get the final classification result.

- Access Program: DocumentSource.

  Service: This function opens the filtered document file in reading mode to get access to the document's words.

- Access Program: CreateDocumentBdd.

  Service: This function takes a copy of the BDD built for the dictionary file to be used as a document BDD. This BDD is used later to build the signature.

- Access Program: GetDocumentName.

  Service: This function extracts the document to be classified name from the path entered by the user. This name is used later to save the signature with the name of the document.

- Access Program: BuildDocumentSignature.

  Service: This function builds the document signature. Search for a word that appears both in document and dictionary files. Then convert the node value of this word in the document to be classified's BDD to 1.

- Access Program: NumberOfOccurrence.

  Service: This function counts the number of occurrences of a word in the document to be classified. Then, send it to *setOcc()* function.

- Access Program: setOcc.

  Service: This function encodes the number of occurrences of each word in the document to be classified as a binary string in the document's signature.

- Access Program: getOcc.

  Service: This function gets and shows the number of occurrences of a specific word from the BDD.

- Access Program: SaveDocumentSignature.

  Service: This function saves the built signature in the appropriate folder with a unique name. The name of the signature consists of the dictionary name, followed by the document to be classified name.

- Access Program: RestoreDocumentSignature.

  Service: This function calls the signature for specific document to be used again.

- Access Program: DocumentsUnion.

  Service: This function makes a logical *or* between two signatures to build a family of documents.

- Access Program: DocumentsIntersection.

  Service: This function makes a logical *and* between two signatures to get the commonalities between their documents.

- Access Program: DocumentsEquals.

  Service: This function checks if two documents are equal by comparing their signatures.

- Access Program: bagEnumerate.

  Service: This function prints out the document's signature to the program.

- Access Program: PositiveNode.

  Service: This function counts the number of nodes with value 1.

- Access Program: set2bag.

  Service: This function converts BDD from set implementation into a bag to allow encoding the number of occurrences of each word in the BDD.

4.2.2.1.2 Parsing Rules Module

This module is called after a signature is built. It takes the document's signature and applies some parsing rules on it to decide if a document belongs to a specific category or not. These rules are created using Bison parser and linked to an execution C++ file which contains the implementation functions of these rules. Choosing the appropriate rules to classify a document is one of the challenges of this tool. One needs to decide the right rules that produce a reasonable classification. We have created three rules to classify a document. These rules are considered general rules that always can be used for classification purposes. Later on, the user can decide which rules meet the classification goal, then add them to the classification process. The first rule is related to the number of occurrences of a word in the document to be classified. The second rule is about the number of matching words between a dictionary and the document

to be classified. The last rule calculates the probability of a word in the document to be classified. The probability of a word in a document is defined as the total frequency of a word in the document over the total number of words in that document [DAM05]. These rules can be used alone or combined by logical $\{and, or\}$. These rules can be entered by the user via file or entered through our program interface.

- Access Program: NumberOfOccurrencesRule.

  Service: This function is called if the rule that corresponds to the number of occurrences of a word in the document to be classified is entered by the user. This function classifies a document based on the number of occurrences of the entered word in the document to be classified signature. If the entered word appears in the document more than or equal to the given threshold, then the document is classified as belong. Otherwise, it does not belong. The grammar used to form the rule using the Backus-Naur Form (BNF) language format [MR03] is given in Table 4.1.

- Access Program: NumberOfMatchingWordsRule.

  Service: This function is called if the rule that corresponds to the number of matching words between the document to be classified and the dictionary is entered by the user. This function classifies a document based on the number of matching words between the document and the dictionary. If the number of matching words is greater than or equal to the given threshold, then the document is classified as belong. Otherwise, it does not belong. The grammar used to form the rule using the BNF language format [MR03] is given in Table 4.1.

- Access Program: ProbabilityRule.

  Service: This function is called if the rule that corresponds to the probability

percentage of a given word in the document to be classified is entered by the user. This function classifies a document based on the probability of an entered word in the document. The probability p of the word w in the document to be classified is calculated using the function: $p(w) = (\frac{total\_occurrence(w)}{\texttt{tw}}) \times 100$, where `tw` is the function that returns the total number of words in the document to be classified. If the probability of the word in the document is greater than or equal to the given threshold, then the document is classified as belong. Otherwise, it does not belong. The grammar used to form the rule using the BNF language format [MR03] is given in Table 4.1.

Grammar Format
<program> ::= <rule> | <rule> <program>
| <program> ::= <rule> & <rule> <program>.
<rule> ::= <NumberOfOccurance> | <MatchingWords> | <Wordprobability>.
<NumberOfOccurance> ::= <DictionaryName>'_'<DocumentName>','<Term>','<Integer> <EOL>.
<MatchingWords> ::= <DictionaryName>'_'<DocumentName>','<Integer> <EOL>.
<Wordprobability> ::= <DictionaryName>'_'<DocumentName>',' <Term>','<Integer>'%' <EOL>.
<DictionaryName> ::= <Term> | <DictionaryName> <Term>
<DocumentName> ::= <Term> | <DocumentName> <Term>
<Term> ::= <Letter> | <Term> <Letter>.
<Integer> ::= <Digit> | <Integer> <Digit>.
<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.
<Letter> ::= 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'.

Table 4.1: The Grammar syntax of the Parsing Rules Written in BNF Language.

4.2.2.2 Machine Learning Approach

ML-based approach is the second approach in this layer. It is an implementation of four ML classifiers: SVM, NB, K-NN, and DT. It was written in Python language and then linked to the user interface layer to be accessed by the program. *Scikit-learn* library enables us to implement these classifiers [PVG+11]. As these classifiers considered as supervised learning methods, the need for a training set is essential. British Broadcasting Corporation (BBC) articles were used as a training set with

a total of 2250 articles. These articles are categorized into five different categories: Technology, business, sport, entertainment, and politics. All training sets articles are converted into a format readable by ML classifiers through the following steps. First, remove unwanted words and stemming. Second, weight each word using conventional TF-IDF weighting scheme. Last, present these words with their weight as VSM. On the other hand, any document to be classified goes through the same process before calling any classifier.

- Module: ML.

  Service: This module gives access to ML classifiers.

- Access Program: SVM.

  Service: This module classifies a document using the SVM classifier.

- Access Program: NB.

  Service: This module classifies a document using the NB classifier.

- Access Program: K-NN.

  Service: This module classifies a document using a K-NN classifier.

- Access Program: DT.

  Service: This module classifies a document using a DT classifier.

4.2.3 Basic Utilities Layer

This layer consists of the basic functionalities that must be performed prior to the classification. These functionalities are divided into three modules: Document pre-processing module, BDD builder module, and BDD basic operations module. This

layer was written using C++ language. A concrete view of basic the utilities layer is shown in Figure 4.6

- Module: DocumentPreprocessing.

  Service: This module performs all the necessary filtering to the document to be classified prior to the classification.

- Module: BddOperations.

  Service: This module contains the functions required to set the value of BDD's nodes to 0 or 1. This module is used in the *Signature Building* module.

- Module: DictionaryBdd.

  Service: This module contains the functions required to build a BDD for the dictionary file.

4.2.3.1 Document preprocessing Module

Document preprocessing is the first module in this layer. This module is to reduce the size of the document to be classified by removing unwanted words and stemming.

- Access Program: RemoveLinkingWords.

  Service: This function removes all linking words from the document to be classified prior to classification. Linking words are saved in a file that is called each time this function start.

- Access Program: RemovePunctuationMarks

  Service: This function removes all punctuation marks from the document to be classified prior to classification.

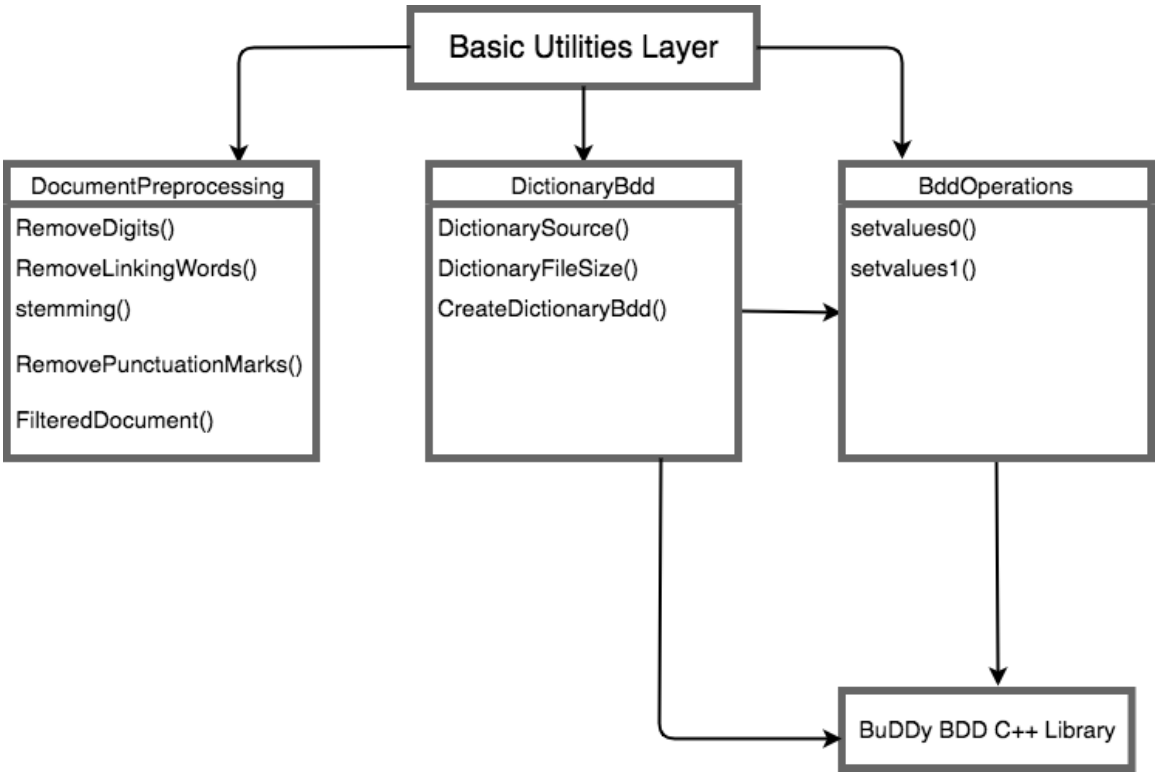

Figure 4.6: Concrete View of Basic Utilities Layer

- Access Program: RemoveDigits.

  Service: This function removes all digits from the document to be classified prior to classification. This includes Roman and English digits.

- Access Program: stemming.

  Service: This function gets the root of each word in the document to be classified prior to classification.

- Access Program: FilteredDocument.

  Service: This function saves the modified version of the document to be classified in a new file called Filtered Document. This file is used in the rest of the program

instead of the original one.

4.2.3.2 BDD Operations Module

In this module, we perform two essential functions on a BDD. We can call the function corresponding to convert the node value to 1, or vice versa. In other terms, this module helps us build the first BDD for the dictionary file, then create the signature for the document.

- Access Program: setvalues0.

  Service: This function converts the node's value inside BDD to 0.

- Access Program: setvalues1.

  Service: This function converts the node's value inside BDD to 1.

4.2.3.3 Dictionary BDD Module

This module holds the BDD corresponding to the dictionary file. This module gets access to the dictionary and counts the number of words in there. It creates a BDD equal to the size of the dictionary. This created BDD is considered the first step in the process of building the document's signature. As the dictionary is an essential component in the construction of our tool, an effort is needed to create the most complete dictionary that captures all the essential terms of a specific category. A complete dictionary is an important asset that affects the overall accuracy of the tool. The hardness of constructing a complete dictionary varies based on the category under consideration. While it is easy to create a dictionary for some categories, it becomes very hard in others. This is due to the limitations in resources to get a full list of the vocabularies that are related to the category.

- Access Program: DictionarySource.

  Service: This function opens the dictionary file in reading mode to get access to the dictionary's words.

- Access Program: DictionaryFileSize.

  Service: This function counts the number of words in the dictionary file.

- Access Program: CreateDictionaryBdd.

  Service: This function uses the size computed in *DictionaryFileSize()* function to build a BDD equal to the size of the dictionary. Each node in this created BDD presents a word in the dictionary. This function uses the *setvalues0* function from the *BddOperations* class to convert all node values to 0s.

4.3 Conclusion

In summary, our SDCS is consisting of three main layers: User interface, classification, and basic utilities layers. Each layer is divided into several modules. These modules contain several functionalities. At the end, these layers are connected to get our final tool. The layers are independent: modification in one layer does not affect the others. In general, the core of our system is located in the classification layer. Where our classification approaches are implemented. The user interface layer is responsible for providing interfaces to interact with the users. Also, it allows us to set up all necessary files and folders. However, the basic utilities layer is for preparing the document to be classified before moving to the classification stage and for building the basic BDD that will be used later to create the document signature.

Chapter 5

Tool Assessment and Usage

In this Chapter, I assess our tool as stand-alone software by assessing its capabilities in classifying documents using rule-based approach applied to the signature and using ML-based approaches. Then I discuss the results.

5.1 Introduction

SDCS includes two essential modules. The first module is about indexing and classifying documents using rule-based approach applied to the signature. In this module, the implementation depends mainly on BDD and its operations. The document could be categorized as belonging or does not belong to a specific category. The second module is about indexing and classifying documents using ML classifiers. Four main ML classifiers were used: SVM, NB, K-NN, and DT. Their training set consists of 2225 articles obtained from BBC. These articles are separated into five categories: Technology, business, sport, entertainment, and politics. Our assessment plan takes into consideration both modules. After that, we make a comparison between our

rule-based approach applied to the signature that is rooted in PFA and ML-based approaches that we are considering.

5.2 Assessment Plan

The assessment plan is separated into two sub-plans: Assessment of the rule-based approach applied to the signature and the ML-based approaches.

5.2.1 Assessment of the Rule-Based Approach Applied to the Signature Module

In this sub-section, I present the main objectives, the data sets that I used, and I explain the methodology of the assessment.

5.2.1.1 Assessment Objectives

The main objective of assessing the rule-based approach applied to the signature is to ensure its ability to correctly index and classify documents based on a given dictionary. We tested the ability of the module to index and classify 50 documents belong to five different categories: Technology, business, sport, entertainment, and politics.

5.2.1.2 Data Set Used

The data set used for assessing the module consists of 50 documents and five dictionaries. The five dictionaries are technology, business, sport, entertainment, and

politics dictionaries. Each dictionary contains 500 words. We use the same categories as in the training set of ML-based approaches. The categories along with their documents are shown in Tables 5.1.

Technology	**Business**	**Politics**	**Sport**	**Entertainment**
[DMGG16]	[com19]	[Emm20]	[Hau20]	[Gaw20]
[DSRA19]	[MT16]	[Min12]	[Bla]	[Kll20]
[ZZ19]	[FKS$^+$08]	[Rog15]	[KNK$^+$17]	[Pre20]
[SA18]	[MT17]	[Ort12]	[Uni20a]	[Zin20]
[Kim18]	[O'H04]	[Jer12]	[oC20]	[Con20]
[PS19]	[Fin19]	[SB10]	[CNR20]	[Gly20]
[Uni20b]	[Jef19]	[Sha16]	[Pas20]	[Mur20]
[Pat20]	[Che10]	[Par13]	[Osa20]	[McR20]
[WHW17]	[Hie13]	[Bar14]	[Bur20]	[Pai20]
[ZLW13]	[Roy19]	[Tro09]	[Rog20]	[Bue20]

Table 5.1: Data Set Used

5.2.1.3 Illustrative Examples for the Usage of the Module

Testing the rule-based approach applied to the signature is divided into two main steps. The first step is about building a signature for the document using BDD. The second step is about using one of the predefined parsing rules to classify the document based on its signature. The classification result changes based on the used dictionary and the used parsing rules. Examples 5.2.1, 5.2.2, 5.2.3, and 5.2.4 illustrate how we can classify a given document using this module.

In Example 5.2.1, I explain the first step to classify a document using this module, which is building a signature for a document.

Example 5.2.1. *Let us assume that we need to classify this document: "Fog Computing: A New Era of Cloud Computing" [DSRA19]. As we mentioned earlier, we start by building a signature for this document. For this purpose, we need to click on the New Signature button as shown in Figure 5.1. Building a signature requires us to enter a dictionary name along with the document to be classified name to build a signature related to the dictionary under consideration. To ensure the uniqueness of the signature, its name consists of two parts: The dictionary name followed by the document name. For instance, the signature name for the previous document built for the technology dictionary is "Technology_Fog Computing: A New Era of Cloud Computing Signature". When a signature constructed a pop-up message appears to confirm and specify the path of the signature as shown in Figure 5.2.*

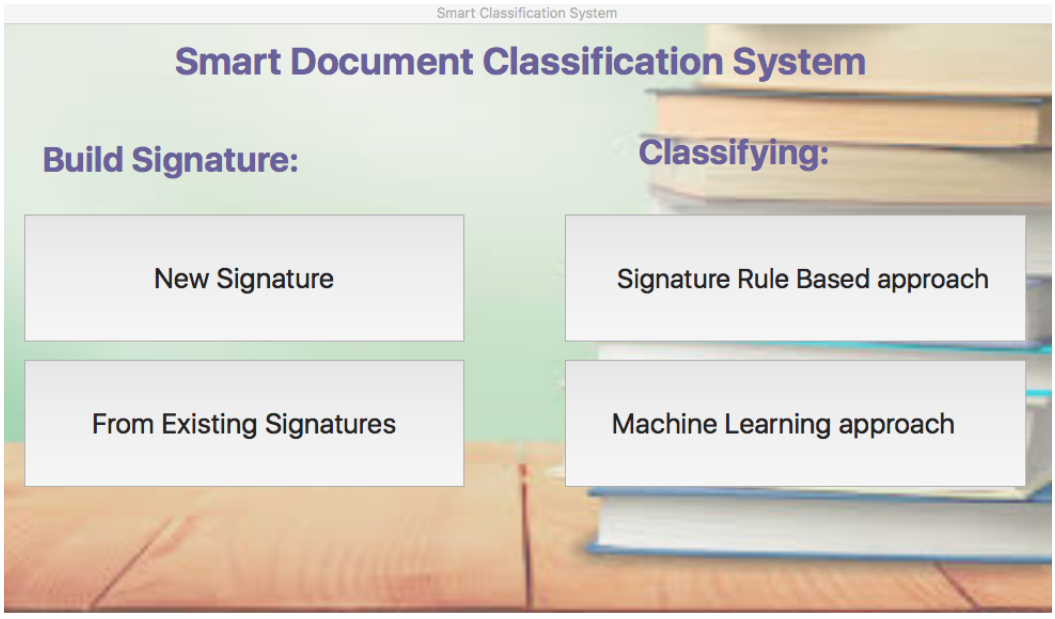

Figure 5.1: Main Interface of SDCS.

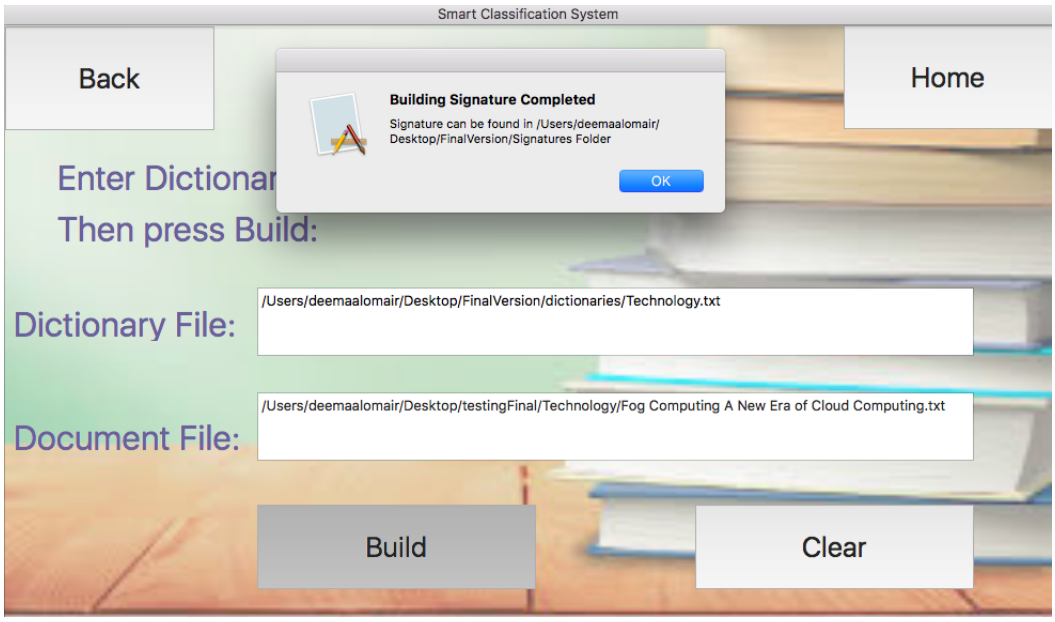

Figure 5.2: Building Signature completed.

In the following, I explain the second step of classification using rule-based approach applied to the signature module. We need to use one of the parsing rules to classify the document relatives to the created signature. Currently, we programmed three main parsing rules as explained in Chapter 4. To use the rule, we need to press the *Signature Rule Based Approach* button to start the classification as shown in Figure 5.3. Then, the system provides the option of getting the parsing rule from a file and the option of typing the parsing rule. Figure 5.4 illustrates the rule entry options screen. Figure 5.5 shows how we can enter a file path to classify a document.

In Example 5.2.2, I use the parsing rule that involves the number of occurrences of a word in the document to be classified. This rule examines the number of occurrences of a given word in the document to be classified by fetching the number of occurrences from the document's signature. If the number of occurrences is greater than or equal

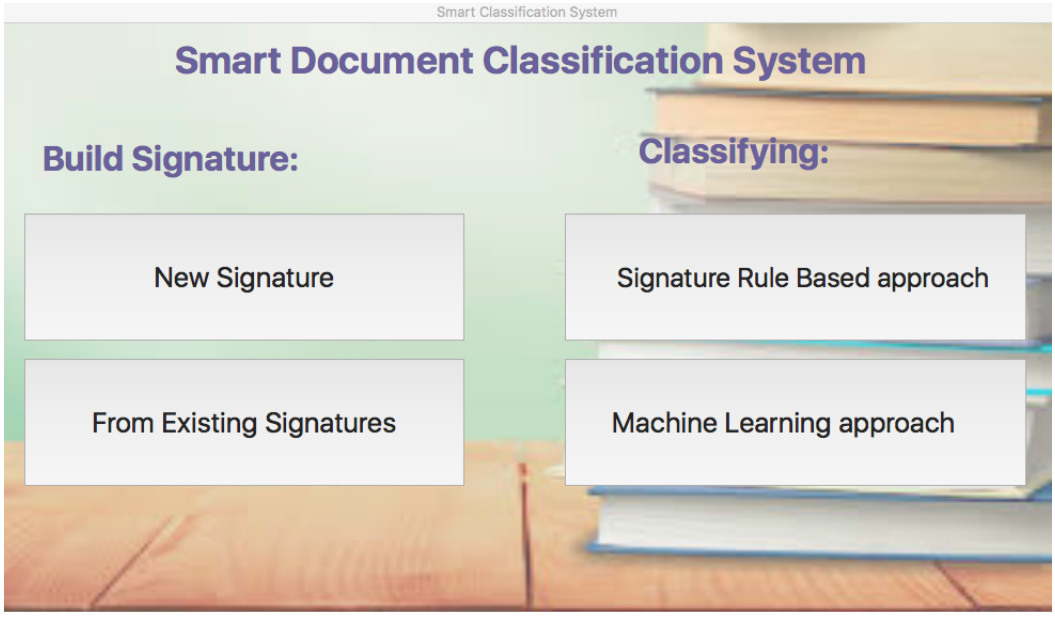

Figure 5.3: Main interface of SDCS.

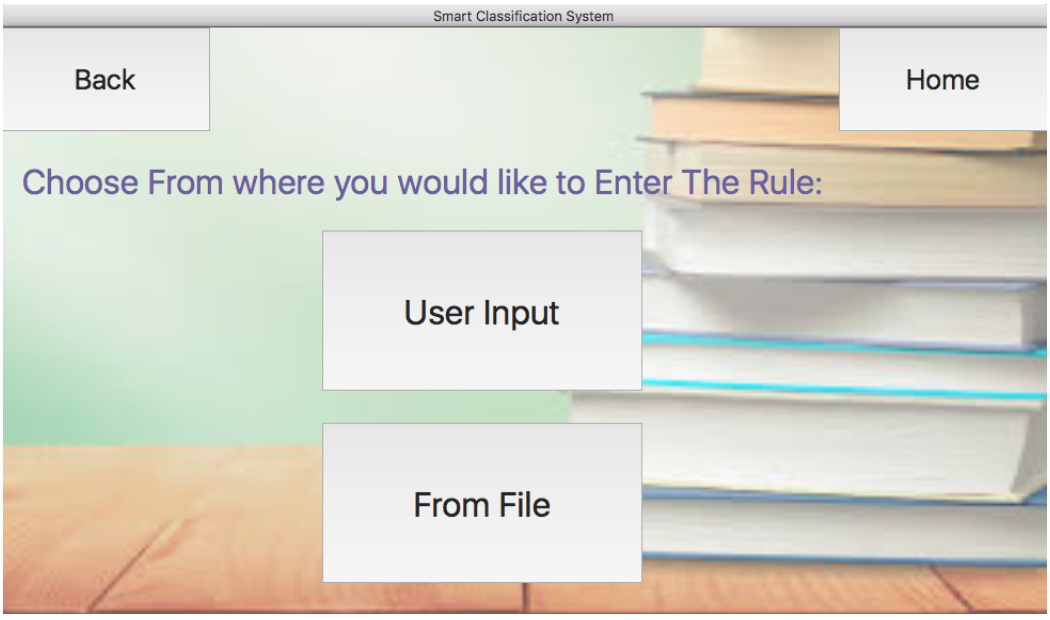

Figure 5.4: Rule-Based Approach Applied to the Signature Options Page.

to a given threshold, then the document belongs to the category of the dictionary. For more information about the grammar and the syntax of this rule, we refer the

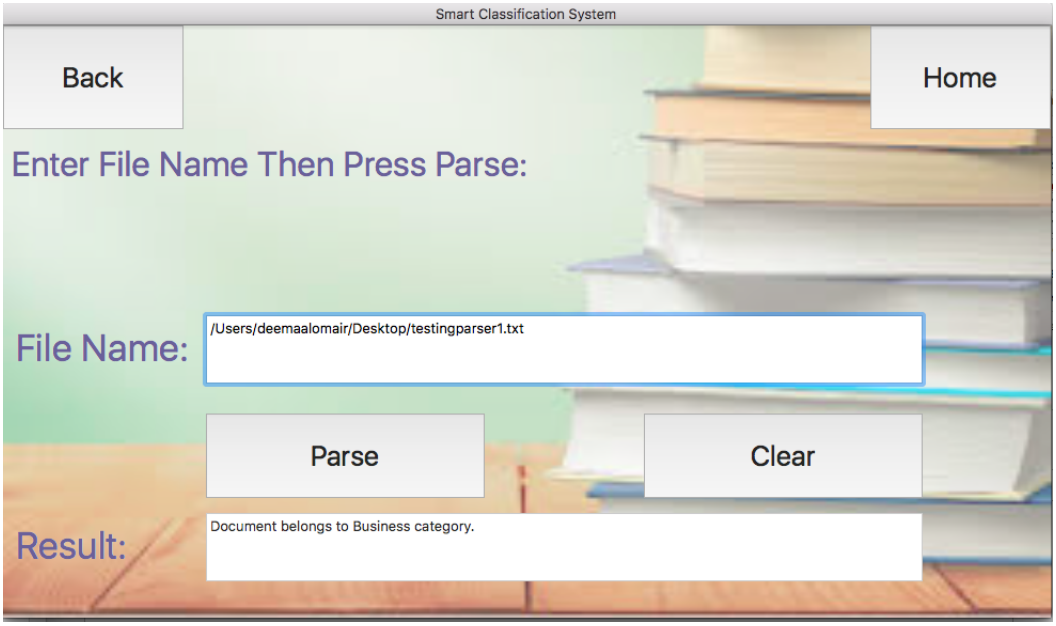

Figure 5.5: Classifying From External File Page.

reader to Table 4.1.

Example 5.2.2. *Let us assume that we require that the word "data" appears more than 20 times in the document "Fog Computing: A New Era of Cloud Computing" [DSRA19] to classify the document as belonging to the technology category. To test the previous condition, we need to use the parsing rule that corresponds to the number of occurrences of a word in the document to be classified. In our case, the rule is written "Technology_Fog Computing: A New Era of Cloud Computing,data,20". The document is categorized as a technology document as shown in Figure 5.6.*

In Example 5.2.3, I explore the parsing rule that counts the number of matching words between the document to be classified and the dictionary. If the matching number is greater than or equal to a given threshold, then the document belongs to the category of the dictionary. For more information about the grammar and the

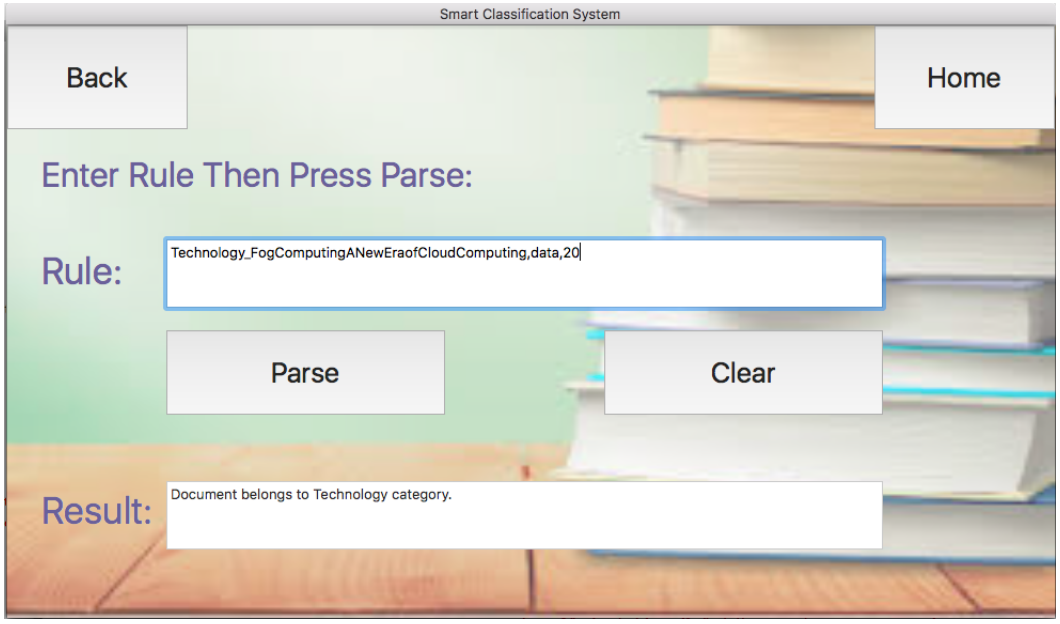

Figure 5.6: Using Number of Occurrence Rule.

syntax of this rule, we refer the reader to Table 4.1.

Example 5.2.3. *Let us assume that we would like to have 50 matching words between the previous document and the technology dictionary. In this case, the rule is written "Technology_Fog Computing: A New Era of Cloud Computing,50". The document belongs to the technology category as shown in Figure 5.7. However, when we increase the threshold to 100, it is no more considered as a technology document as illustrated in Figure 5.8.*

In Example 5.2.4, I explore the parsing rule that corresponds to the probability of a given word in the document to be classified. This parsing rule calculates the probability of a given word in the document to be classified. For more information on how probability is calculated, we refer the reader to Section 4.2.2.1.2. If the probability of the word is greater than or equal to a given threshold, then the document belongs

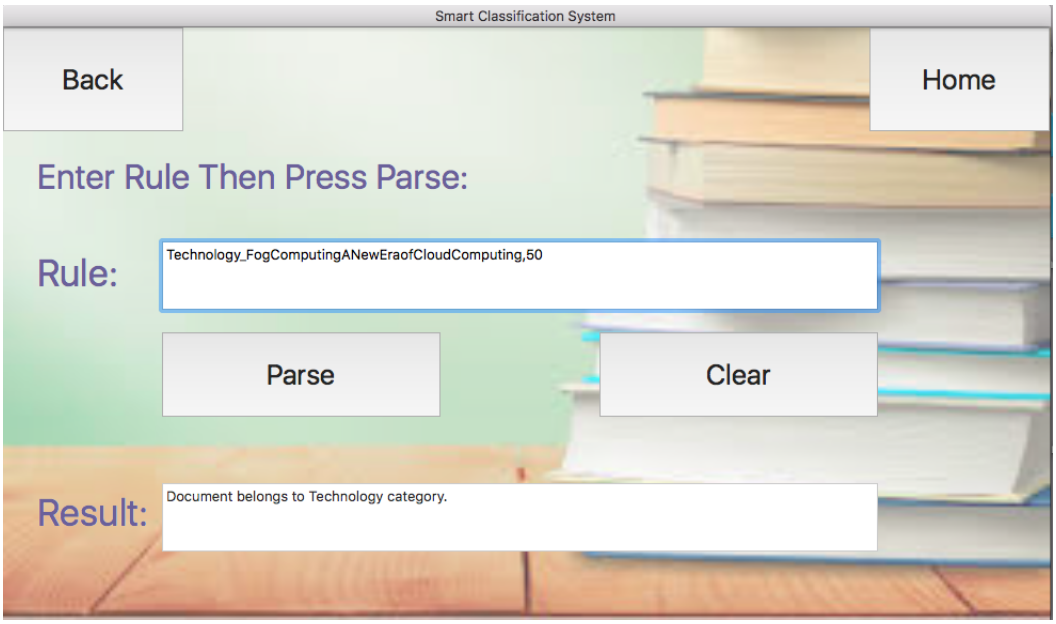

Figure 5.7: Using Matching Rule-First Case.

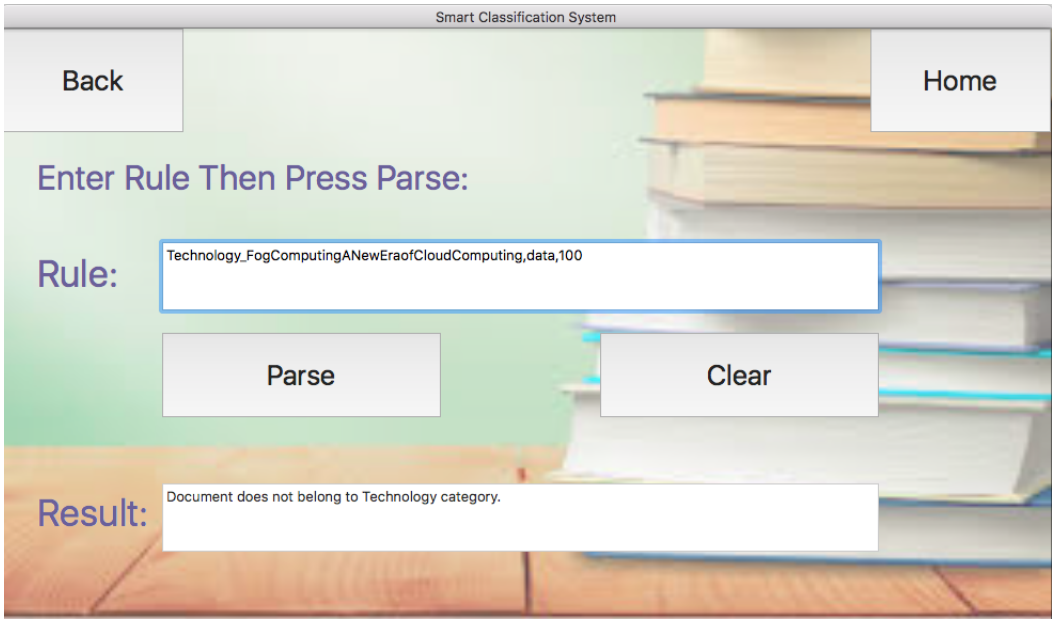

Figure 5.8: Using Matching Rule-Second Case.

to the category of a given dictionary. For more information about the grammar and the syntax of this rule, we refer the reader to Table 4.1.

Example 5.2.4. *Let us assume that we set the probability threshold of the word "data" to 2%. In this case, the rule is written "Technology_Fog Computing: A New Era of Cloud Computing,data,2%". Figure 5.9 represents that this document does not belong to the technology category.*

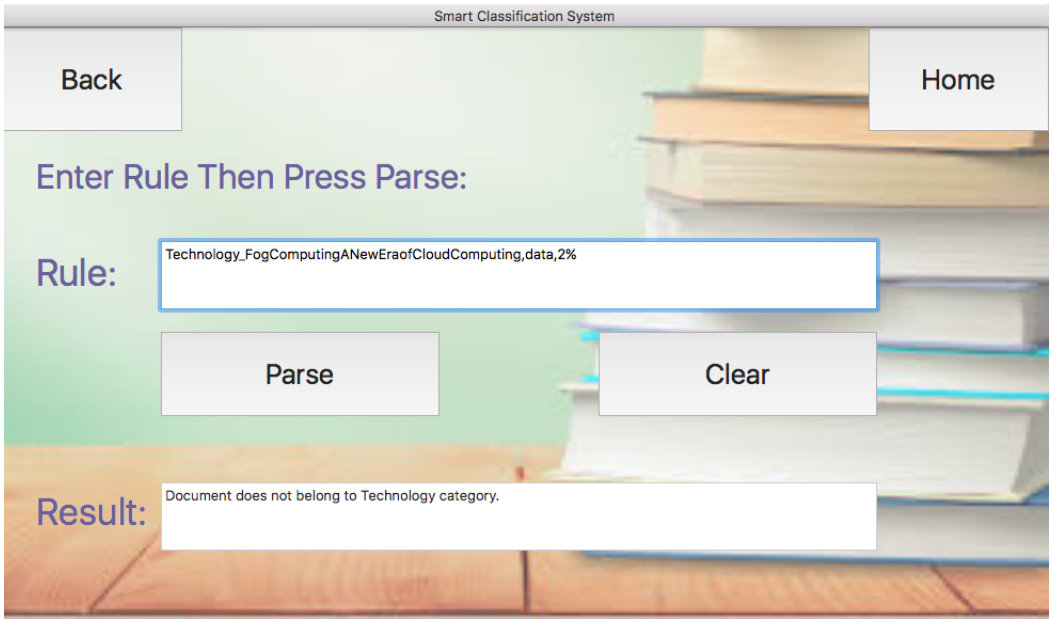

Figure 5.9: Using Probability Parsing Rule.

Assessing the ability of the module to get the commonalities or the accumulations of two documents is straightforward. Examples 5.2.5 and 5.2.6 give more details about these cases.

Example 5.2.5. *Let us assume that we need to build a family of two documents for the sport category. The first document is "How to break new records in the 200 meters" [CNR20]. The second document is "Does Heads Up Football Make the Sport Safer for Kids?" [Bla]. We start by building a signature for each document using the sport dictionary. Then, we click on From Existing Signatures button in the main*

interface window. After that, we choose Building a Family of Documents option as shown in Figure 5.10. Then, we enter the two signatures paths and press Build. When the signature is created, a pop-up message appears to give us the path of the new signature as shown in Figure 5.11. Then, we can use any of the parsing rules to classify the created signature.

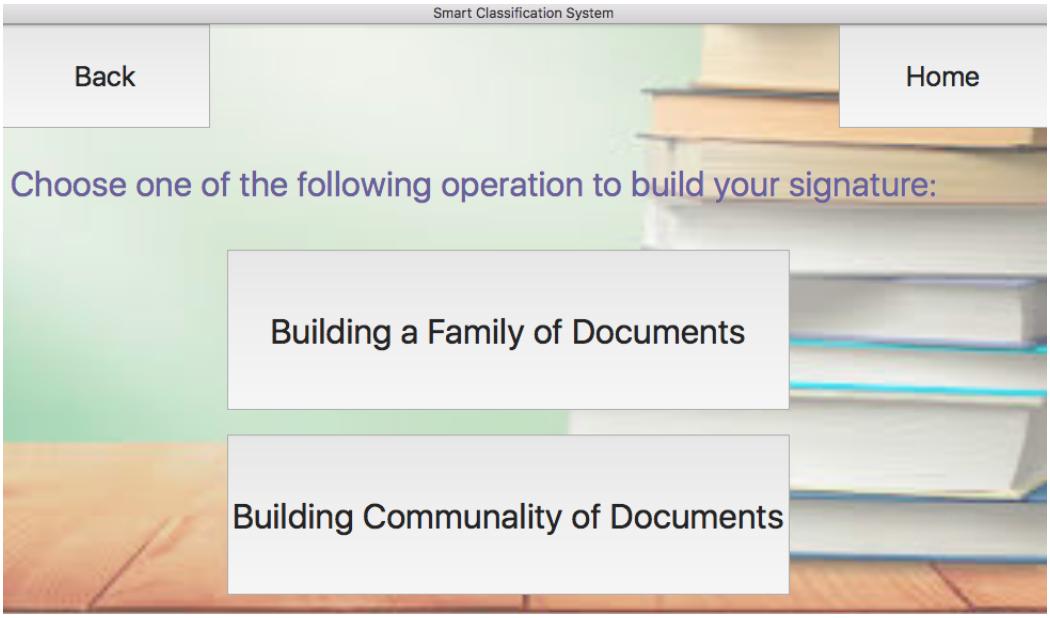

Figure 5.10: Options of Operations On Signatures

Example 5.2.6. *Let us assume that we need to get the common words between the previous documents. To get the commonalities, we need to choose Building Commonality of Documents as shown in Figure 5.10. Then, enter the paths of the signatures. When the signature is created, a pop-up message appears to give us the path of the new signature as shown in Figure 5.12.*

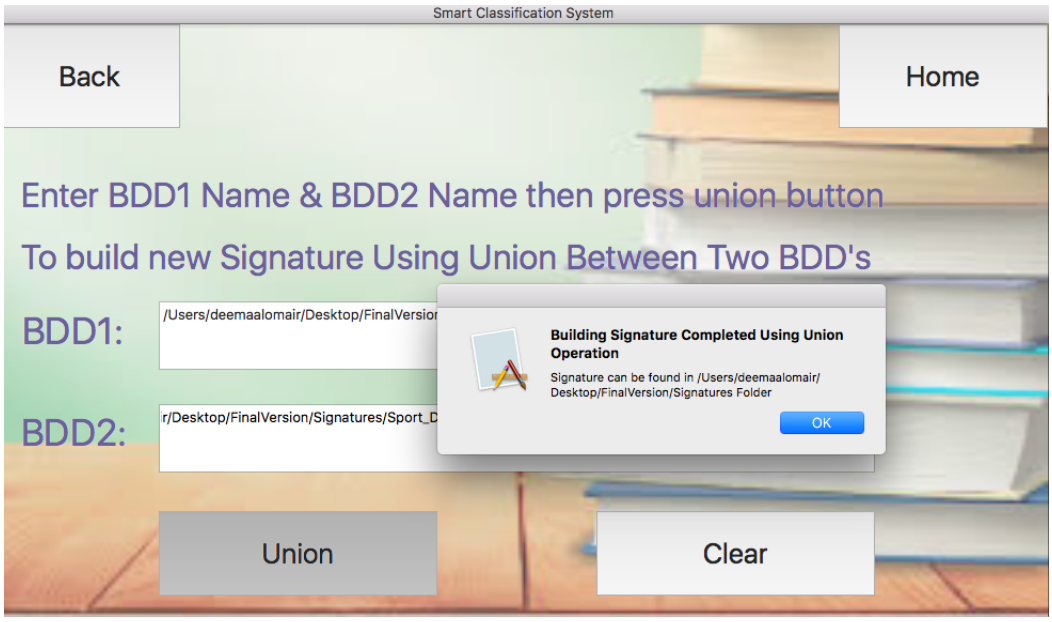

Figure 5.11: Family of Documents Created.

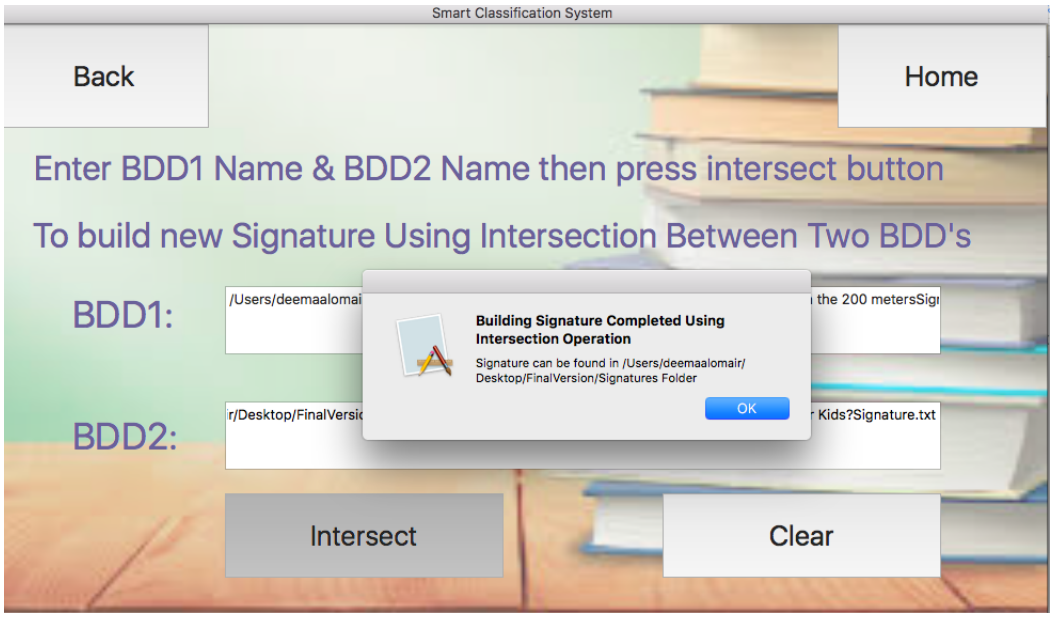

Figure 5.12: Commonality of Documents Created.

In Examples 5.2.1, 5.2.2, 5.2.3, 5.2.4, 5.2.5, and 5.2.6, I gave comprehensive cases

to explore the rule-based approach applied to the signature. In the next section, I give more details about the ML-based classification module. Then, explore the main differences between the rule-based approach applied to the signature module and ML-based module.

5.2.2 Assessment of ML-Based Module

In this sub-section, I explain how we assess the ML classification approach. I present the main objectives, the data sets that I used and I explain the methodology of the assessment.

5.2.2.1 Assessment Objectives

The main objective of assessing this module is to compare its result with the previous one. We tested 50 documents; every 10 documents belong to one of the predefined training set's categories. We applied the testing to four different ML classifiers: SVM, NB, K-NN, and DT.

5.2.2.2 Data Sets Used

I use the same testing data sets of the first module, presented in Section 5.2.1.2.

5.2.2.3 Illustrative Examples for the Usage of the Module

To test a document using one of the given ML classifiers, we need to press the *Machine Learning Approach* button in the main interface. Then choose one of the options as shown in Figure 5.13. Examples 5.2.7, 5.2.8, 5.2.9, and 5.2.10 give details on how to

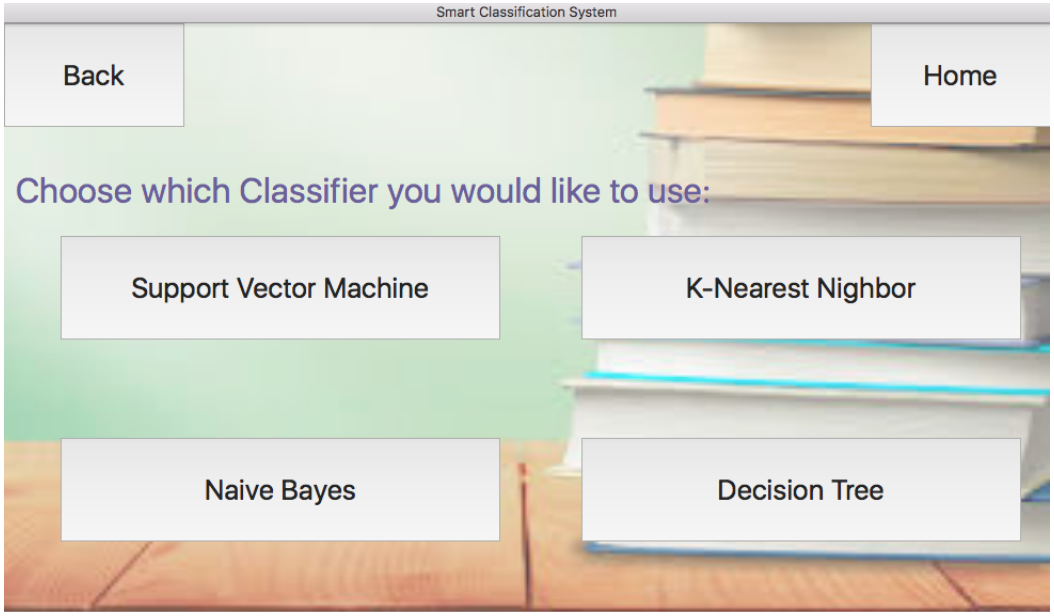

Figure 5.13: Machine Learning Options

classify a document using one of the four ML-based options. Each ML classifier is explained in a single example.

Example 5.2.7. *This example is to illustrate the usage of the SVM classifier. To classify a document using SVM, we need to enter the path of the document and click classify. In our case, we aim to classify the same document "Fog Computing: A New Era of Cloud Computing" [DSRA19]. SVM classifies the document into the technology category which is considered as the right classification. Figure 5.14 gives the output of this example.*

Example 5.2.8. *This example is to illustrate the usage of the NB classifier. To classify a document using NB, we need to enter the path of the document and click classify. In our case, we aim to classify the same document "Fog Computing: A New Era of Cloud Computing" [DSRA19]. NB classifies the document into the technology category which is considered as the right classification. Figure 5.15 gives the output*

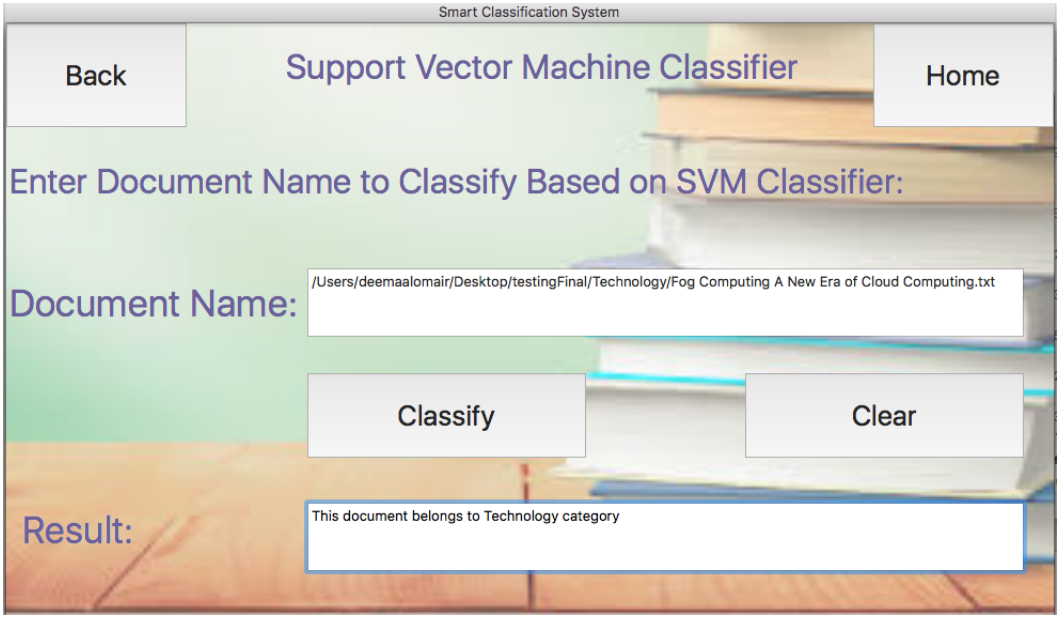

Figure 5.14: Support Vector Machine Classifier.

of this example.

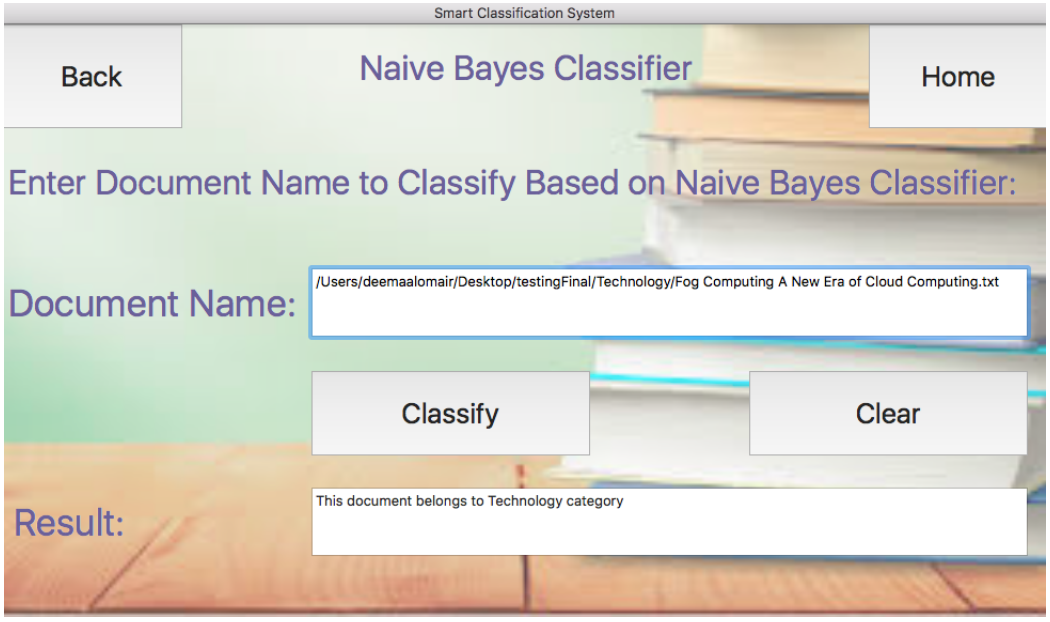

Figure 5.15: Naive Bayes Classifier.

Example 5.2.9. *This example is to illustrate the usage of the K-NN classifier. To classify a document using K-NN, we need to enter the path of the document and click classify. In our case, we aim to classify the same document "Fog Computing: A New Era of Cloud Computing" [DSRA19]. K-NN classifies it into the technology category which is considered as the right classification. Figure 5.16 gives the output of this example.*

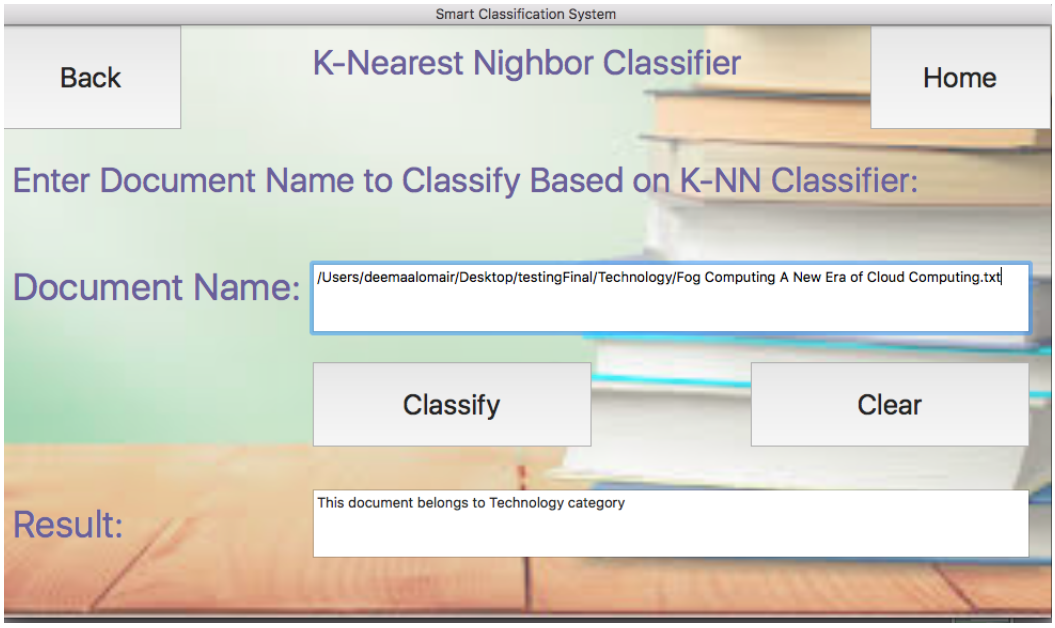

Figure 5.16: K-Nearest Neighbor Classifier.

Example 5.2.10. *This example is to illustrate the usage of the DT classifier. To classify a document using DT, we need to enter the path of the document and click classify. In our case, we aim to classify the same document "Fog Computing: A New Era of Cloud Computing" [DSRA19]. DT classifies the document into the technology category which is considered as the right classification. Figure 5.17 gives the output of this example.*

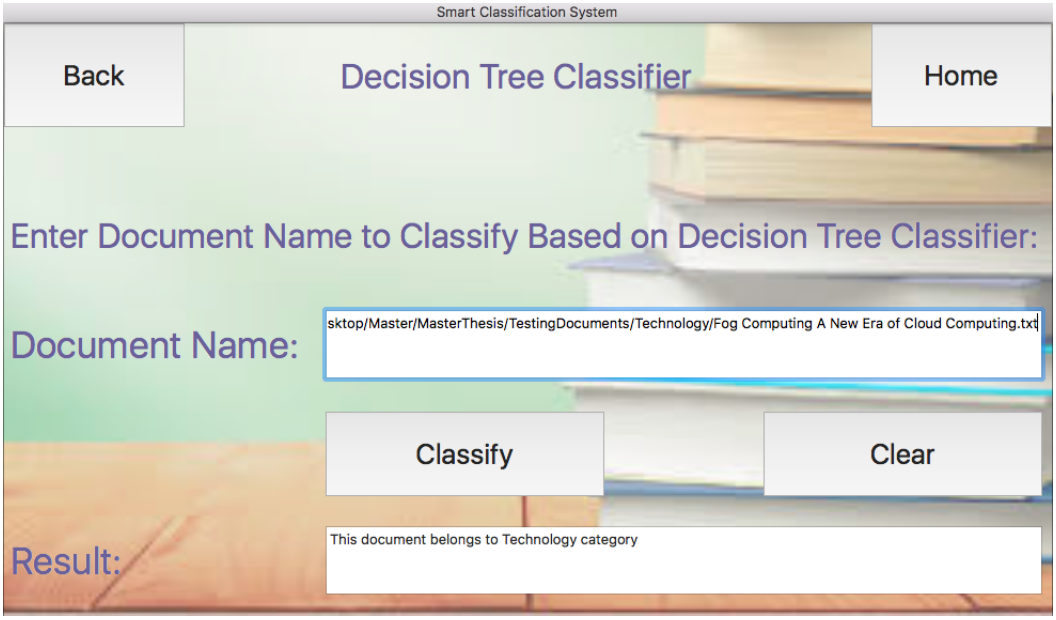

Figure 5.17: Decision Tree Classifier.

5.3 Assessment Results

In this section, I show and discuss the results for each module.

5.3.1 Results of the Assessment of the Rule-Based Approach Applied to the Signature

Measurement of the accuracy in this module is checking if the system is capable of building the correct signature and of giving the right classification result based on the chosen rules. All testing applied using different rules gives the expected correct classification. The result of the rule-based approach applied to the signature module depends mainly on the chosen rules to classify the document. In our testing phase, we use one rule or a combination of rules for each category. We try to diversify between

the rules to highlight the effect of the rules on the classification result. When we apply the rules, we try to use a small threshold value because our testing documents are not too long. Parsing rules along with their associated categories are given in Table 5.2. The result is shown in Figure 5.18.

Category	Rule Used	Rule Syntax
Technology	50 matching words between dictionary and document (Rule number two).	Dictionary name_Document name, 50.
Business	Number of occurrences of word *organizations* ≥ 10, or the number of occurrences of word *business* ≥ 10 (Rule number one).	Dictionary name_Document name,organizations,10+ Dictionary name_Document name,business,10.
Politics	10 matching words between dictionary and document (Rule number two) and the number of occurrences of word *politics* ≥ 10 (Rule number one).	Dictionary name_Document name,10& Dictionary name_Document name,politics,10
Sport	20 matching words between dictionary and document (Rule number two), or probability of the word *sport* in the document $\geq 2\%$ (Rule number three) and the number of occurrences of word *sport* ≥ 6 (Rule number one).	Dictionary name_Document name,20+ Dictionary name_Document name,sport,2%& Dictionary name_Document name,sport,6.
Entertainment	9 matching words between dictionary and document (Rule number two)	Dictionary name_Document name,9.

Table 5.2: Parsing Rules used for the Assessment Phase

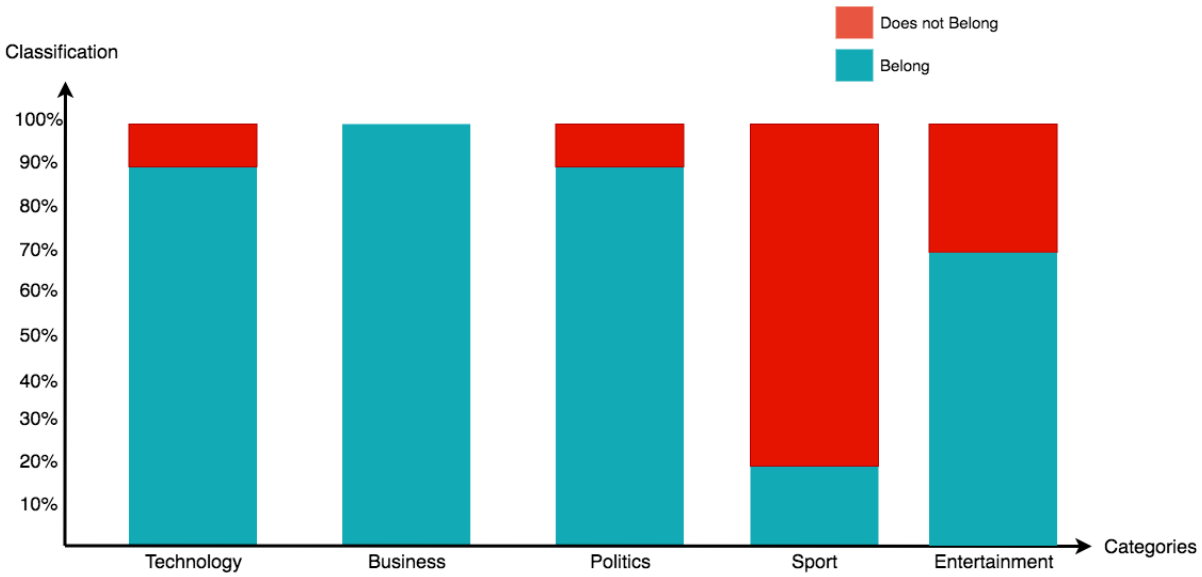

Figure 5.18: Result of Rule-based Approach Applied to the Signature Module

It is noticeable from Figure 5.18 that if we use more combinations of rules or increase the threshold value, the number of classified documents as belong to the category

decreased as in the *Sport* category. Moreover, there is no right or wrong classification. User can use the parsing rule that meets the purpose. Flexibility is the main advantage one can get from this approach. Adding the parsing rule that matches the demand is the only effort needed to classify a document. In addition, we can attempt to classify the same document with an unlimited number of categories by creating a signature for that category then classify.

5.3.2 Results of the Assessment of ML-Based Module

ML accuracy results could be divided into four main Charts. Each of which corresponds to one ML classifier. Figure 5.19, 5.20, 5.21, and 5.22 show the accuracy results of K-NN, NB, SVM, and DT, respectively. Figure 5.23 shows the average accuracy for each ML classifier.

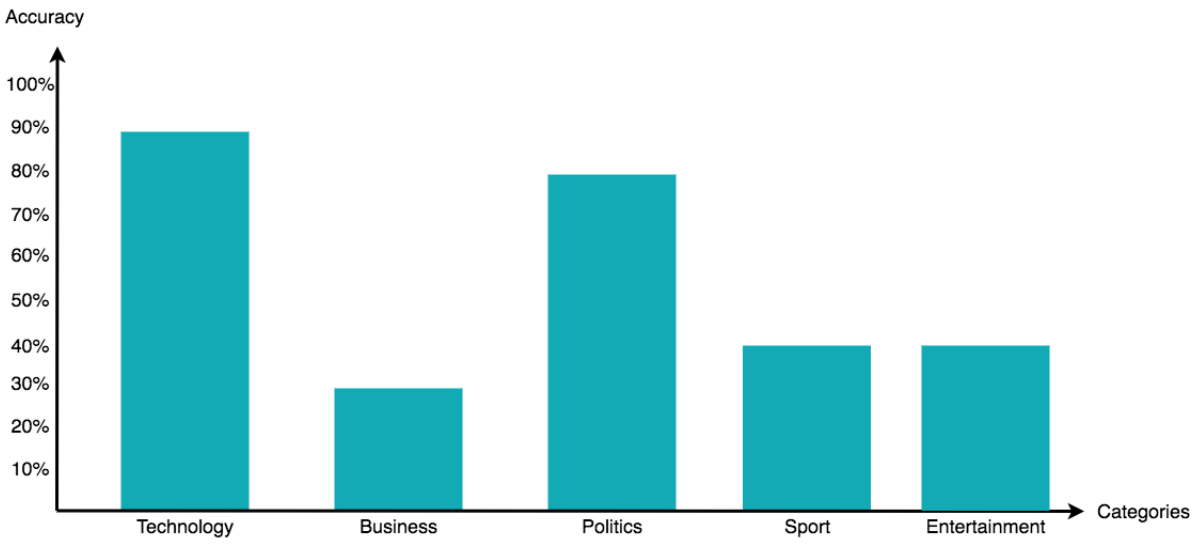

Figure 5.19: Accuracy Result for K-Nearest Neighbor.

It is clearly shown that SVM has the highest accuracy with an average of 84%. It

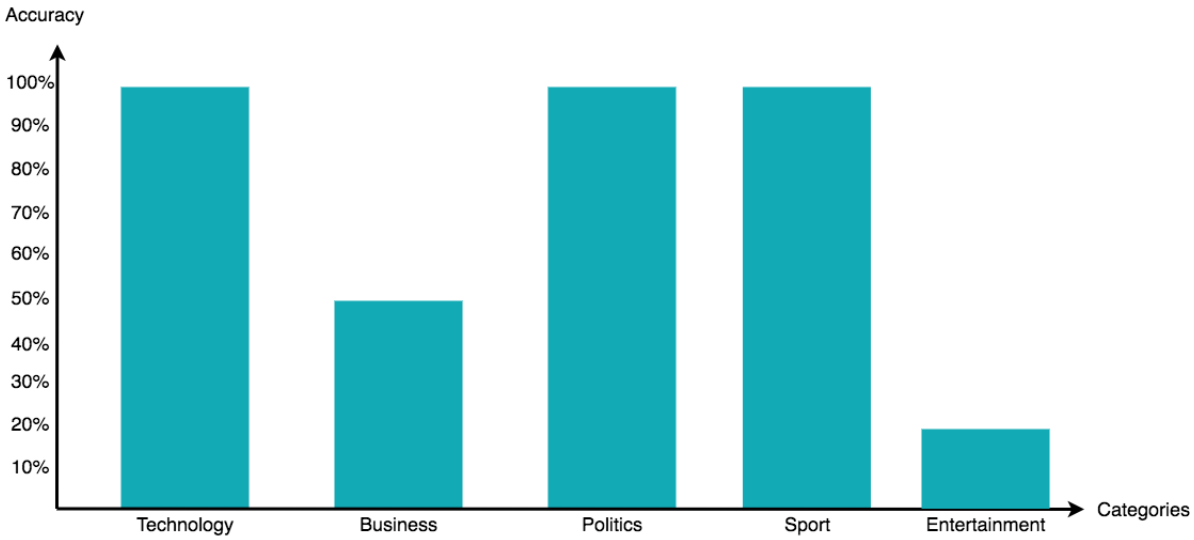

Figure 5.20: Accuracy Result for Naive Bayes.

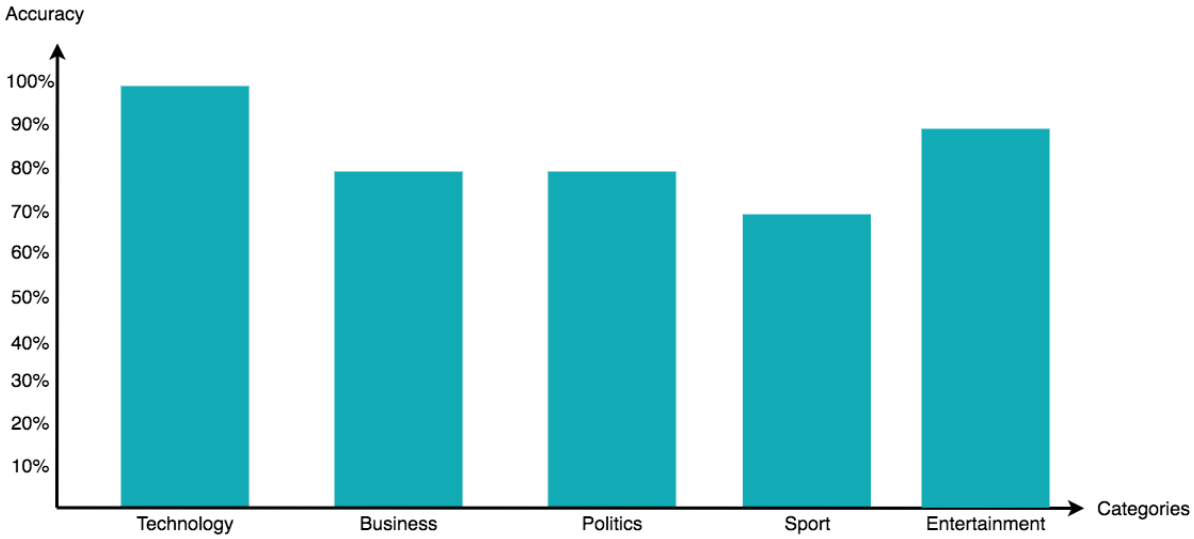

Figure 5.21: Accuracy Result for Support Vector Machine.

always showed good classification results for all categories, where the lowest accuracy of SVM was for the *Sport* category with 70%. Next in rank is the NB classifier with an average of 74%. NB gave good classification results for all categories except for the

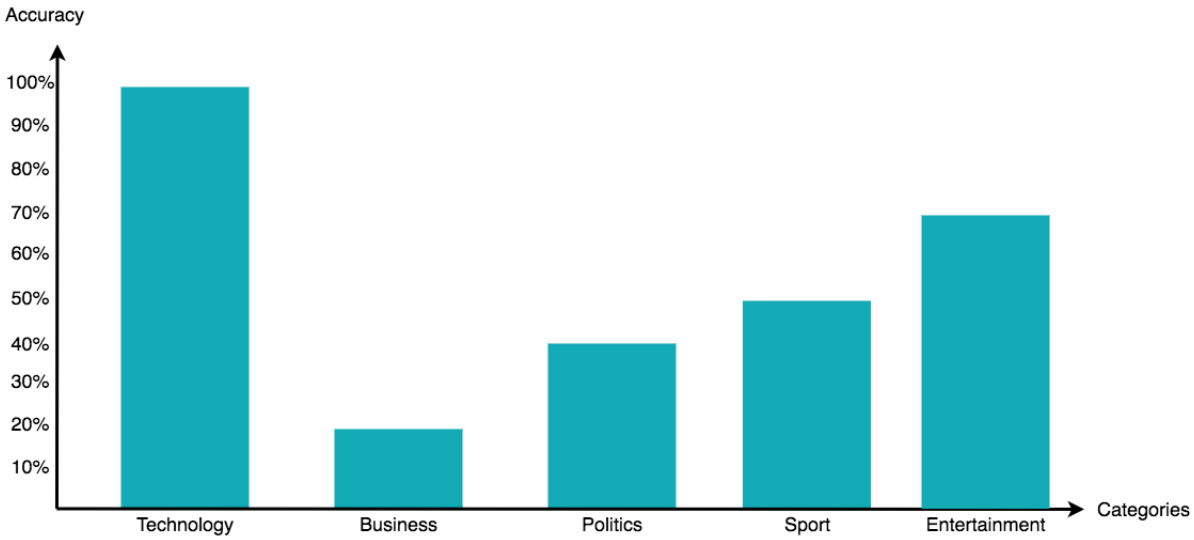

Figure 5.22: Accuracy Result for Decision Tree.

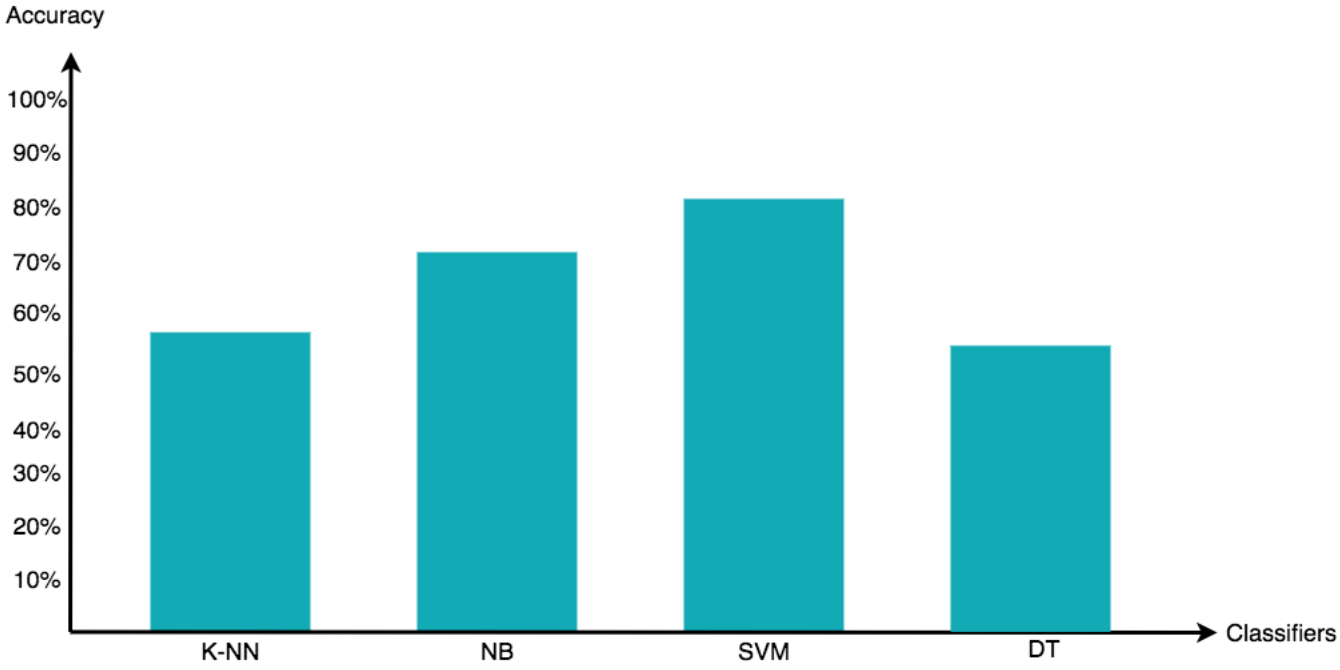

Figure 5.23: Average Accuracy Result For Machine Learning classifiers.

Entertainment category. It was not able to figure out this category correctly. Then K-NN classifier with average of 58%. The least accurate ML classifier is DT with an average of 56%. From a different angle, it was noticeable that all the classifiers could classify the *Technology* category very easily and recorded the highest accuracy for it.

5.4 Discussion

The assessment of the rule-based approach applied to the signature module and ML-based module brings out some important points that I should emphasize.

In a rule-based approach applied to the signature module, there is no right or wrong classification. The user decides which rule fits the purpose. When a rule is chosen, the result is given according to the rule. The main gain of this module is the flexibility we get in meeting user demand by adding parsing rules to perform new classification. However, the classification result is affected by the used parsing rule. Parsing rule is the determining factor that decides if a document belongs to a certain category or not. As a consequence, when we increase the number of restrictions used to classify documents, we reduce the number of documents that belong to a certain category. The reason is that the inclusion criteria for a category get stricter. This module is category oriented. The user can test the same document for different categories by creating a signature for each category. The use of the dictionary in this module could be treated as a replacement of the training-set in ML-based module. One of the main strengths of the rule-based approach applied to the signature module is its ability to build a family of documents, then classifies them at once. Moreover, its capability to get the commonality between two documents, then classifies them at once.

In the ML-based module, almost all used classifiers gave satisfactory classification

results. SVM was the best classifier with the highest classification accuracy value. However, DT got the lowest performance accuracy. It was noticed that the performance of ML classifiers is affected by the category of the document to be classified. For instance, K-NN and DT could not recognize *Business* documents, while NB performance drooped down significantly in classifying the *Entertainment* category. On the other hand, all used ML classifiers perform very well on classifying documents that belong to the *Technology* category. In general, the performance fluctuated and unpredictable.

5.5 Conclusion

In this Chapter, we assessed and illustrated the usage of our tool as a stand-alone system. We focused on two independent modules: the rule-based approach applied to the signature module and the ML-based module. Despite the efforts needed to choose then create the right classification rules and dictionaries, we found that the first module is more flexible and can serve user demand by adding classification rules. In addition, it can use more than one classification rule at the same time with the help of the operations on signatures. Moreover, one can classify more than one document at once using commonality or accumulation features. On the other hand, ML classifiers had satisfactory results. However, the performance was not at the desirable level for some categories.

Chapter 6

Conclusion and Future Work

In this thesis, we explored the topic of indexing and classifying unstructured textual documents. We gave a comprehensive explanation of our framework. The goal of our thesis is to produce a deterministic, fast, and effective tool to index and classify individual textual document or families of documents. To assess the capabilities of our tool, we compared its performance to four ML classifiers. In this Chapter, I summarize the main contributions of our work. I present the main strengths and limitations of our tool. I conclude with the expected future work that we could continue with.

6.1 Contributions

In this section, I summarize the main contributions of our work. Our classification process starts with the construction of a signature for a document or a family of documents using the BDDs. Second, we use the created signature to guide the classification process. Third, we implement four ML-based classifiers: SVM, NB, K-NN,

and DT. Fourth, we build a software called SDCS that can classify a document or a family of documents using the rule-based approach applied to the signature or using ML-based classifiers. Last, we compare the performance of our rule-based module with the aforementioned ML-based classifiers and list the main outcomes.

6.2 Interpretation and Value of the Contributions

In this section, I explain the main advantages and limitations of the rule-based approach applied to the signatures. The main strengths of our approach are summarized as the following; the constructed signature is a multi-uses signature. In other terms, we can classify a document using the same signature with different classification criteria. In addition, the ability of the module to create a signature for a family of documents and classify it at once. Moreover, the flexibility of our approach to classify a document based on the user demands by adding a rule that meets the purpose. Also, many signatures can be constructed from a single document, where each signature serves a specific category. Despite the aforementioned strengths, there are some limitations in our approach that I need to point out. First, only the presence of the words that are taken into account. The context in which the word is present is not considered. This might leads that different documents might have the same signature. In addition, the proposed approach depends mainly on dictionaries. To be able to use our approach, a user needs to set up the dictionary file that serves a specific category . The difficulty of creating a dictionary depends on the category under consideration. For some general categories it should be easy, while it might be a challenging for others. Another limitation of this approach is that the user needs to go through two steps to finish the classification process. First, create the signature.

Then, use this signature to classify a document. At last, coming up with the right classification rules to classify a given document should be done with caution. A user needs to decide which rules could be used to give the best classification results and meet the purpose of the classification.

6.3 Future Work

Electronic web pages are one of the main resources of data. With the increasing number of this source, we need to ensure access to secure content. One way to distinguish malicious web pages from secure ones is to classify the web page. Our next goal is to explore the classification of web pages using the rule-based approach applied to the signature approach. The aim is to categorize web pages into secure or malicious. Then, classify the secure ones into their right categories. Another direction of our future work is to apply the rule-based approach applied to the signature approach in the documents archiving field. Our tool could be used in archiving documents into their appropriate storage based on their category.

Glossary

class A class or a category is a set of documents that share the same topic and have common terms. Each class might have a different size depending on the number of documents it contains. 3

commutative Mathematical property where exchange the position of the elements in the operation will not change the result.. 27, 28

Cutter Number Is the third part of the recall number that indicates the author name based on Congress Classification System. 1

idempotent Mathematical property where the element is not changing when operated with itself.. 27, 28

Lazy Learner Is type of machine learning algorithm in which the generalization of the training data is performed when the classification phase starts to get the most recent training data.. 4

Machine Learning Is the science that involves mathematical models and training data to help its approaches in making decisions and predictions without human interaction. It is a subset of the Artificial Intelligence field.. iii

Partial Matching or Recording Is a method of editing the word after removing its suffix to turn into a valid word. 11

Recall Number The number that is labeled on the library books to be used in many classification systems. 1, 2

Sentence Ranking Is a method of indexing the document based on the number of occurrences of a word and the location of the word in the document. 5

weighting schemes Giving each word a special weight of importance based on some criteria. 12

Bibliography

[Alt10] F. Alturki. *Jory: A Tool for Feature Modelling Based on Product Families Algebra and BDDs*. PhD thesis, McMaster University, 2010.

[BA13] F. Boubekeur and W. Azzoug. Concept based indexing in text information retrieval. *International Journal of Computer Science and Information Technology*, 5(1), February 2013.

[Bar14] B. Barber. The politics of development. *Human Development*, 57(6):319–321, 2014.

[BEMS17] M. Bounabi, K. El Moutaouakil, and K. Satori. A comparison of text classification methods method of weighted terms selected by different stemming techniques. In *Proceedings of the 2Nd International Conference on Big Data, Cloud and Applications*, BDCA'17, pages 43:1–43:9, New York, NY, USA, 2017. ACM.

[BG12] H. Bhavsar and A. Ganatra. A comparative study of training algorithms for supervised machine learning. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(4):2231–2307, 2012.

[BK19] B. Behera and G. Kumaravelan. Towards the deployment of machine

learning solutions for document classification. *International Journal of Computer Sciences and Engineering*, 7:193–201, 03 2019.

[Bla] M. Bland. Does heads up football make the sport safer for kids? https://www.activekids.com/football/articles/does-heads-up-football-make-the-sport-safer-for-kids.

[BO15] K. Boukhari and M.N. Omri. Said: A new stemmer algorithm to indexing unstructured document. In *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 59–63, Decemeber 2015.

[Bue20] A. Bueno. Keanu reeves kept getting original 'john wick' movie title wrong, so they changed it. https://www.etonline.com, May 2020.

[Bur20] A. Burke. Paul lambert's borussia dortmund dream 23 years on. http://www.bbc.com/sport/football/52624318, May 2020.

[Che10] H. Chesbrough. Business model innovation: Opportunities and barriers. *Long Range Planning*, 43(2):354 – 363, 2010.

[CIW16] L.M. Chan, S.S. Intner, and J. Weihs. *Guide to the Library of Congress classification*. ABC-CLIO, 2016.

[CNR20] CNRS. How to break new records in the 200 meters. www.sciencedaily.com/releases/2020/03/200325120837.htm, March 2020.

[com19] 1800Accountant company. Which business entity is right for you? https://smallbizclub.

com/startup/making-your-business-official/
which-business-entity-is-right-for-you/, August 2019.

[Con20] C. Contreras. See how this mom and her 5-year-old daughter recreated iconic album covers. https://www.eonline.com/news/1149885/, May 2020.

[DA16] F.H. Dsouza and V.S. Ananthanarayana. Document classification with a weighted frequency pattern tree algorithm. In *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, pages 29–34, March 2016.

[DAM05] A. Doucet and H. Ahonen-Myka. A method to calculate probability and expected document frequency of discontinued word sequences. In *ELECTRA Workshop on Methodologies and Evaluation of Lexical Cohesion Techniques in Real-world Applications (Beyond Bag of Words)*, page 33, August 2005.

[DMGG16] A. De Mauro, M. Greco, and M. Grimaldi. A formal definition of big data based on its essential features. *Emerald Group Publishing Limited*, 65(3):122–135, 2016.

[DSRA19] S. Delfin, N. P. Sivasanker, N. Raj, and A. Anand. Fog computing: A new era of cloud computing. In *019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1106–1111, 2019.

[Emm20] M. Emmett. Public policy and constitutional rights in times of crisis. *Canadian Journal of Political Science*, pages 1–5, 2020.

[Fin19] Fora Financial. What to do if your business loan application was declined. `https://www.forafinancial.com/blog/working-capital/steps-can-take-business-loan-application-declined/`, June 2019.

[FKS+08] M. Filipowska, M. Kaczmarek, P. Starzecka, Stolarski, and A. Walczak. Semantic enterprise description for the needs of business process automation. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 987–992, 2008.

[Gaw20] P. Gawley. Dj pauly d and vinny guadagnino will help viral prank victims get revenge on new mtv show. `https://www.etonline.com`, May 2020.

[Gly20] P. Glynn. Uk film and tv: 'expect lots of dramas in space or under the sea'. `https://www.bbc.com/news/entertainment-arts-52663585`, May 2020.

[Gol15] J. Golbeck. *Introduction to Social Media Investigation*. Syngress, Boston, March 2015.

[Har18] M. Haroon. Comparative analysis of stemming algorithms for web text mining. *International Journal of Modern Education and Computer Science(IJMECS)*, 10(9):20–25, 2018.

[Hau20] J. Haughey. Derek graham: Belfast running great has irish records restored after 53 years. `https://www.bbc.com/sport/athletics/`

52759157#:~:text=Belfast%20man%20Derek%20Graham%20has,airbrushed%20from%20Irish%20sporting%20history.&text=Arising%20from%20that%2C%20Athletics%20Ireland,to%20its%20national%20records%20timelines, May 2020.

[Hie13] M. Hiebl. Management accounting in the family business: tipping the balance for survival. *Journal of Business Strategy*, 34(6):19–25, 2013.

[HKM11] P. Höfner, R. Khedri, and B. Möller. An algebra of product families. *Software & Systems Modeling*, 10(2):161–182, May 2011.

[Jat18] V. Jatana. *Dive Into Machine Learning*. November 2018.

[Jef19] A. Jeffries. Make sure to consider these common expenses when preparing your startup. https://smallbizclub.com/, August 2019.

[Jer12] R. Jervis. Politics and international politics scholarship. *International Studies Quarterly*, 56(3):623–625, 2012.

[JR11] B. Janet and A.V. Reddy. Cube index for unstructured text analysis and mining. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, pages 397–402, New York, NY, USA, 2011. ACM.

[Kap01] C. Kapitzke. Information literacy: The changing library. *Journal of Adolescent and Adult Literacy*, 44(5):450–456, 2001.

[KBLK10] A. Khan, B. Baharudin, L.H. Lee, and K. Khan. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1):4–20, February 2010.

[Kim18] S. Kim. Developing the strategies for ai products based on the technology decomposition framework. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1250–1254, 2018.

[KL05] B.Y. Kang and S.J. Lee. Document indexing: a concept-based approach to term weight estimation. *Information Processing and Management*, 41(5):1065 – 1080, 2005.

[Kll20] M. Klle. Hailee steinfeld looks back on 'true grit' ahead of 10-year anniversary. `https://www.etonline.com`, May 2020.

[KNK+17] R. Kobal, F.Y. Nakamura, K. Kitamura, C.C. [Cal Abad], L.A. Pereira, and I. Loturco. Vertical and depth jumping performance in elite athletes from different sports specialties. *Science and Sports*, 32(5):e191 – e196, 2017.

[LMG09] A. H. Lashkari, F. Mahdavi, and V. Ghomi. A boolean model in information retrieval for search engines. In *2009 International Conference on Information Management and Engineering*, pages 385–389, 2009.

[LN10] J. Lind-Nielsen. Buddy: A bdd package. `https://sourceforge.net/projects/buddy/`, 2010.

[MBMN09] J.S. Mitchell, J. Beall, W.E. Matthews, and G.R. New. Dewey decimal classification. *Encyclopedia of Library and Information Science*, 2009.

[McR20] R. McRady. Sofia richie celebrates brother's birthday after kardashians throw scott disick a party. `https://www.etonline.com`, May 2020.

[Min12] M. Minch. Women and politics. *The Indian Journal of Political Science*, 73(3):489–492, 2012.

[MM09] K. Mertsalov and M. McCreary. Document classification with support vector machines. *ACM Computing Surveys (CSUR)*, pages 1–47, 2009.

[MMM11] S. Maiti, D.P. Mandal, and P. Mitra. Sentence ranking for document indexing. In *Pattern Recognition and Machine Intelligence*, pages 274–279, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[MR03] D. McCracken and D. Reilly. *Backus-Naur Form (BNF)*, pages 129–131. John Wiley and Sons Ltd., GBR, 2003.

[MS17] A. Moldagulova and R.B. Sulaiman. Using knn algorithm for classification of textual documents. In *2017 8th International Conference on Information Technology (ICIT)*, pages 665–671, May 2017.

[MT16] M. Munsamy and A. Telukdarie. Agile energy modelling: A business centric approach. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 40–45, Decemeber 2016.

[MT17] C. Medoh and A. Telukdarie. Business process modelling tool selection: A review. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 524–528, Decemeber 2017.

[Mun07] D. Munteanu. Vector space model for document representation in information retrieval. *Annals of Dunarea de Jos*, Decemeber 2007.

[Mur20] D. Murphy. Lindsay lohan cuts her own hair on 'haircut night in america' and jerry o'connell is 'starstruck'. https://www.etonline.com, May 2020.

[MV15] A. Mishra and S. Vishwakarma. Analysis of tf-idf model and its variant for document retrieval. In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 772–776, Decemeber 2015.

[oC20] University of Colorado. For blade runners taller doesn't necessarily mean faster. www.sciencedaily.com/releases/2020/02/200220182745.htm, February 2020.

[O'H04] C. O'Hare. Business gateway : Business support in lowland scotland. *Business Information Review*, 21(4):245–251, Decemeber 2004.

[Ort12] M. Ortuoste. Youth, life, and politics: Examining the everyday in comparative politics. *PS: Political Science and Politics*, 45(2), 2012.

[Osa20] Naomi osaka: Japanese tennis star becomes world's highest-paid female athlete. http://www.bbc.com/sport/tennis/52774191, May 2020.

[Pai20] G. Paige. Lady gaga and blackpink team up for dance track 'sour candy. https://www.etonline.com, May 2020.

[Par13] S. Parashar. Feminist (in)securities and camp politics. *International Studies Perspectives*, 14(4):440–443, 2013.

[Pas20] J. Passan. Minor league baseball cuts hundreds of players amid pandemic. https://abcnews.go.com/Sports/

minor-league-baseball-cuts-hundreds-players-amid-pandemic/story?id=70943386#:~:text=Minor%20League%20Baseball%20cuts%20hundreds%20of%20players%20amid%20pandemic%2C%20sources%20say,-By&text=Hundreds%20of%20minor%20league%20baseball,be%20canceled%2C%20sources%20told%20ESPN., may 2020.

[Pat20] L. Patringenaru. Engineers develop low-cost, high-accuracy gps-like system for flexible medical robots. www.sciencedaily.com/releases/2020/05/200518111721.htm, May 2020.

[Pre20] Canadian Press. Disney, seaworld announce plans for florida parks to reopen. https://ckpgtoday.ca/2020/05/27/disney-seaworld-announce-plans-for-florida-parks-to-reopen/, May 2020.

[PS19] D. Palka and K. Stecuła. Concept of technology assessment in coal mining. *IOP Conference Series: Earth and Environmental Science*, 261, 2019.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(oct):2825–2830, 2011.

[Rog15] S. Rogers. Symposium on the politics of immigration: Introduction. *Polity*, 47(3):279–282, 2015.

[Rog20] G. Rogin. We are grown men playing a child's

game. https://vault.si.com/vault/1963/11/18/we-are-grown-men-playing-a-childs-game, 2020.

[Roy19] E. Roy. Identify, analyze and mitigate the social risks to your business. http://www.tradeready.ca/2019/topics/feasibility-of-international-trade/identify-analyze-mitigate-social-risks-business/, April 2019.

[RS13] F. Ren and M.G. Sohrab. Class-indexing-based term weighting for automatic text classification. *Information Sciences*, 236:109–125, 2013.

[RS17] Z.E. Rasjid and R. Setiawan. Performance comparison and optimization of text document classification using k-nn and naïve bayes classification techniques. *Procedia computer science*, 116:107–112, 2017.

[SA18] D. Sehgal and A. Agarwal. Real-time sentiment analysis of big data applications using twitter data with hadoop framework. In *Soft Computing: Theories and Applications*, pages 765–772, Singapore, 2018. Springer Singapore.

[SB10] L. Schwindt-Bayer. Comparison and integration: A path toward a comparative politics of gender. *Perspectives on Politics*, 8(1):177–182, 2010.

[SC17] G. Sabarmathi and R. Chinnaiyan. Investigations on big data features research challenges and applications. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 782–786, June 2017.

[Seb02] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[Sha16] J. Sharfstein. Rethinking science and politics. *The Milbank Quarterly*, 94(1):93–42, 2016.

[Sha17] S. Sharma. Activation functions in neural networks. *Towards Data Science*, September 2017.

[SJD13] S. Sarangi, V. Jaglan, and Y. Dash. A review of clustering and classification techniques in data mining. In *International Journal of Engineering, Business and Enterprise Applications (IJEBEA)*, 05 2013.

[SJV17] S. Shinde, P. Joeg, and S. Vanjale. Web document classification using support vector machine. In *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, pages 688–691, September 2017.

[SSR12] R. Sagayam, S. Srinivasan, and S. Roshni. A survey of text mining: Retrieval, extraction and indexing techniques. *International Journal of Computational Engineering Research*, 2(5):1443–1446, 2012.

[SWLH16] N. Shanavas, H. Wang, Z. Lin, and G. Hawe. Supervised graph-based term weighting scheme for effective text classification. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1710 – 1711. IOS Press, 2016.

[TAAEG10] B. Tahayna, R.K. Ayyasamy, S. Alhashmi, and S. Eu-Gene. A novel weighting scheme for efficient document indexing and classification. In *2010 International Symposium on Information Technology*, volume 2, pages 783–788, 2010.

[TLH09] F. Towhidi, A. Lashkari, and R.S. Hosseini. Binary decision diagram (BDD). In *2009 International Conference on Future Computer and Communication*, pages 496–499, April 2009.

[Tro09] J. Trounstine. All politics is local: The reemergence of the study of city politics. *Perspectives on Politics*, 7(3):611–618, 2009.

[Uni20a] Purdue University. Football offensive linemen should start plays upright to avoid hits to the head. `www.sciencedaily.com/releases/2020/04/200421112543.htm`, April 2020.

[Uni20b] Texas A'&'M University. Design of inorganic materials for brain-like computing. `www.sciencedaily.com/releases/2020/03/200303155700.htm`, March 2020.

[VG17] K. Venkatram and M.A. Geetha. Review on big data and analytics – concepts, philosophy, process and applications. *Cybernetics and Information Technologies*, 17(2):3–27, 2017.

[WHW17] I. M. Wirawan, H. W. Herwanto, and I. D. Wahyono. Virtual programming laboratory um (vlpum) for programming competition. In *2017 5th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, pages 158–162, 2017.

[Zin20] J. Zinoman. Hannah gadsby's 'douglas' doubles down on the comedy of contradiction. `https://www.thestar.com/entertainment.html`, May 2020.

[ZLW13] Q. Zhang, Y. Lin, and Z. Wang. Cost-effective capacity migration of peer-to-peer social media to clouds. *Peer-to-Peer Networking and Applications*, 6(3):247–256, 2013.

[ZZ19] H. Zhizhou and J. Zhao. Application and analysis of computer simulation in machining process. *Journal of Physics: Conference Series*, page 052025, June 2019.