Enhanced Root Extraction and Document Classification Algorithm for Arabic Text

by

Amal Alsaad

A thesis submitted for the degree of

Doctor of Philosophy

Department of Electronic and Computer Engineering

College of Engineering, Design and Physical Sciences

Brunel University London

February 2016

Abstract

Many text extraction and classification systems have been developed for English and other international languages; most of the languages are based on Roman letters. However, Arabic language is one of the difficult languages which have special rules and morphology. Not many systems have been developed for Arabic text categorization. Arabic language is one of the Semitic languages with morphology that is more complicated than English. Due to its complex morphology, there is a need for pre-processing routines to extract the roots of the words then classify them according to the group of acts or meaning.

In this thesis, a system has been developed and tested for text classification. The system is based on two stages, the first is to extract the roots from text and the second is to classify the text according to predefined categories. The linguistic root extraction stage is composed of two main phases. The first phase is to handle removal of affixes including prefixes, suffixes and infixes. Prefixes and suffixes are removed depending on the length of the word, while checking its morphological pattern after each deduction to remove infixes. In the second phase, the root extraction algorithm is formulated to handle weak, defined, eliminated-long-vowel and two-letter geminated words, as there is a substantial great amount of irregular Arabic words in texts. Once the roots are extracted, they are checked against a predefined list of 3800 triliteral and 900 quad literal roots. Series of experiments has been conducted to improve and test the performance of the proposed algorithm. The obtained results revealed that the developed algorithm has better accuracy than the existing stemming algorithm. The second stage is the document classification stage. In this stage two non-parametric classifiers are tested, namely Artificial Neural Networks (ANN) and Support Vector Machine (SVM). The system is trained on 6 categories: culture, economy, international, local, religion and sports. The system is trained on 80% of the available data. From each category, the 10 top frequent terms are selected as features. Testing the classification algorithms has been done on the remaining 20% of the documents. The results of ANN and SVM are compared to the standard method used for text classification, the terms frequency-based method. Results show that ANN and SVM have better accuracy (80-90%) compared to the standard method (60-70%). The proposed method proves the ability to categorize the Arabic text documents into the appropriate categories with a high precision rate.

Table of Contents

I would like dedicate my thesis to my beloved parents
Munirah Alabdulrahman and Salman Alsaad

Acknowledgements

First, I would like to express my deep and sincere gratitude to my supervisor Dr. Maysam Abbod, whom I am greatly indebted to for the continuous advice and guidance during this research. I appreciate his patience, support, and encouragement, in which this research would not be completed without. I am very grateful for him for giving me the opportunity to do and achieve this PhD. Dr Maysam Abbod is an outstanding supervisor and I have been very fortunate to be a PhD student under his supervision. Also, thanks to my second supervisor Dr Tatiana Kalgenova for her great advice and encouragement which gave me the key to overcome the obstacles in which I faced during the journey of my PhD.

Many thanks to my dear parents and my beloved siblings, Amani, Abdullah and Imtinan for their love and support, for having faith in me, and being there for me at all times. A big thank you to my dear friend Hasna Alqahtani, for helping me gain a strong faith and confidence in myself, for her continuous and immense support, and for brightening my days and standing by me all the way. Last but not least, many thanks to my precious friends, Sara Althowaini and Amal Bensasi for their great support and encouragement, for cheering me up and putting a smile on my face when I most needed, and for being there for me in the good and the hard times through this rough journey.

Declaration

I certify that the effort in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree. I also certify that the work in this thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been duly acknowledged and referenced.

Signature of Student

[illegible]

Amal Alsaad

February 2016, London

List of Figures

List of Tables

List of Abbreviations

AC	Associative Classification
AFP	Agence France Press
AMI	Average Mutual Information
ANN	Artificial Neural Networks
CCA	Corpus of Contemporary Arabic
FS	Feature Selection
IDF	Inverse Document Frequency
IR	Information Retrieval
LDC	Linguistic Data Consortium
MSA	Modem Standard Arabic
M-SVM	Multi-Category Support Vecto Machine
SVM	Support Vector Machine
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TR	Trigger Classification
TULM	Topic Unigram Language Model

Chapter 1

Introduction

During the past few years, the construction of digitalized content is rapidly increasing, raising the demand of information retrieval, text classification and automatic data tagging applications. There are few researches in this field for Arabic data due to the complex nature of Arabic language and the lack of standard corpora.

Text classification, which is also known as text categorization or topic identification, is the assignment of discovering if a piece of text belongs to any of a predefined set of classes [1]. Another definition states that the goal of text classification is to learn classification methods which can be implemented to classify documents automatically [2]. Text classification requires the use of text pre-processing methods to represent the text before processing text classification. Such methods include text stemming methods, such as removal of insignificant characters, affixes and stop words.

However, most work focuses on improving Arabic stemming algorithms, or topic identification and classification methods and experiments. No work has been conducted to include an efficient stemming method within the classification algorithm, which would lead to more efficient outcome.

In this thesis, an advanced stemming algorithm for Arabic text is developed. In addition, a new approach to identify significant keywords for Arabic corpora is presented. As well, different text classification methods are implemented using the extracted keywords as the main features for classification. At last, the results of the classifications methods are evaluated and compared to each other in terms of accuracy and efficiency.

1.1 Motivations

The rapid increase of digitalized textual data has raised the demand for text mining and national language processing tools and methodologies, to represent these data in an as efficient way as possible. This requires feature extraction and different implementations of text processing algorithms for representing the data as required. Text mining methods and algorithms are used in many different information retrieval systems such as search engines, clustering, classification, and other text mining systems.

Arabic language is the 5th amongst the most used languages around the world, and according to Internet World Stats (www. internetworldstats.com), it is the 4th most used language with more than 168 millions of online users as of November 2015, as seen in Figure 1.1. In addition, Arabic is on the top of the list as the fastest growing language on the web, with a growth rate of 6,592.5% between 2000 and 2015, as shown in Figure 1.2.

This indicates that the construction of Arabic data is growing rapidly, rising the need for standard data mining and natural language processing tools to represent and analyze this data in the most efficient way. Yet because of its complex morphological structure there were no available standard Arabic text mining and morphological analysis tools until recently [3]. However, many studies have been conducted to get efficient stemming results in Arabic Information Retrieval systems [4].

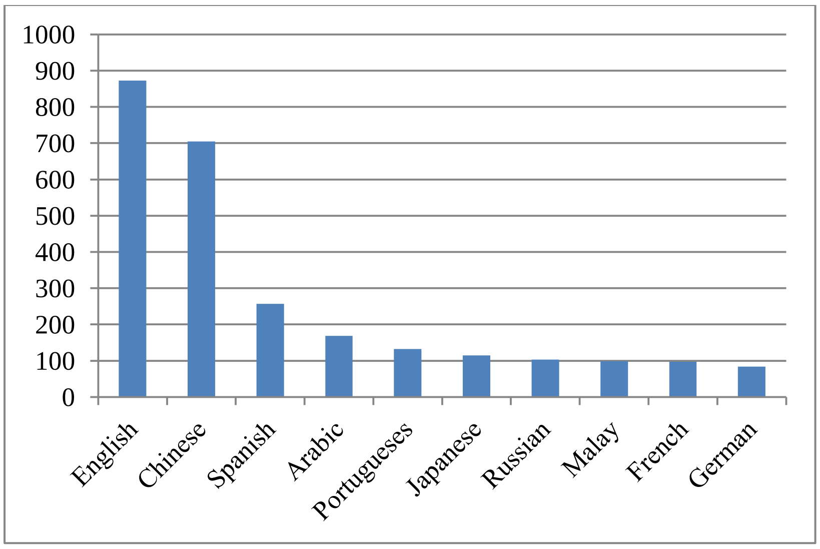


Figure 1.1. Top Ten Languages in the Internet 2015 – in millions of users.

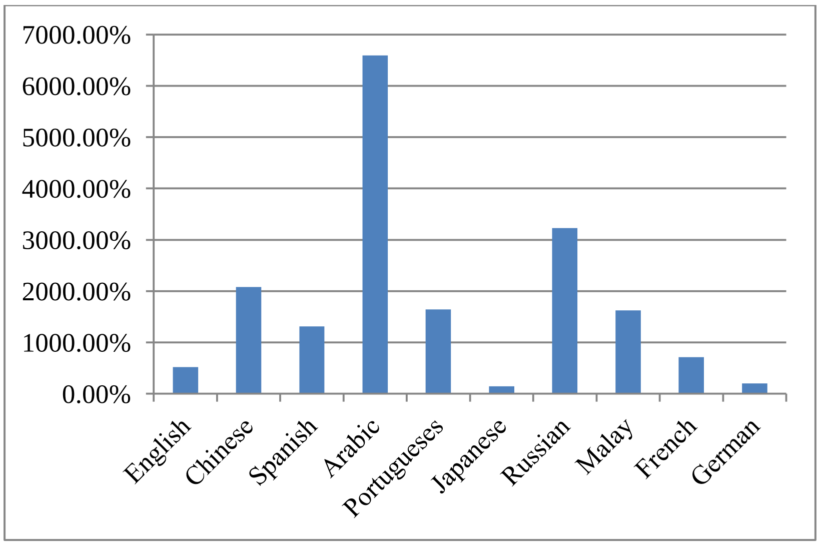


Figure 1.2. Percentages of Users Growth in the Internet by Language between 2000 – 2015.

In Arabic, both orthography and morphology lead to a great amount of lexical variations where one given word can occur among a large number of dissimilar forms [5]. This would enlarge the indexing structure volume and reduce the performance of the system. Another difficulty in Arabic morphological analysis arises because of the different letter forms of the tri-literal verbs. Verbs, which have tri-literal roots, are categorized to sound and unsound verbs [6]. Sound (صحيح) verbs are those which do not include a vowel nor a long vowel in their root. Sound verbs are divided into three different types, consonant verbs, double-lettered (geminated) verbs and hamzated verbs. Consonant verbs are those that do not have a *hamza* in their roots nor any duplicate letters like the verb (نَصَرَ), and they represent 58% of tri-literal root types popularity in Arabic language as can be seen in Figure 1.3 below. Double-lettered (geminated) verbs are those that end with two identical letters in there root such as (مَدَّ), representing about 8% of Arabic root types popularity. The third type of sound verbs includes verbs that have a *hamza* in their root like the verb (أَكَلَ). As shown in Figure 1.3, the hamzated letter could be in the beginning (*hamzated faa*), in the middle (*hamzated ain*) or at the end (*hamzated lam*), and these represent about 7% of Arabic root types popularity.

Unsound verbs are divided into three types, quasi-sound verbs, hollow verbs and defective verbs. Quasi-sound verbs are those where the first root letter is a weak letter such as (وَعَدَ). Hollow verbs, are verbs whose second root letter is a weak letter, for example the verb (قال). The middle weak letter of these roots can be a *waw*, *yaa* or *alif*, and the *mid-waw* and *mid-yaa* types of verbs represent around 12% of Arabic root types popularity as can be seen in Figure 1.3. Defective verbs are verbs whose third root letter is a weak letter such as the verb (دَعا). In some cases, weak verbs are written with a long vowel that is different from the one of their root following specific Arabic linguistic rules. In other cases, long vowels in verbs are deleted depending on the tense of the verb, becoming eliminated-long-vowel verbs. These cases represent about 30% of the Arabic text [7]. Yet the majority of Arabic stemmers lack the capability of handling these cases. In Figure 1.3, the popularity percentage in Arabic text of different verb cases is presented [5].

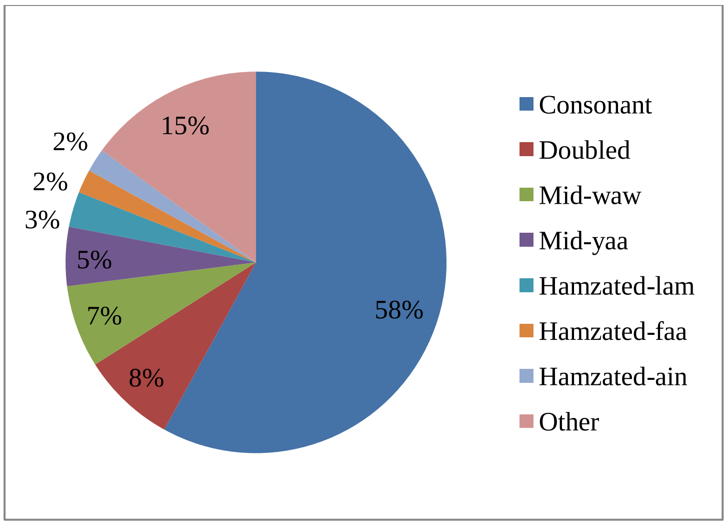


Figure 1.3. Tri-literal Root Types Popularity in Arabic Text

1.2 Contribution to Knowledge

In this thesis, it is contributed to the field by introducing an advanced root extraction algorithm for Arabic text, as well as developing and testing a classification system for Arabic text which employs an advanced root based stemming method. The root extraction algorithm is derived following two main steps. The first is to process the text to remove affixes from the text, including prefixes, suffixes and infixes. At the same time as removing affixes from the text, the word is checked against its morphological pattern depending on its length after each deduction. If a morphological pattern is matched, the root will be extracted. Otherwise, the process continues to the second step of the root extraction, that is where the algorithm is extended to handle different types of words. These include weak, *hamzated*, eliminated-long-vowel and two-letter geminated words. After the roots are extracted, they are verified by checking them against a list of predefined roots, containing 3800 trilateral and 900 literal roots. The performance of the algorithm is then tested by performing a number of experiments. The results which are achieved shows that the introduced algorithm is more accurate than the root extraction algorithms that are developed until now.

The second contribution is the document classification where the root extraction algorithm is employed in the stemming stage of the classification. In this part, two

main non-parametric classifiers are tested. These classifiers are Artificial Neural Networks (ANN) and Support Vector Machine (SVM). The system is trained on six different categories: culture, economy, international, local, religion and sports. The system is trained on 80% of the collected database. From each category the top ten frequent terms are selected as features. Testing the classification methods has been done on the remaining 20% of the documents. The results of ANN and SVM are compared to the standard method used for text classification, the terms frequency-based method. The results obtained indicate that ANN and SVM have better accuracy (80-90%) compared to the standard method (60-70%). The proposed method proves the ability to categorize the Arabic text documents into the appropriate categories with a high precision rate while selecting the top features of each category and employing the root extraction algorithm in the text preprocessing stage.

1.3 Aims and Objectives

In this work, the main aim is to introduce a root algorithm that handles the cases as shown in Figure 1.3, as well as affixes removal via implementing morphological analysis techniques and specific linguistic rules. It is also intended to deliver a new approach to identify significant keywords for Arabic corpora, and implement and evaluate different text classification methods. The aim of this project will be achieved by the following objectives:

- Literature review of previous work related to Semitic Languages Data Mining, Arabic Text Root Extraction and Text Classification algorithms.
- Derive an efficient algorithm for the Arabic Text Root Extraction system via a linguistic approach handling weak, hamzated, eliminated-long-vowel and two-letter geminated words.
- Design and building of the Root Extraction System, as well as conducting a series of simulation experiments to test the accuracy of root extraction and overall performance of the system.
- Derivation and implementation of the Feature Selection algorithm, where features of text are selected to be employed for classification.
- Conduct further research on Arabic Text Classification methods, and implement efficient and suitable classification algorithms comprising the Root Extraction system. These include Terms Frequency-based method, Artificial Neural Networks (ANN), and Support Vector Machine (SVM).
- Selecting and collecting of Arabic data sets and corpuses to evaluate and to test and evaluate the classification algorithms.
- Design and building of each classification algorithm.
- Design and carry out a series of simulation experiments to test the accuracy and efficiency of the classification systems.

1.4 Thesis Structure

The thesis is divided in 6 chapters which are as follows:

- Introduction: This chapter talks about the research briefly, the objectives of the research, the place which the research is designed for, the addition to knowledge and the research outline.
- Literature Review: This chapter illustrates some previous studies in the research field including Arabic language morphology.
- Arabic Language Characteristics: In this chapter, the language complexity is explained including its orthography and morphology which made it challenging to find the words roots using standard text mining algorithms. The challenges are elaborated and the challenges are demonstrated with associated examples for clarification.
- Roots Extraction: Roots extraction methods are reviewed in this chapter and tested. A new methodology is designed and developed that is more accurate than the existing methods. The results show the improvements of the proposed methodology.
- Document Classification Methods: This chapter shows three text classification methods, namely, terms frequency-based method, support vector machine, and neural networks. The results for all the methods are presented and compared eventually.
- Conclusion and Future Work: This is the final chapter which illustrates the stages of the research in brief and presents some ideas that may improve the system in the future.

1.5 List of Publications

Parts of the work detailed in this thesis have been presented in national and international scholarly publications, as follows:

1. Amal Alsaad, The Research Student Poster Conference 2012, held on the 14th March 2012.
2. Amal Alsaad, and Maysam Abbod. Brunel University's School of Engineering and Design Research Conference, ResCon12, 18-20 June 2012, Brunel University.
3. Amal Alsaad, Maysam Abbod and Tatiana Kalgenova. "Arabic Textual Data Mining via Machine Learning & Linguistic Constraints". Brunel University's School of Engineering and Design Research Conference, ResCon13.
4. Amal Alsaad, Maysam Abbod and Tatian Kalgenova. "Enhanced Root Extraction Algorithm for Arabic Text" Brunel University Research Student Conference, Showcasing the best in postgraduate research, 11–12 March 2014.
5. Amal Alsaad, and Maysam Abbod. "Arabic Text Root Extraction via Morphological Analysis and Linguistic Constraints". The 16th UKSim-AMSS International Conference on Modelling and Simulations, 26-28 March 2014. Cambridge University, Emmanuel College.
6. Amal Alsaad, and Maysam Abbod. "Enhanced Topic Identification Algorithm for Arabic Corpora". The 17th UKSim-AMSS International Conference on Modelling and Simulations, 25-27 March 2015. Cambridge University, Emmanuel College.

Chapter 2

Literature Review

In this chapter, a discussion of previous work which has been conducted to develop Arabic topic identification and documents classification methods is presented.

2.1 Arabic Stemming and Root Extraction

Arabic morphology and root extraction is a significant aspect of Arabic information retrieval and data mining. Many scholars and researchers have long noted the complexity of written Arabic language, as compared to English and many other European languages which have traditionally been used in testing. Heavy inflection distinguishes meanings of words, as do patterns of prefixes, suffixes and even infixes. As a result, morphological analysis is an extremely important pre-processing step in the preparation of any Arabic text for analysis. In recent years, an increasing number of studies have been conducted to examine and evaluate methodologies for Arabic stemming, Arabic morphological analysis and Arabic text root extraction [8] [9] [10].

The main two approaches consist of light stemmers, and root-based stemmers [9]. Light stemmers are employed mainly in information retrieval, where the main concept is to eliminate prefixes and suffixes from a word, generating a stem. In that way, the ideal forms of representative indexing for words is derived [11]. In Figure 2.1, is an example of the general steps of Arabic light stemming algorithms [3]. The reason why light stemmers are not concerned with root extraction is that words variants do not always have the same meaning even if they were generated from the same root [5] [12]. Thus, light stemmers intend to improve reduction of feature/keyword whilst maintaining the exact meaning of the word.

1. Normalize word
   - Remove diacritics
   - Replace آ، أ، إ with ا
   - Replace ة with ه
   - Replace ى with ي
2. Stem prefixes
   - Remove prefixes: الـ ، والـ ، بالـ ، كالـ ، فالـ ، للـ ، و
3. Stem suffixes
   - Remove suffixes: ها ، ان ، ات ، ون ، ين ، ية ، ه ، ي

Figure 2.1. Steps of Arabic Light Stemming Algorithms

The second approach is root-based stemming, where roots of the words are extracted by defining morphological analysis techniques. As the roots are extracted, the words are then grouped accordingly [9]. The fundamental two steps in root-based stemmers are to firstly remove prefixes and suffixes, and to secondly extract the roots by analyzing the words depending on their morphological components. That is usually achieved by identifying rule-based techniques, patterns table lookup, or by a combination of both. Root based stemmers take into account that Arabic complex morphology leads to a great amount of lexical variation, and as mentioned previously, this would enlarge the indexing structure volume and reduce the performance of the system. Therefore, root-based stemming is ideal for minimizing the index volume while enhancing the system performance, bearing in mind that the meanings words which are formed of the same root are similar. As well, tri-literal roots represent the vast majority of Arabic language roots, comprising the basis of 80-85% of all Arabic words. Another cause why root-based stemmers are used is that words entered as user query in information retrieval systems does not exactly match those included in the relevant documents [13]. In Table 2.1 shows a simple example of words of the same root processed by both a light stemmer and a root-based stemmer [14].

Word	**Meaning**	**Light stemming output**	**Meaning**	**Root-based stemming output**	**Meaning**
المكتبة	the library / the bookcase	مكتبة	library / bookcase	كتب	to write
الكاتب	the writer	كاتب	writer	كتب	to write
الكتاب	the book	كتاب	book	كتب	to write

Table 2.1. Outputs of Light Stemming vs Root-based Stemming.

In 2010, M. K. Saad and W. Ashour carried out a study evaluating the different stemming algorithms available to date for use with Arabic language text [3]. Regarding to light stemming, the researchers emphasised that this approach is usually used to ensure that the original meaning of the words is retained, despite having the same root and similar meanings in many cases. One example of standard Arabic light stemming tools is implemented in Apache Lucene in Java.

With regards to root-based stemmers, the researchers noted the tri-literal root-based stemmer proposed by Al-Shalabi et al. in 2003 [15]. Al-Shalabi, Kanaan, and Muaidi have developed a root extraction algorithm for tri-literal roots, which does not make use of any dictionary. The algorithm counts on giving weights to a word letters, for each letter, the weight is multiplied by the position of it. Consonant letters were weighted of zero, where different weight values were assigned to the letters in the word *sa'altumuniha* - سألتمونيها, as affixes are formed by a mixture of these letters. Specific computations are then performed over these weights to extract the correct root. Another similar stemming algorithm for Arabic tri-literal words is Al-serhan stemmer [16]. It employs Back Propagation Neural Network to extract roots from five letters Arabic words. Four types of input were generated encoded with binary digits, one relates to the original letters, while the other three classify the letters group of the word *sa'altumuniha* - سألتمونيها depending on their occurrence frequency as an affix letter.

Saad and Ashour have also evaluated Khoja’s root-based stemmer, which is one of the earliest and most well-known techniques developed for Arabic text stemming that was introduced in 1999 [17]. Referring to a study by Sawalhi and Atwell, Khoja stemmer was found to achieve the best performance in terms of accuracy for tri-literal roots, with an accuracy measure of 75% [18]. However, in the field of natural language processing, accuracy is significantly important, and 75% still falls below the target.

Also, Saad and Ashour identified a number of weaknesses in Khoja’s stemming algorithm [3]. Khoja’s stemmer eliminates the longest suffix and the longest suffix and prefix, and then matches the rest of the word against a list of verbal and noun patterns to extract the root. Another step is performed to check the correctness of the root by checking it against a list of roots. If the extracted root is found, it is then preserved as the root of the word. The stemmer utilizes a number of linguistic data files, including lists of all diacritic characters, punctuation characters, definite article, and stop words. The stemmer also handles some cases of Arabic tri-literal words that are weak, hamzated, geminated or eliminated-long-vowel. But the algorithm has a number of weaknesses. Firstly, the word *munaddamat* - منظمات which means (organizations) is stemmed to the root *dama-aa* - ظمآ which means (he became thirsty) instead of the correct root *nadama* - نظم. Another issue is that when the word is deducted to a tri-literal word, the weak letter is deleted in the first place, and then the last letter is doubled, or another weak letter or an *alif* is added to the word. That leads to extracting a root that is of another word, which is not related to the word. For example, the extracted root of the word *riwayat* - روايات is *rayaya* - ريي, where the correct root is *rawaya* - روي. As well, the extracted root for the word *akhar* - آخر is *kharara* - خرر, where the correct root is *akhara* - أخر. The use of a dictionary, for instance, entails maintenance to ensure that as new words are discovered they continue to be handled correctly. While rigid adherence to the processing order of affixes can similarly result in errors, for example the words *tastaghrik* - تستغرق and *rukbataih* - ركبتيه were found not to be stemmed to their roots *gharaka* - غرق and *rakaba* - ركب, respectively.

Lastly, the researchers offer new on-line tools for scholars and evaluators, integrating stemming and light-stemming algorithms and a stop-words list into the popular Machine Learning and Data Mining tools WEKA and RapidMiner.

Another study on Arabic stemming and root extraction was performed by E. M. Saad, M. H. Awadalla and A. Aljami in 2010 [19]. The study focussed on the challenge of text representation, extracting features that are most representative of the text, and mainly on Arabic verb pattern extraction. Proponents of the value of patterns in accuracy of feature extraction, the researchers proposed an enhanced approach to extraction of both the stem and the lemma, using pattern matching. Their approach implemented an encoding scheme distinguishing original from non-original letters which was generated for each pattern and then matched against input text to extract the root, pattern and lemma of a word. Yet the corpus used for their study was not specified.

The first step in the proposed process is to divide the words into two groups – one consisting of letters that are likely affixes and the other consisting of letters that are likely an original part of the stem. Each letter in the affixes group is then assigned a unique code, whilst the original letters were all assigned the code "0000". The researchers also retained and assigned to the *shaddah* (ّ◌) a unique code, in contrast to many approaches, which opt to repeat the letter before the *shaddah*, noting that in some patterns it serves as one of the distinguishing letters. A sample of the encoding table used is shown in Table 2.2. A pattern code table is then generated from all possible combinations of the letter codes, in which the letters "ف", "ع" and "ل" are the stem characters. This results in each pattern having 1260 possible combinations. Examples given are "فعل", which is encoded as "000100001001", and "فاعل", which is encoded as "0001**1010**00001001", wherein the "**1010**" represents the added "ا".

Letter	Code	Letter Pronunciation
س	0001	Seen
أ	0010	Hamza
ل	0011	Lam
ت	0100	Ta’a
م	0101	Meem
و	0110	Waw
ن	0111	Noon
ي	1000	Ya’a
ه	1001	Ha’a
ا	1010	Alef
شدُة	1011	Shaddah
others	0000	-

Table 2.2. Letters Encoding for the Word (سألتمونيها).

Arabic morphological rules are then applied to accomplish further reduction of the codes for each pattern. For example, some patterns such as معتل ،صحيح and مهموز, do not exist in some verb forms and therefore can be eliminated from the code table. Samples of input and output encoding for lemmas and patterns are shown in Table 2.3.

Input	Output code	Pattern	Lemma
0001-0000-0000	1-1-1	فعل	فعل
0001-0000-0000-0100	1-1-1-0	فعلت	فعل

Table 2.3. Example of Encoding for Lemmas and Patterns.

The researchers then applied their algorithm which we summarise in the following steps:

Step 1: generation of codes table

- Eliminate incorrect case records

Step 2: normalization

- Read a word
- Eliminate diacritics, except *shaddah*
- Change all *hamzah* forms to (أ)

- Change *maddah* to *hamzated alef* (أ)
- Change *alef maqsora* to *alef* (ا)

Step 3: encoding and matching

- Encode input word as in encoding table
- Match input word against the code table to extract codes of equal value (root, pattern, and lemma)
- If there is more than one code match, then go to post processing

Step 4: post processing

- Manual check for the highest percentage of occurring patterns using conjugation reference work

In analysing their results, the researchers ran the same data through a Khoja stemmer for comparison. They found that the proposed method achieved an overall rate of accuracy of 92%, which increased to 96% after adding in the post-processing step. A comparison of the two methodologies' results by type is shown in the table below. The researchers' proposed methodology using encoding and pattern matching appears to have achieved a higher rate of accuracy than the Khoja stemmer.

Type	**#Tested Words**	**Proposed Method**	**Khoja's Method**	
			Not Stemmed	**Accuracy**
سالم (Salem)	1019	99%	256	70%
ناقص (Nakes)	478	89%	41	50%
أجوف (Ajwaf)	962	98%	72	61%
مثال (Methal)	647	97%	71	69%
مهموز العين (Hamza-Ain)	368	98%	153	48%
مهموز اللام (Hamza-Lam)	404	99%	126	58%
مهموز الفاء (Hamza-Faa)	610	96%	63	73%
Total	4488	96%	782	61%

Table 2.4. Results of Proposed vs Khoja's Method.

Likewise, Al-Nashashibi, Neagu and Yaghi have developed an improved root extraction technique for Arabic words, while recognising the prevalence of irregular words in Arabic texts [10]. They determined that irregular forms such as weak, two-letter geminated, hamzated and eliminated-long-vowel words accounted for about 30% of Arabic texts. Table 2.5 summarises their analysis.

Form	**Percentage**
weak	13%
two-letter geminated	7%
hamzated	11%
eliminated-long-vowel	2%*
Total	**33%**

* of weak words: 12%

Table 2.5. Percentages of Irregular Words in Arabic.

Taking into account that many of the stemmers developed since Khoja's stemmer did not further address these areas, in 2010 the researchers proposed an approach to root extraction which met this need by focussing on improvements in the variety of irregular linguistic cases handled [10]. Their approach used a rule-based light stemmer based on the model developed by Al-Ameed [20]. The stemmer included a pattern-based infix remover and then was enhanced with an algorithm to address weak, eliminated-long-vowel, hamzated and geminated words. Accuracy of the roots extracted was checked against a pre-defined list of 5,405 tri-literal and quad-literal roots.

For their study, the researchers used a corpus consisting of a collection of 380 texts taken from online Arabic sources (newspapers, magazines, academic and other) published between 23/7/2008 and 1/2/2009, covering nine different categories. These are: politics, economics, social, sports, music, education and health, arts, culture and literature, and religious texts. The distribution of the collection was approximately fifty texts per category. As a pre-processing step, English letters, punctuation, nunations, assimilation marks, short vowels, kasheeda, function words and numerals were removed from the experimental data set. For their function words list, the researchers developed a list of 2,549 words including prepositions, pronouns, conjunctions, interjections and verbs such as *kana wa akhawatuha* - كان و أخواتها and similar verbs such as *asbaha* - أًصبح, *amsa* - أًمسى and *ma zala* - مازال. The plural and dual forms of the function words were included in the list as well. The Al-Ameed root extraction process used for this study consists of two parts: a rule-based light stemmer and a pattern-based infix remover. The light stemmer removed affixes using four pre-defined lists of prefixes and two pre-defined lists of suffixes, as shown in Table 2.6 [10].

Proposed Prefix lists	***Prefix List1***	*{ A, t, n, w, y, m, k, b, f, l }*
	Prefix List2	*{ Al, ll, sy, sA, st, sn, kA, fA, bA, ly, lt, ln, ft, fy, fn, wt, wy, wm, wA, wb, bm, km, mt, fl, An, mn, lA, wl, wn, wk, fb, fm, lm, yt, yn, tt, tn, tm, nt, nn, At }*
	Prefix List3	*{ wAl, bAl, fAl, kAl, wll, wsy, wst, wsn, wsA, wlA, wly, wlt, wln, fsy, mst, Ast, Alm, AAl, sAl, fAn, AA, yst, nst, tst }*
	Prefix List4	*{ wbAl }*
Proposed Prefix lists	***Prefix List1***	*{ A, t, n, w, y, h, k, Y, p}*
	Prefix List2	*{ An, yn, wn, At, hm, hn, hA, km, kn, nA, wA, tm, ny, tn, th, yh, A', yA, tA, tk, yp, np, nh, ty, Aw, nk, hmA, kmA, thm, tkm, wt }*

Table 2.6. Al-Ameed's Prefix and Suffix Lists.

After that, infixes were removed based upon pattern-matching tri-literal and quad-literal words to pre-defined lists, as shown in Table 2.7.

Triliteral Infix Patterns	*{ fEl, fAEl, fAEwl, fEA'l, fEAl, fEwl, fEyl, fwEl, fwAEl, fwAEyl, fyEl, fyAEl, fEyEyl, fEyEAl, fwyEl, ftEl, ftEAl, fAEyl, fEwEl }*
Quadliteral Infix Patterns	*{ fEll, fEAlyl, fEAll, fElAl, fElwl, fElyl, fEyll }*

Table 2.7. Al-Ameed's Infix Pattern Lists.

After removing the infixes, the resultant roots were matched against the predefined list of tril-iteral and quadliteral roots. In cases where a match was not found, the researchers' "correction algorithm" was applied. The correction algorithm contains 5,737 corrections words in 71 predefined lists, based on linguistic works by Ar-Rhazi [21], and Bayyomee, Kolfat and Al-Shafe'e [22]. These rules target in particular irregular words. For example, in weak words, the long vowel was replaced according to Arabic grammar rules. For two-letter geminated words, eliminating the extra letter where found and doubling the appropriate stem letter. However, it is not given how the algorithm handles the cases of neither eliminated-long-vowel words, nor words containing *hamza*.

As indicated above, the accuracy of the output was determined by matching each root against predefined lists of tri- and quad-literal roots, and then counting the number of successful matches. As well, it does not appear that the researchers did any manual checking of those matches, to ensure that they were indeed correct. As a measure of comparison, the same data was run through the Al-Ameed process only, without the researchers' correction algorithm. The full results by category are shown in Table 2.8, wherein the authors use "RB-A" to refer to the test using the Al-Ameed method alone and "RB-A-corr" to refer to the test using the Al-Ameed method plus their correction algorithm.

Category	**RB-A (%)**	**RB-A-corr (%)**
Politics	58.89	73.3
Economics	58.16	71.39
Religious Texts	62.99	75.01
Social	60.56	74.79
Music	58.7	73.78
Educational	60.67	74.81
Sports	56.91	70.37
Arts	61.41	74.27
Average	***59.79***	***73.47***

Table 2.8. Performance of the Algorithm in All Categories.

The researchers found that on average the correction algorithm improved the accuracy of the results by about 14%, with a relative improvement of about 23%. They identified also some limitations in the rule lists for future improvement. For example, adding rules for extracting two-letter geminated roots and more infix patterns. For fuller context, the researchers proposed applying the correction algorithm to various other stemmers in future tests.

Last but not least, in 2007 researchers Darwish and Oard developed two different tools which aimed at improving Arabic morphology techniques used in Information Retrieval (IR) [23]. The first tool they developed, dubbed Sebawai, is a shallow morphological analyser which generates stems and probability measures. The second tool, dubbed Al-Stem, is a light stemmer which utilises the probability statistics generated by Sebawai to develop the set of prefixes and suffixes to be removed. By combining the roots and stems generated by Sebawai with the stems generated by Al-Stem, the researchers intended to improve upon what they had identified as weaknesses in prior-developed methodologies based upon finite state transducers. That is where uncertainties were being introduced into indexes, and the datasets that could be analysed were limited.

The authors' Sebawai analyser was based on the commercial Arabic morphological analyser ALPNET, which was developed by Beesley and Buckwalter. In their work, rules are programmed into a finite state transducer using a set of 4,500 roots [24] [25] [26]. However, Sebawai trains on a list of word-root pairs to derive templates to produce stems from roots, construct a list of prefixes and suffixes, and estimate the occurrence probabilities of templates, stems and roots. The word-root pair list to be used with a tool such as Sebawai could be one that is already available in a pre-existing analyser, generated manually, or automatically constructed through the parsing of a dictionary. For their study, the researchers used the pre-existing list provided by ALPNET, which in turn generated this list from two other lists. One list which is extracted from a corpus of Arabic text taken from a 14th century religious work entitled Zad al-Me'ad (زاد المعاد), and contained more than 9,606 words. The other list was extracted from the Linguistic Data Consortium (LDC) Arabic corpus of newswire stories and containing 562,684 words.

Words from the first list which were not successfully analysed by ALPNET were excluded. The total number of words which were successfully analysed was 9,606 words. While words from the LDC corpus which were not successfully analysed, a total of 292,216, were separated from those which were successfully analysed (270,468), and two lists were then retained: "LDC-Pass" and "LDC fail". It was found that ALPNET tended to fail on words which are of the following types:

a- Named entities and Arabized words, which are words that are adopted from other languages. An example of these includes the word *dymwqratiyah* - ديموقراطية (Democracy).

b- Misspelled words.

c- Words with roots not in the root list: An example of that is the word *jawaĎ* - جواظ (seldom used word that means pompous).

d- Words with templates not programmed into the finite state transducer. ALPNET uses a separate list of allowed templates for each root. These lists are not comprehensive. An example of that is the word *musaylamah* - مسيلمة (miniature of *mslmh* – مسلمة, a person who is safe or peaceful).

e- Words with complex prefixes or suffixes: An example of that is the word *bilkhurafat* - بالخرافات (with the superstitions).

Nonetheless, words that were successfully analysed were rechecked by manual verification and were found to be correct in all cases. As well, in cases where words yielded more than one analysis, all combinations of the word plus all roots for each of those combinations were found correct and were also utilised for the training.

During training, the Sebawai analyser aligned the characters in word pairs and applied regular expressions in order to isolate the prefix, suffix and stem template. As an example, the authors gave the pair *kataba* - كتب and *wakitabahum* - وكتابهم, from which would be ascertained the letter *waw* - 'و' (meaning and, written as wa in the beginning of the word) as the prefix, and *hum* - 'هم' as the suffix and CCAC as the stem template, where "C" represents the letters pertaining to the root. Simultaneously, the training module increased its number of observed occurrences of the prefix, suffix and stem template by one. At the end of the training, probabilities were calculated using equations (2.1-2.3), where *S1* and *S2* represent character strings and *T* represents a template:

$$P(S1\ begins\ a\ word, S1\ is\ a\ prefix) = \frac{(No.of\ words\ with\ prefix\ S1)}{(Total\ No.of\ training\ words)} \qquad (2.1)$$

$$P(S2\ begins\ a\ word, S2\ is\ a\ suffix) = \frac{(No.of\ words\ with\ suffix\ S2)}{(Total\ No.of\ training\ words)} \qquad (2.2)$$

$$P(T\ is\ a\ template) = \frac{(No.of\ words\ with\ Template\ T)}{(Total\ No.of\ training\ words)} \qquad (2.3)$$

Within the root detection process, the Sebawai analyser reads in an Arabic word and generates all possible combinations of the prefix, suffix and template letters by breaking the word into three parts. In such that: the first part always consists of letters which constitute a valid entry in the list of prefixes, the last part always constitutes a valid entry in the list of suffixes, and the middle part contains no less than two letters. The first and last parts could also be null. As an example, the authors provided an example of all possible analyses for the word *Ayman* - ايمان is shown in Table 2.9.

Stem	**Prefix**	**Template**	**Suffix**	**Root**
ايمان - AymAn	#	فيعال - CyCAC	#	أمن - Amn
يمأن – ymAn	ا - A	فعال - CCAC	#	يمن – ymn
مأن – mAn	اي – Ay	فعل – CCC	#	مأن – mAn
أيم – Aym	#	فعل – CCC	ان - An	أيم – Aym
يمأ – ymA	ا – A	فعل – CCC	ن – n	يمأ – ymA

Table 2.9. Possible Analysis for the Word AymAn – ايمان .

If the stem is found to fit one of the templates, then a root is generated. The root is then checked against a list of approximately 10,000 roots generated from an Arabic dictionary. The probability for the root is calculated as in equation (2.4) :

$$P(root) = P(S1\ begins\ a\ word, S1\ is\ a\ prefix) * P(S2\ ends\ a\ word, S2\ is\ a\ suffix) * P(T\ is\ a\ template) \qquad (2.4)$$

While the word analysed is of two letters only, initial testing showed that the Sebawai analyser failed on all 2-letter stems, thus adjustments were made to compensate. Two-letter stems were ‘corrected’ by doubling the last letter and by adding weak letters before or after. Similarly, for stems with a weak middle letter, new stems were generated by substituting the middle letter with other weak letters. Additional probabilities were calculated for these three cases. The root probability formula was also adjusted to account for these additional probabilities through equation (2.5) as:

$$P(root) = P(S1\ begins\ a\ word, S1\ is\ a\ prefix) * P(S2\ ends\ a\ word, S2\ is\ a\ suffix) * P(T\ is\ a\ template) * P(letter\ substitution\ or\ letter\ addition) \qquad (2.5)$$

Another improvement made was the application of “smoothing”, or discounting erroneously-produced suffixes and prefixes resulting from misalignment during the word-root pairing process. To achieve that, the researchers employed the Witten-Bell discounting technique [23]. In addition, a list of particles was developed using the Arabic grammar book An-Nahw Ash-Shamil [27]. Also, a stem check against these particles was applied. The complete list of the particles is incorporated in the distribution of Sebawai by K. Darwish in 2002 [28]. Finally, a normalization strategy was employed to account for variations in spelling as well as to improve analysis performance. The strategy included the normalization of the letters *yaa* - ي and the letter *alif* - ا variants, as well as stripping all diacritics.

In order to evaluate and measure the accuracy and performance level of the Sebawai analyser, the researchers compared the results of Sebawai to those of ALPNET, considering matches as correct and non-matches as false. A further, manual check of the failures was then performed to confirm if the words were in fact incorrect, or simply not identified by ALPNET. For the first list, which was constructed with the aid of Zad al-Me'ad, Sebawai analyser analysed all words where the initial results showed that 8,206 roots out of 9,606 were found correctly (86.4%). Following manual evaluation, however, the actual number of correct roots was estimated to be 8,800 (92.7%). For the second (*LDC*) list, the analyser analysed 128,169 words out of 292,216 (43.9%) where 58,000 roots, (21%) only, were estimated to be correct.

After the prefixes and suffixes are generated by the Sebawai analyser along with their corresponding probabilities, they are used to build the researchers' second tool contribution, the Al-Stem stemmer. Affixes with a probability above 0.5 were accepted as candidates, and then the candidate list was manually confirmed. The contents of the final affixes lists are shown in Table 2.10.

Prefixes	**Suffixes**
وال، فال، بال، بت، يت، لت، مت، وت، ست، نت، بم، لم، وم، كم، فم، ال، لل، وي، لي، سي، في، وا، فا، لا، با	ات، وا، ون، وه، ان، تي، ته، تم، كم، هم، هن، ها، ية، تك، نا، ين، يه، ة، ه، ي، ا

Table 2.10. List of Prefixes and Suffixes.

At last, each of Sebawai morphological analyser and Al-Stem stemmer were evaluated in an experiment using the LDC2001T55 corpus collection, containing 383,872 articles from the Agence France Press (AFP) Arabic newswire of around 50 different topics, with a collection averaging 118 documents per topic. Each document in this corpus contains a title field, a description field and a narrative field. For the experiment, two queries were formed for each topic: one against the title and description fields (td), and one against just the title field (t). Tests were then performed for each length of query using index terms which consisted of words (w), lightly-stemmed words (lw), stems and roots. Two methods were used in obtaining the stems: either the top stem found by the Sebawai analyser or the top stem found by both the Sebawai analyser and ALPNET, if generated. Similar methods were used for obtaining the roots: either the top root found by the Sebawai analyser or the top root found by both the Sebawai analyser and ALPNET, if generated.

2.2 Documents Classification

One valuable contribution in the field of Text Classification came from a study by Abbas and Smaili's in 2005 [29]. The study focuses on comparing topic identification methods. Abbas and Smaili examined the effectiveness of the Term Frequency-Inverse Term Frequency (TF.IDF) approach versus the Support Vector Machines (SVM) approach in identifying topics within a corpus comprised of 5120 articles (2,855,069 words) taken from Akhbar Al Khaleej - أخبار الخليج daily newspaper, covering the topics of sport, local news, international news and economy.

For their experiments, the researchers defined "topic" as "a subset of the language associated to particular events", and a document would be considered relevant whenever its content is connected to the associated event. To begin, the researchers tokenize the corpus by splitting each word found into prefix, stem and suffix, and then remove the suffix. Non-content words were removed from the results as well. The frequency which each word appeared was calculated, as well as its "documents frequency", which is the number of documents in which the word appeared at least once. The results were then reduced to distinct words, and finally to those which appeared more than two times. The final vocabulary used consisted of 42,877 words. Next, an internal representation of each document was constructed by transforming it into compact vector form, whereby the dimension of the vector corresponded to the number of distinct tokens found, and each entry in the vector represented the weight of each token.

In the TF.IDF approach, each document *d* is represented as a vector ***D = (d₁, d₂,…,dᵥ)***. Each element in the vector represents the weight of a given word w_i in the document. That is calculated as ***d_i = TF(w,d) * IDF(w).*** That is Term Frequency ***TF(w,d)***), which is the number of times a word *w* occurs in a given document *d*, multiplied with the inverse document frequency (***IDF(w)***). ***IDF(w)*** is calculated as ***log(N/DF(w))***, where *N* represents the total number of documents and ***DF(w)*** represents the number of documents in which a word *w* occurs at least once. Equation (2.6) was used to calculate the similarity between documents (***Sim(Dj,Di)***):

$$Sim(D_j, D_i) = \frac{\sum_{k=1}^{|V|} d_{jk} d_{ik}}{\sqrt{\sum_{k=1}^{|V|}(d_{jk})^2 \sum_{k=1}^{|V|}(d_{ik})^2}} \qquad (2.6)$$

In the SVM approach, two vectors are used, and the relationship is defined by equation (2.7) as:

$$f(x) = \sum_{i=1}^{n} w_i * x_i + b \qquad (2.7)$$

where *w* is the vector orthogonal to the hyperplane and *b* is the distance from the hyperplane to its origin. For their experiment, the researchers used only linear kernels and the tool SVM^{light}.

The researchers evaluated their results based on recall, precision and the measure ***F_1 = 2*Recall*Precision/Recall+Precision***, as shown in Table 2.11.

Topic	**International**			**Local**			**Economy**			**Sport**		
	Rec	**Prec**	**F1**	**Rec**	**Prec**	**F1**	**Rec**	**Prec**	**F1**	**Rec**	**Prec**	**F1**
International	-	-	-	99.22	100	99.61	100	99.22	99.61	100	100	100
Local	99.22	100	99.61	-	-	-	89.06	92.68	90.83	97.66	99.21	98.43
Economy	100	99.22	99.61	89.06	92.68	90.83	-	-	-	97.66	100	98.81
Sport	100	100	100	97.66	99.21	98.43	97.66	100	98.81	-	-	-

Table 2.11. Recall, precision and *F1* for SVM bi-class discrimination

Traditional advantages of the SVM method are seen as the ability to handle a large number of features and to work with real and large-scale data, and Abbas and Smaili found this to be the case in their experiments as well. As seen in Table 2.12 SVM showed an advanced ability to discriminate topics, and outperformed TF.IDF with an overall F1 score of 97.88% versus 90.95%.

	Recall	**Precision**	**F1 measure**
TF.IDF	90.82	91.18	90.95
SVM	97.26	98.52	97.88

Table 2.12. Mean Values of Recall, Precision and F1.

Another important study on Arabic text classification that been carried out to evaluate different topic identification methods was done by Abbas, Smaili and Berkani in 2011 [30]. Abbas et al. evaluated six topic categorisation methods over a large corpus of Arabic data. The methods evaluated were the Support Vector Machines (SVM), Triggers-based Classifier (TR), Topic Unigram Language Model (TULM), Neural Networks, Term Frequency/Inverse Document Frequency and Multi-Category SVM (M-SVM). Their findings were that the SVM was clearly the superior method for their corpus. The study used an in-house corpus derived from a selection of articles from an online Arabic newspaper, consisting of 9,000 documents pertaining to six categories and totalling 9,813,366 words.

For their experiments, the text was pre-processed to remove punctuation, digits and stop words, and a light stemmer was also used. Smaller, category-specific vocabularies of about 300 words were built for testing the TR method, and a larger general vocabulary was built for use by the other five methods by concatenating the individual vocabularies. The researchers used the term frequency (TF) method in constructing the vocabularies, and the similarity function in equation (2.8) for the TF-IDF test:

$$Sim(D_j, D_i) = \frac{\sum_{k=1}^{|V|} d_{jk} d_{ik}}{\sqrt{\sum_{k=1}^{|V|} (d_{jk})^2 \sum_{k=1}^{|V|} (d_{ik})^2}} \qquad (2.8)$$

For the Neural Networks method, the researchers opted to use a separate network per category for training and a multi-layer perception for categorization. For the TR method, Average Mutual Information (AMI) values were calculated for each of the words in the vocabularies to determine the most important triggers. The researchers chose 250 triggers to use for each topic.

For the M-SVM method, the researchers used 1,400 documents for training and reserved 10% of the corpus for test, and the size of the vocabulary used was 8,000 words. Measures used in the evaluation of their results were recall, precision and F1. As seen in as seen in Table 2.13 and Table 2.14, the results of the study clearly showed SVM to be the superior method on all three measures:

Method / ***Topic***	**SVM**			**TR**			**TULM**		
	Re	Pr	F1	Re	Pr	F1	Re	Pr	F1
Culture	97.33	95.51	96.41	82.66	80.55	81.60	70.55	89.5	78.90
Religion	96.93	99.32	96.11	96.33	83.56	89.50	94	86.33	90.00
Economy	96.26	96.57	96.42	83.50	84.05	83.77	82.66	82.33	82.50
Local	96.13	96.55	96.34	86.25	82.53	84.35	78.33	80	79.15
International	98.26	96.88	97.56	93.33	90.66	91.97	94.50	85.66	89.86
Sports	99.20	99.59	99.40	96	97.33	96.66	93.66	98.33	95.94
average	**97.35**	**97.40**	**97.37**	**89.67**	**86.44**	**88.02**	**85.61**	**87.02**	**86.31**

Table 2.13. Performance of SVM, TR and TULM

Method / Topic	Neural Networks			TF-IDF			M-SVM		
	Re	Pr	F1	Re	Pr	F1	Re	Pr	F1
Culture	75	86.66	80.40	71.33	88.43	78.96	75	78	76.47
Religion	92	87.33	89.60	93.33	86.95	90.03	95	96	95.50
Economy	81.33	85.33	83.30	83.33	80.64	81.96	83.5	75	79.02
Local	75.25	90.50	82.20	80	76.92	78.43	74	64	68.64
International	92.55	83.33	87.70	93.33	84.33	88.60	86.75	83	84.83
Sports	94.50	90.66	92.54	94	100	96.91	90	89.5	89.75
average	**85.10**	**87.30**	**86.20**	**85.88**	**86.21**	**86.04**	**84.04**	**80.91**	**82.44**

Table 2.14. Performance of Neural Networks, TF-IDF and M-SVM

The researchers found it noteworthy that the M-SVM method appears to have achieved the lowest level of performance, given its strong theoretical background. However, they noted also that M-SVM is also generally used over a larger vocabulary. The researchers also noticed greater variability over certain topics, such as “Local”, particularly in terms of recall; for other topics, such as “Economy”, there seemed to be significantly more convergence of the performance indicators. They attribute the success of the TR method to the smaller size of its vocabularies, noting also an increase in performance as the number of triggers increases per topic.

Another method applied for text classification with various algorithms is Associative Classification (AC), and it was presented as a new approach for classification in the last decade of research on the subject. One of the latest studies on associative classification was published in 2012 by Ghareb, Hamdan and Abu Bakar [4]. In their study, the researchers compared the single rule prediction and multiple rule prediction methods of Associative Classification. Their findings showed superior accuracy from the multiple rule prediction method.

The researchers used corpora which consisted of 5640 documents derived from articles from a selection of Arabic online news sources and from two established online corpora: Dr. Mourad Abbas’ corpus of 2004 news articles from Al-Khaleej - الخليج and Al-Watan - الوطن, and Dr. Latifa Al-Sulaiti’s Corpus of Contemporary

Arabic (CCA). Significantly, the researchers believed this to be the largest Arabic-language corpora used in such experiments to date. The corpora consist of seven different categories; culture, economy, politics, sports, education, health and information technology.

The process of constructing the associative classifier starts with a text pre-processing phase, in which non-Arabic letters, digits, punctuation marks and stop words are removed. As well, a stemming process was also applied at this phase. However, no lists or details on the stemming process were specified. Next, the data undergoes a feature selection (FS) process, in which the researchers used the TF-IDF method to assign weight values and distinguish the more important features. From this point, class association rules were generated using an Apriori algorithm similar to that used by Antoine and Zaiane in their 2002 study [31]. In determining the importance of the class association rules, the researchers used measures for "minimum support" and "minimum confidence". The support threshold determines the minimum frequency of occurrence within the data. Each candidate term set, beginning with 1-term sets, then 2-term sets, and so on through the m-term set, was generated from terms in the previous set which passes the minimum support threshold.

The Support of the class association rules was calculated as the percentage using equation (2.9):

$$Support(T_j \Rightarrow C_i) = \frac{Support(T_j \Rightarrow C_i)}{N} \qquad (2.9)$$

where T_j corresponds to a set of frequent terms [*t1* & *t2* &…& *tm*] which represents the "rule terms", C_i signifies the Category of this rule which is the "rule head", $sup(T_j \Rightarrow C_i)$ is the number of documents in the data set that match terms of *R*, and *N* is the total number of documents in the class data set.

The accuracy of the class association rules, or “rule confidence”, was calculated by the probability as in equation (2.10):

$$Rule\_accuracy = \frac{(D_{tot}(R)+1)}{(D_{tot}(R)+N_c)} \qquad (\ 2.10\ )$$

where $D_{tot}(R)$ is the number of documents in the training set containing the rule body, and N_c is the number of classes.

Once the rules are generated, they are pruned and ordered. The pruning process involves removing those rules which did not meet the minimum confidence level, and also removing those rules which were “redundant”: rules which matched another rule body, or subset of another rule body, and had less confidence than the matched rule. The remaining rules were then ordered according to confidence and support, whereby confidence was ranked highest, followed by support and finally the number of terms in the rule body.

Finally, the class association rules are applied for prediction. The two methods that were applied and evaluated in this study are single rule (“ordered decision list”) prediction and multiple rule (“majority voting”) prediction. In the ordered decision list method, a document is assigned to the class associated with the first rule that it matches. In the majority voting method, all rules that a document matches are retained, and the document is assigned to the class that is associated with the majority of those rules.

The results of the researchers’ experiments were evaluated as in equation (2.11):

$$Accuracy = \frac{TrueC}{Total} \qquad (\ 2.11\ )$$

where *TrueC* represents the number of correctly-classified documents and *Total* represents the total number of documents in the test. Their results are shown Table 2.15.

Minimum Support = 10%			**Accuracy**		**Testing Time (m)**	
Confidence	***NO. of Rules***	***Training Time (m)***	***Majority Voting***	***Ordered Decision List***	***Majority Voting***	***Ordered Decision List***
50%	95	3.26	0.771	0.787	1.39	1.22
70%	82	3.19	0.803	0.792	1.27	1.38
80%	50	3.17	0.846	0.812	1.32	1.35

Table 2.15. Results of Associative Classifier for the Arabic text data set.

From the results, we can see that at the lowest confidence level, the ordered decision list method was seen to outperform majority voting in terms of both testing time and accuracy. However, as the confidence level increased, the majority voting method proved superior on both measures. A similar pattern is seen in a second experiment, where the minimum support level was decreased to 5%. At minimum support level = 5%, however, ordered decision list outperformed majority voting at the lowest confidence level (50%) only in terms of accuracy; in terms of testing time, it only outperformed majority voting at the 80% confidence level. Table 2.16 shows the results of their second experiment.

Minimum Support = 10%			**Accuracy**		**Testing Time (m)**	
Confidence	***NO. of Rules***	***Training Time (m)***	***Majority Voting***	***Ordered Decision List***	***Majority Voting***	***Ordered Decision List***
50%	633	3.54	0.719	0.750	1.40	1.43
70%	359	3.50	0.801	0.733	1.22	1.38
80%	345	4.00	0.797	0.749	1.35	1.32

Table 2.16. Results of Associative Classifier with Large Number of Classification Rules.

Another finding of this study was that, in the case of both methods, accuracy increased when rule confidence increased but also when there were fewer rules in the rule set. The researchers noticed that when the minimum support level was decreased to 5% in the second experiment, the number of rules in the result set increased by a factor of about 6. However, the accuracy of the results decreased for both methods, at all three confidence levels.

2.3 Summary

In this chapter, we reviewed various significant studies and experiments which have been conducted to develop and improve Arabic Information Retrieval and Data Mining approaches. We focused on the work done to develop and improve Arabic stemming and root extraction algorithms, and feature selection and document classification methods, which is the most relevant to our work.

Looking at the work done on Arabic stemming and root extraction, a number of downsides can be noticed. For example, there is no global testing database to be used when experimenting with Arabic text. Also, many researches measure the accuracy of the system without manual checking of the roots, which could not be related to the original stemmed word [10] [32]. For example, Khoja's stemmer, which is one of the strongest stemmers, different cases of Arabic tri-literal words that are weak, hamzated, geminated or eliminated-long-vowel. But the algorithm has a number of weaknesses. Firstly, the word *munaddamat* - منظمات (organizations) is stemmed to the root *dama-aa* - ظمأ (he became thirsty) instead of the correct root *nadama* - نظم. Another issue is that when the word is deducted to a tri-literal word, the weak letter is deleted in the first place, and then the last letter is doubled, or another weak letter or an *alif* is added to the word. That leads to extracting a root that is of another word, which is not related to the word. For instance, the extracted root of the word *riwayat* - روايات is *rayaya* - ريي, where the correct root is *rawaya* - روي. As well, the extracted root for the word *akhar* - آخر is *kharara* - خرر, where the correct root is *akhara* - أخر. Nevertheless, the extracted root for a word is considered correct as long as it appears it the roots lists, even if it was not related to the original word.

Having reviewed the most significant work on Arabic documents classification, we noted that the classification algorithm is always applied after light stemming text while identifying the roots as features, which would improve the classification process, was not considered.

In this work, we introduce and implement an improved root extraction algorithm for Arabic words, which is based on morphological analysis and linguistic constraints. After that, we focus on implementing the most significant classification methods while utilizing our root extractor in indexing and identifying documents features within the classification process.

Chapter 3

Arabic Language Characteristics

Arabic language complexity including its orthography and morphology made it challenging to find a standard Arabic text mining algorithms and tools. In the following chapter, these challenges are elaborated on and explained further with associated examples where possible.

3.1 General Characteristics

Orthography

Orthography in Arabic is less ambiguous and more phonetic with the use of diacritics. For example, a word can be written using the same characters and be pronounced differently. Diacritics in Arabic language contain short vowel marks, known as *harakat* - حركات, and other vowels and constant diacritics [33] [34]. They are used mainly to provide a phonetic aid to show the correct pronunciation. Arabic short vowel marks include but are not limited to, *Fat-hah* - فتحة, *Kasrah* - كسرة, *Dammah* - ضمة, and *Sukun* - سكون. Other diacritics consist of *Tanween* - تنوين (Nunation), which represents a long vowel, and *Shaddah* - شدة (Gemination), which represents a constant gemination mark. The definition of each of these diacritics is shown below.

- Fat-hah ◌َ : The *fat-ha* - فتحة, which literally means opening, is a small diagonal line positioned on top of a letter representing a short /**a**/. The mark is referred to as fat'ha, because the pronunciation of any letter with it requires opening the mouth. For example, the word كَتَبَ here is a three letters word pronounced as *kataba* (he wrote).
- Kasrah ◌ِ : The *kasrah* - كسرة, which literally means breaking, is as well a small diagonal line, yet placed below a letter referring to a short /**i**/. For example, the word مِن is pronounced *min* (from).
- Dammah ◌ُ : The *dammah* - ضمّة, is a short vowel mark that looks like a small curl and is positioned above a letter to indicate a short /**u**/ or /**o**/ as in the English world 'to'. For example, the word كُتُب here is pronounced as *kutub* (books), noting that it is written with the same letters as in the first example.
- Sukoon ◌ْ : The *sukoon* - سُكون, is a small diacritic looking like a small circle positioned on top of a letter to indicate that the consonant it is placed on is not followed by a vowel. For example, the word مَكْتَبَة is pronounced *ma**k**tabah* (library).
- Tanween ◌ً, ◌ٍ and ◌ٌ : The *tanween* - تنوين , which is also known as nunation, is used to show that the consonant is followed by an 'n', by doubling one of the three short vowels (*fat-hah, kasrah* or *dammah*) at the end of the word, as it can only be added to the last letter. The diacritics from left to right refer to /**an**/, /**in**/, and /**un**/ or /**un**/. They are mostly used in classical or literary Arabic to indicate non-pausal grammatical indefinite case endings.
- Shaddah ◌ّ : The *shaddah* - شدّة , is a diacritic that looks like a small Latin 'w' and is positioned on top of the letter to represent gemination. That is doubling or adding an extra length to the letter it is placed above, which is considered phonemic in the Arabic language. *The shaddah* is the only diacritic which is

usually used in ordinary spelling to avoid uncertainty. This can be seen given the words *madrasah* - مدرسة (school), and the word *mudarrisah* - مدرّسة (female teacher).

- Hamzah إ , أ , ؤ , ئ and ء: The *hamzah* is a glottal stop which is not considered a short vowel. It could appear as a letter by itself ‘ء’ or on top of the letters *waw* - ‘ؤ’ or *yaa* - ‘ئ’, or above or under an *alif* - ‘أ , إ’. In most cases that depends on the short vowel (*fat-hah*, *kasrah*, *dammah* or *sukoon*) attached to the previous letter of it. That can be seen in the words, رَأس (head) pronounced *ra’s*, سُؤال (question) pronounced *su’aal*, مِئة (hundred) pronounced *mi’ah*, and شيْء (thing) pronounced *shay’*.

The pronunciations of these diacritics are represented in Table 3.1 using the letter ب. However, in Modem Standard Arabic (MSA), diacritics marks are not usually included in printed and electronic text, and the understanding and correct pronunciation of the word is determined within its context by the reader.

Double Constant / Gemination	**Double Vowel / Nunation**			**Short Vowels**			
	Kasrah	**Dammah**	**Fat’hah**	**Kasrah**	**Dammah**	**Fat’hah**	**Sukun**
بّ /bb/	بٍ /bin/	بٌ /bun/	بً /ban/	بِ /bi/	بُ /bu/	بَ /ba/	بْ /b/

Table 3.1. Arabic Diacritics

Word Meanings

One word could have several meanings depending on its position and context, despite it having the same pronunciation. For example, as shown in Table 3.2, the Arabic word *qalib* - قلب could have three meanings or more as a noun.

Word Meaning	**Sentence**
core	في **قلب** الأحداث
heart	أجرى عملية **قلب** مفتوح
center, middle	الكرة في **قلب** الملعب

Table 3.2. Meanings of the Word (قلب)

Variations of Lexical Category

One word can belong to more than one lexical category depending on its meaning and context. Lexical categories include nouns, verbs, adjectives and more. In Table 3.3 an example is given of the word *ain* - عين, belonging to different lexical categories depending on its meaning [3].

Word Meaning	**Word Category**	**Sentence**
Ain	Proper-Noun	**عين** جالوت
wellspring	Noun	**عين** الماء
eye	Noun	**عين** الإنسان
delimitate/ be delimitate	Verb/Passive Verb	**عُيِّن** وزيراً للخارجية

Table 3.3. Lexical Categories of the Word (عين)

Dual Root

Some words can be formed from more than one root such as the word *riyadh* - رياض, which is derived from the roots *radha* - راض, and - *rayadha* ريض [13].

The Exchange Process

The exchange process *al-ebdal* - الإبدال, depends on a phonetic rule instead of a syntactic one. For example, the word *qiyam* - قيام of the pattern *fiaal* - فعال, is derived from the root *qama* - قام. However, from the consonants in the pattern, the extracted root would be *qima* - قيم, where the letter 'ي' should be exchanged to 'ا'. The exchange process mostly occurs with the vowel letters ا, و and ي. The cases of exchanged vowel letters are taken in account in the proposed root extraction algorithm. That is explained in the second phase of the algorithm in section 4.1 of chapter 4. Nevertheless, it can also arise with other letters as in the word *alsirat* - السراط, where the letter *sin* - س is changed to the letter *ṣād* - ص to be written as *alṣirat* - الصراط.

Deleted Letters of Words

In some cases, a letter in the pattern of the word is deleted affecting the process of root extraction, like in the word *ra'aa* - رأى of the pattern *fa'al* - فعل. The present tense of the word should follow the pattern *yaf'al* - يفعل becoming *yar'aa* - يرأى . Instead, the letter *alif* - أ is deleted, and the word becomes *yara* - يرى for morphological reasons, so the letter *ain* - ع of the pattern is deleted, becoming *yafl* - يفل instead of *yaf'al* - يفعل [35].

Ambiguous Words

Some words in Arabic starts with the letter *waw* – 'و' or the letter *baa* – 'ب'. These words are ambiguous when it comes to Arabic data mining. That is because these letters can be part of the original root like in the words *wojood* - وجود and *bohooth* - بحوث, while in other words they maybe a prefix like in the words *bisu'aalihum* - بسؤالهم and *wa'awamirhum* - وأمرهم. This case is handled in the proposed root extraction algorithm through the pattern checking and suffix/prefix removal phase, which is explained in details in section 4.1 in the fourth chapter of the thesis.

Morphological Characteristics

Arabic language has a very complex morphology when compared to English [36] [37]. Words in Arabic can be formed of a stem alongside affixes and clitics. The stem is composed of a consonant root (جذر صحيح) and a pattern morpheme. The affixes consists of inflectional markers which determines the gender, tense or/and number, while clitics can be prepositions (حروف جر), conjunctions (حروف عطف), determiners (محددات), possessive pronouns (ضمائر ملكية) and pronouns (ضمائر).

Morphemes in Arabic are mostly identified by three consonant letters which for the root of the word, as well as several affixes which could be added to the root to form a word. For instance, given the root *kataba* - كتب, which is the root of the noun *kitabah* - كتابة (writing), we could inflect various number of words related to the concept of writing. That include the words *kataba* - كَتَبَ (he wrote), *kitab* - كتاب

(book), *kutuk* - كُتُب (books), *yaktub* - يكتب (he writes), *katib* - كاتب (writer), *maktabah* - مكتبة (library) and more.

In addition, the translation of one Arabic word in English can sometimes be composed of a number of words in English. For example, the Arabic word *wabitatheeriha* - وبتأثيرها means (and by her influence). Therefore, segmentation of Arabic textual data is more difficult than it is in Latin languages.

Also, one root can be used to form several words that have different meanings which are not exactly similar to each other. For instance, the root *alama* - علم can form several words of different meanings when adding affixes as shown in Table 3.4. Another example of morphological variation is of the word *thahaba* - ذهب of the verb (to go), where different affixes are added in different tenses depending on the gender and number of subjects is shown in Table 3.5. The morphological structure of the words and the addition of prefixes, infixes and suffixes, are considered in the root extraction algorithm proposed in this thesis. This is explained in details in section 4.1 in the fourth chapter of this thesis.

Meaning	Suffix	Infix	Prefix	Word
scientific	ية	-	-	علمية
taught us	تنا	-	-	علمتنا
his science	ه	-	-	علمه
scientists	اء	-	-	علماء
scientist	-	ا	-	عالم
teaching	-	ي	ت	تعليم
teacher	-	-	م	معلم
sciences	-	و	-	علوم
informative	ية	ا	است	استعلامية
information	ات	و	م	معلومات

Table 3.4. Variations and Meanings of the Word (علم).

Time	Gender of Subject/s	# of Participants	Verb
Past	Male	1	ذهب
Past	Female	1	ذهبت
Past	Male	2	ذهبا
Past	Female	2	ذهبتا
Past	Male	3 or more	ذهبوا
Past	Female	3 or more	ذهبن
Present	Male	1	يذهب
Present	Female	1	تذهب
Future	Male	1	سيذهب
Future	Female	1	ستذهب
Future	Male	3 or more	سيذهبوا
Future	Female	3 or more	سيذهبن

Table 3.5. Morphological Variations of the Word (ذهب).

3.2 Arabic Morphology

3.2.1 Definition of Morphology

Morphology or derivation in general means to take a part of an object or half of it, and to derivate a word, is to change it by adding or omitting letters from the original word [38]. Whereas in linguistics it has been given various definitions which are not too far from the general definition, we choose among them the following [39]:

a- Deducting a branch from the original word containing the letters of the original word.

b- Forming a word from another one by amending the order of the letters or adding others to them.

c- Creating a new utterance from another one suiting each other in meaning and structure while they are different in the form.

3.2.2 Conditions of Derivation

Arabic grammarians have set a number of conditions that should be fulfilled to accept the derivation process as follow [40]:

a- The derived word, either a noun or a verb, should have an original utterance, since the derivate word is a branch that has been taken from an original word as we have learned from the definitions above. Thus, if the word is an original one it won't be considered as a derivate. But the question here is what the original word of the new one is? Is it a gerund or a verb? Since there is a disagreement among the Arabic grammarians as we will see later.

b- The derivate should be adequate with the source in the used letters with regards the number and order where they agree with the order used in the source.

c- The meaning of the derived word should suite the original source in containing the same meaning with slight deference in the way it exists. For instance, by clarifying the number of times, gender or tense, such as *dharaba* - ضَرَب (he beat), *dharb* - ضَرْب (beating) and *dharib* - ضَارِب (beater). Since, the Kofians grammarians mentioned that it can be the same letters but different vocalization as in *dharaba* - ضَرَب and *dharb* - ضَرْب above.

3.2.3 Origins of Morphology

The origin of the word in Arabic which morphology relies on to drive the new words have raised a controversial great and interesting debate among various grammarians of the most famous schools of Arabic grammar , Kufans and Basrians. That is where the Basrians see that the verb is derived from the source which is the gerund (المصدر) since they have provided the following proofs [41]:

a - The gerund indicates an absolute and open period of time, while the "verb" shows a specific tense of time such as past or present etc. So the absolute case should be the origin of the restricted one.

b- The gerund shares its origin letters in all tenses and doesn't' share some, and excludes other tenses, and when they need to indicate a specific time they derive the verb and adverb all together from the gerund.

c- The gerund is a noun (verbal noun) which a word that can stands and understood by itself with no need to a verb, while the verb doesn't work separately.

d- The verb indicates the event and the time while the gerund indicates only the action only.

e- The verb indicates the meaning of the gerund, whereas the gerund does not indicate the meaning of the verb such as *nasara* - نصَرَ, which means (lead to victory), and carries the meaning of *nasr* - نَصْر meaning (victory), since the verb is a branch that should have an origin.

f- The gerund has one example while the verb has more than one example.

g- If the gerund has been derived from the verb it would have a fixed pattern similar to the verb instead of having deferent patterns for nouns such as the noun of the subject or the passive participle.

On the other hand, the Kufans have provided their point of view to support their position of considering the verb as the origin of the derived words in Arabic as follow:

a- The gerund is derived from the verb and it is a branch of it as in *kataba*: *kitaabatan* - كتب: كتابةً (to write: writing), *qama* :*qiyaaman* - قامَ قياماً : (to stand , standing), *ista'lama: isti'laaman* - استعلم: استعلاماً (to inquire: inquiry), and *iftataha: iftitaahan* - افتتح: افتتاحاً (to open: opening).

b- The gerund follows the verb in is vocalization which means it becomes defective if the verb is defective and vice versa, such as *kharaja: khuroujan* - خرج: خروجاً (to go out: going out) and *qaama: qiyaaman* - قامَ قياماً : (to stand: standing).

c- The verb controls and affects the gerund and not the opposite, such as *darabtu darban* -ضربتُ ضرباً (I beat hardly) or *ustuqbila al mad'owwna istiqbalan haarran* -أُسْتُقْبِلَ المدعوون استقبالاً حاراً (the guests have been welcomed very warmly). So it means that the verb governs the gerund, and is higher in strength than it is as an original word.

d- The gerund is used to strength the act (the verb) and gives it an emphatic situation, such as *tafattahat alworood tafattuhan* - تفتحت الورود تفتحاً (the roses opened widely) and *inshaqqat alardhu inshiqaaqan* - انشقت الارض انشقاقاً (the earth have cracked hugely). So, it is clear that the verb exist before the gerund in the previous examples, showing that it is the origin of the words while the gerund is a branch.

e- In Arabic there are a number of verbs that do not have a gerund (مصدر), such as *ni'ma* - نِعمَ (what a good or how good), *bi`sa* - بئسَ (what a bad or how bad), *'asaa* - عسى (may be or perhaps), *laysa* - ليسَ (not) and the verbs of interjection (التعجب), *ma af'alu*: *ma ajmalu assamaa`* - ما أجملُ السّماء (what a beautiful sky). So they stated that the verb can exist without the gerund which means it is the original.

Nevertheless, linguist Ibn Al-Anbaari argues in his book 'Alinsaaf fi Masaa`il Alkhilaaf' regecting the Kufans argument, explaining that the Kufans are not correct in the last three points because of the following reasons:

These evidences are rejected since we say in Arabic, as an example, *jaa` Zaidun Zaidun* جاء- زيدٌ زيدٌ (Zaid came himself*), raa`ytu Zaidan Zaidan* - رأيتُ زيداً زيداً (I saw Zaid himself), and *marartu be Zaidin Zaidin* - مررتُ بزيداً زيداً (I passed by Zaid himself). Ibn Al-Anbaari states that the second noun confirms and emphasizes the first one in these three sentences whereas it wasn't derived from the first, and is not a branch either.

Also, the derivate can be used even if we do not use the source, while the source will remain a source and the derivate remain as it is. It is said: *tayrun 'abaabeel* أبابيل - طيرٌ (separate birds), where the Arab used the plural without using a singular word for birds, as *tayr* - طير (bird). They also said *tayrun abaabeel* which means in groups, and never used a plural for birds, as *toyur* - طيور (birds). Since, some grammarians claim that it doesn’t have singular. In addition there are some gerunds that do not have verbs such as *wailahu* - ويله *or waihahu* - ويحه (woe unto him).

Briefly, the contemporary grammarians tend to accept the standards of the Kufans relaying on the studies of the comparative linguistics. That is according to what they have extracted from the out coming theories and knowledge about various types of languages. Also, with regards to that position, Wil Vincent stated in his book ‘The History of the Semitic Languages’:

“The majority of the words have been derived from an original word of three letters which is a verb added with one or two letters or more at the beginning (prefix) or at the end (suffix) where one word can provide deferent images indicating different meanings”.

Finally, a number of contemporary researchers see that derivation does not have one original source. As well, the Arab have derived words from nouns, verbs, particles as well as prepositions such as *‘ala: ‘alaa* - علا : على (on: to go high) and *‘an: ‘an’ana* عنعن :عن - (about: to till chain of narrators’ of prophet statements). However, since it looks difficult to find absolute evidence that can tell which is the right school, it was decided to adopt the indications that show that the verb is the source of the majority of words used in Arabic text.

3.2.4 The Use and Purpose of Morphology

There are several purposes and uses of morphology/derivation [42]. Below is a list of the main benefits and uses of morphology [38]:

a- This linguistic art of morphology have expanded the list of Arabic vocabularies. For example, we could see that one of the most popular Arabic- Arabic dictionaries, Lisaan Alarab (لسان العرب) by Ibn Manzuur, includes more than 7,500,000 words. With the use of morphology/derivation it was less difficult to form adjectives, verbs, adverbs, name of places and many derived words. Also, derivation and Arabic dictionaries such as Lisaan Alarab, have helped poets to have more control on the rhythms of their poems. They were able to develop their skills in speech, which led to enrich their text with rhymed prose and poetry.

b- By learning and understanding derivation, grammarians and linguists were enabled to recognize the additional letters and the original words and their meanings.

c- It helped to determine the purification of the word being originally Arabic, and it is a way to distinguish between the original pure word and the foreign one, where we cannot find an origin source to the latter one either in utterance or semantically. For example, when we look to words such as *assiraat* - الصراط (the way), or *alfirdaws* - الفردوس (the paradise), and other Arabicized words, we will notice that they don't have an origin in Arabic. That means that there are no sources for the words *assiraatt* and *alfirdaws*, since the presence of a root or origin indicates the Arabic origin of the word.

3.3 Words Derivation Methods

Arabic linguists and grammarians have categorised derivation to four different types according to the number of root letters and the way of deriving new words and their meanings. These are listed below as the following:

3.3.1 Minimal Derivation (الإشتقاق الأصغر)

Minimal, small and general are different names given to this type, which focuses on deriving a word from an original one where both should agree in meaning [43], the original letters and their order. For example, the word *dhaarib* - ضارب (beater), and the passive participle *madhroob* - مضروب (beaten), are both derived from the source, *dharb* - ضرْب (beating), according to Basrians. It was said it is derived from the root verb *dharaba* - ضرَب (to beat), according to the Kufans.

According to Ibn jinny in his book Alkhasa’is [44], it is that type used by the majority of people and consists of one idea keeping the same order. On the other hand there is also a disagreement among grammarians, where some of them believe that some words are derived. However some others do not, such as Sybawayeh, Abu Obaidah, Al Asma’I, Abu Omar and Alkisa’I [assuyoti, Almuzhir - part one]. On top, a third group of grammarians believe that all Arabic words are derived from a root.

This type of derivation is the most common one used in Arabic and it is the most important too. So when the word: *ishtiqaq* - إشتقاق (derivation) is mentioned, it represents this type of derivation and doesn’t mean any other type except with some restrictions. Nevertheless, both scholars of Syntax and linguistics have approved of minimal derivation. But the first have treated it by looking to the form of and shapes of the words as a result of derivation, while the linguists are looking to the relation between the two words, how much they suite each other and share in meaning, and the number of letters they are formed of, without looking to the vocalization or constant cases. Last but not least, scholars gave different opinions about the size of circle that includes the words derivation process in this type.

3.3.2 Supreme Derivation (الإشتقاق الكبير)

In this type of derivation, the letters of the root are disordered. According to Ibn Jinni, in his book Alkhasa’is, this type is where you choose a three letters root word and form six words of those letters by changing their order in a word, having a related meaning among them [45]. On top, if there is any different in that meaning we choose the impeaded meaning as the derivation specialists do with a specific structure. As well, Ibn Jinni was the only one who believes that this type of derivation is of “the supreme derivation” type. The following is an example given by him:

The root letters of the word *jabara* - جبر , which refers to strength and protections and as well to heal a broken bone, can be reordered and used as *jaraba* - جرب , which is to put something in a pocket to protect it as a bag for money or put socks on. Also, from roots that are built using these three letters, different words can be formed. Below are few examples of that:

- Rajulun mujarrib - رجل مجرب, meaning an experienced man whom his experiences in life have strengthened him by knowledge and attitude to improve his life.
- Juraab - جراب, is a sock, in which protects the feet from cold and harms.
- Burj - برج, is a tower which is a strong building and is used for protection against enemies.
- Rajab - رجب , which refers to strength or to glorification and exalt.

So, we notice that Ibn Jinni have circulated the three root letters forming six deferent word order but still go around the same meaning (*jabara* - جبر, *rajaba* - رجب, *jaraba* - جرب, *baraja* - برج). However, this theory was hard to prove in some cases, as Ibn Jinni found it difficult to generalize when he tried applying it with the words of four letters root or more, therefore, he restricted it with the words of three letters root.

In addition, this idea of circulating was treated earlier by Alkhalil bin Ahmad in his book Al'ayn [46], where he registered all used Arabic words by circulating the letters order in all possible ways, showing which word is used and which is not. Alkhalil have also been followed by Ibn Duraid and others, but he did not see that all the six different forms of roots of specific three letters should lay under the same root meaning, whatever their location or order are.

On the other hand, Ibn Jinni's teacher Aba Ali Al-farsi have approved of this idea, despite the fact that it was clear that Ibn Jinni was not very successful in his theory to be generalized for quad-literal words. As well, it was difficult in some cases to find the meaning relations between words which are formed of the same three letters.

3.3.3 Grand Derivation (الإشتقاق الأكبر)

This type of derivation also called linguistics replacement *alibdaal allughawi* - الإبدال اللغوي. It is defined as the one where most of the letters in two words are united and related such as in, nahaqa - نهق (to bray) and *na'aqa* - نعق (to croak) [47]. The grand derivation is different from the phonetical one *alibdaal assarfi* - الإبدال الصرفي, since the later exists for a phonetic reason by replacing a phonetical letter in a word by another sound or letter when the letters appear close in pronunciation. An example of that is seen in the words *Saama* - صام (to fast) from the root *sawma* - صوم (fasting).

The grand derivation is considered broad according to the size of letters, where it includes new letters which did not appear in the original word. However, some linguists believe that it could include all the alphabets, while others restricted them to those close to each other in exit and sound. Some scholars such as Ibn Jinni and Assuyouti have looked at this type of derivation as a special one while others have rejected the idea, arguing that it is against the nature of derivation. Those scholars considered this type as a phonetical phenomenon built on replacing letters instead of other ones, because of reasons such as:

- Mishearing the words which led to different repentance.
- Phonetical development of the replaced letter.
- Misreading or mispronunciation.

More examples for this type of derivation include: *sahala* - صهل (to neigh), *za'ra* -زأر (to roar), and *sa'la* - سعل (to cough). So these three verbs indicate sounds representing the sound of the horse, the lion and human being. When we compare these forms with each other, we find that the first letters of the three verbs, *sad* - ص, *zay* - ز and *sin* - س , are hissing or whistling letters, while the middle letters, *haa'* - هـ, *hamzah* - ء and *'ayn* - ع, are guttural letters, noticing that the *lam* - ل is shared at the end of two verbs while it is a *raa'* - ر in the other, and these are the letters of derivation.

3.3.4 The Giant Derivation (الإشتقاق الكُبَّار)

This type of derivation is also called *annaht* - النحت (the carving, here it means to carve or coin a word) [48]. The giant derivation has been named by Abdullah Ameen back in 1956 in his book Alishtiqaaq for what is called *annaht* - النحت [47]. In which the derivation is done by taking some letters from two or more words or from a sentence, and utilizes them to create a word that suite them in meaning and utterance. Arab has used this process to abbreviate complex structures such as the following:

- Basmala - بسمل , which is to say *bismillah* - بسم الله (in the name of Allah).
- Sabhala - سبحل: to say *subhaan Alla* - سبحان الله (glory be to Allah).
- Hay'ala - حيعل: to say *hayya 'ala alfalah* - حي على الفلاح (come for success).
- Hawqala - حوقل: to say *la hawla wala qowwata illa billaah* - لا حول ولا قوة إلا بالله (no power and no capability but with Allah).
- Hallala - هلل: to say *la ilaha illa Allah* - لا إله الا الله (no god but Allah).
- Kabbara - كبر: to say *Allahu akbar* - الله أكبر (Allah is the greatest).
- Istarja'a - استرجع: to say *inna lillahi wa inna ilayhi raji'oun* - إنا لله وإنا إليه راجعون (we belong to Allah and we will return to him).
- Jawqala - جوقل: to say *hamal jawwan* - حمل جواً (to carry things through air).

Also, this type is applied to create words for abbreviation to relate people or nouns to their origin or their tripes. For example, the word *abshami* - عبشمي is used to when referring to someone or something that belongs to the tripe of Abdu Shams - عبد شمس. However, linguist Ibn Faris said that most of the four and five letters root words are derived through this type of derivation, while only some of them were original words or were derived through other types [49].

This type of derivation is considered as a way to generate new words to indicate new meanings, where it was approved contemporary by Arabic complexes in necessity. An example of that can be seen in the word *barmaa'I* - برمائي

(amphibious), derived from the words *bar* - بر and *maa* - ماء (land and water). As well, that process is applied in generating abbreviation names of companies, medical structures or military expressions.

3.4 Roots and Derivation Patterns

Derivation is used to derive words that are verbs, including tenses like past, present and imperative in different pattern forms, which are called *awzaan* - أوزان in Arabic. As well, it is used to derive words that are nouns, such as names of places, tools, people, places, gerunds, adjectives and adverbs, including numbers and genders of those nouns in addition to relations and abbreviations.

In the following, we provide examples of most famous derivation patterns used in Arabic for both verbs and nouns in tables. That includes showing how they are organized in a helpful way to understand the language for learners and researchers, as well as those who are interested in Arabic literature and Islamic studies, or working on developing the language threw modern technology like computing and software engineering.

3.4.1 Verb Patterns

Verbs in Arabic language have ten basic pattern forms, according to linguist Mahmoud Al-batal [50]. In the most commonly used Arabic-English dictionaries, these are numbered I-X with roman numerals by convention. In Arabic literature tradition, by contrast, those are not numbered. They are referred to by *wazin* - وزن (pattern), with a representative of the three letters root constants, *fa-aa-la* - فَعَلَ. These three letters act as symbols that stand for three letters that make up a root. The letter *faa* - ف pronounced *fa* - فَ in this form, represents the first letter of the root. The letter *ain* - pronounced *aa*, represents the second letter, while the letter *lam* - pronounced *la*, represents the third letter of the root.

Pattern one I in Table 3.6 below, is considered to be the basic form that gives the core meaning from which others are derived. The other patterns build upon that basic meaning, each in a particular way, for example by making it transitive or passive. Table 3.6 below shows a list of the main verbs patterns sorted depending on the tense of the verb. Verb patterns are included in the implementation of the proposed root extraction algorithm, and is explained in details in chapter 4 of the thesis. However, not all verbs can be derived to all ten patterns, as for some the maximum derivation can be nine only, as can be seen in the example given in Table 3.7.

Pattern	Past Tense	Present Tense	Gerund
I	فَعَلَ/فَعِلَ/فَعُلَ	يَفْعَلُ/يَفْعُلُ/يَفْعِل	varies
II	فَعَّلَ	يُفَعِّلُ	تَفْعِيل
III	فَاعَلَ	يُفاعِل	مُفَاعَلَة
IV	أفْعَلَ	يُفْعِلُ	إفْعَال
V	تَفَعَّلَ	يَتَفَعَّلُ	تَفَعُّل
VI	تَفَاعَلَ	يَتَفاعَل	تَفَاعُل
VII	انْفَعَلَ	يَنْفَعِل	انْفِعَال
VIII	افْتَعَلَ	يَفْتَعِلُ	افْتِعَال
IX	اِفْعَلَّ	يَفْعَلُّ	اِفْعِلال
X	اِسْتَفْعَلَ	يَسْتَفْعِلُ	اِسْتِفْعَال

Table 3.6. Verb's Basic Pattern Forms.

Pattern	Past Tense	Present Tense	Gerund	Meaning
I	قَطَعَ	يَقْطَعُ	قَطَعَ	To cut
II	قَطَّعَ	يُقَطِّع	قَطَّعَ	To chop up
III	قَاطَعَ	يُقَاطِع	قَاطَعَ	To cut of
IV	أقْطَعَ	يُقْطِع	أقْطَعَ	To divide up (land)
V	تَقَطَّع	يَتَقَطَّع	تَقَطَّع	To be chopped up
VI	تَقَاطَعَ	يَتَقَاطَع	تَقَاطَعَ	To intersect
VII	اِنْقَطَعَ	يَنْقَطِع	اِنْقَطَعَ	To be cut of
VIII	اِقْتطَعَ	يَقْتَطِع	اِقْتطَعَ	To take a cut of
IX	-	-	-	-
X	اِسْتَقْطَعَ	يَسْتَقْطِع	اِسْتَقْطَعَ	To deduct

Table 3.7. Basic Pattern Forms of the Verb (قطع).

In previous examples in Table 3.6 and Table 3.7, patterns are derived from the three letters root in the past and present tenses, considering the subject of the third person being singular and masculine. In addition to these patterns, suffixes and prefixes could change, or more of them must be added differently. That depends on the person or object and its gender and number. In Table 3.8, more examples are shown adding suffixes and prefixes depending on the person, number of subjects and gender, in the past and present tenses with examples of their pronouns as well.

Present Tense	**Meaning**	**Past Tense**	**Meaning**	**Pattern Form**	**No. of Subjects**	**Person**	**Gender**	**Pronoun**	**Meaning**
أكتبُ	I write	كتبتُ	I wrote	فَعلتُ	1	1st	male & female	أنا	I am
نكتبُ	We write	كتبْنا	We wrote	فعلنا	Plural (2+)	1st	male & female	نحنُ	We are
تكتبُ	You write	كتبْتَ	You wrote	فعَلتَ	1	2nd	male	أنتَ	You are
تكتبِين	You write	كتبْتِ	You wrote	فعَلتِ	1	2nd	female	أنْتِ	You are
تَكْتُبَان	You write	كتَبْتُمَا	You wrote	فعَلْتُما	Two	2nd	male/female/both	أنتُمَا	You are
تكتُبُون	You write	كتبْتُم	You wrote	فعلْتُم	Plural (3+)	2nd	male	أنتم	You are
تكتُبْنَ	You write	كتَبْتُنَّ	You wrote	فعلْتُنَّ	Plural (3+)	2nd	female	أنتُنَّ	You are
يكتُبُ	He writes	كتَبَ	He wrote	فعَلَ	1	3rd	male	هُو	He
تكْتُبُ	She writes	كتَبَتْ	She wrote	فَعَلَتْ	1	3rd	female	هِى	She
يكتُبَانِ	They write	كتَبَا	They wrote	فعَلا	Two	3rd	male	هُما	They
تكْتُبَانِ	They write	كتَبتَا	They wrote	فعَلَتَا	Two	3rd	female	هُما	They
يكتُبونَ	They write	كتبُوا	They wrote	فعَلوا	Plural (3+)	3rd	male	هُم	They
يكتبن	They write	كتبن	They wrote	فعلن	Plural (3+)	3rd	female	هن	They

Table 3.8. Patterns of the Verb (كتب) Considering the Person, Number and Gender.

3.4.2 *Noun Patterns*

Table 3.9 shows different patterns of noun derivation from the root, depending on various letters order, and regarding positions of long vowels or root vocalizations.

Plural of the place	Place	Source	Passive participle	Active participle	Root	Meaning
مَكَاتِب makaatib	مَكْتَب maktab	كِتَابَة kitaabah	مَكْتُوب maktuob	كاتِب kaatib	كَتَبَ kataba	To write
مَسَاكِن masaakin	مَسْكَن maskan	سَكَن sakan	مَسْكُون maskuon	سَاكِنَ saakin	سَكَنَ sakana	To live
مَشَارِب mashaarib	مَشْرَب mashrab	شُرْب shurb	مَشْرُوب mashroub	شَارِب shaarib	شَرَبَ sharaba	To drink
مَدَاخِل madaakhil	مَدْخَل madkhal	دُخُول dukhoul	مَدْخُول madkhoul	دَاخِل daakhil	دَخَلَ dakhala	To enter
مَفَاتِح mafateh	مَفْتَح maftah	فَتْح fateh	مَفْتُوح maftouh	فَاتِح faatih	فَتَحَ fataha	To open
مَقَاطِع maqaati’	مَقْطَع maqta’	قَطْع qate’	مَقْطُوع maqtou’	قَاطِع qaati’	قَطَعَ qata’a	To cut
مَنَامَات manamaat	مَنَامَة manaamah	نَوْم nawm	مُنَام munaam	نائِم naa’im	نَامَ naama	To sleep
مَآكِل ma’akil	مَأكَل ma’kal	أَكْل akel	مأكُول ma’koul	آكِل aakil	أَكَلَ akala‘	To eat
مَهَادِي mahaadi	مَهْدَى mahda	هَدْي hady	مَهْدِي mahdi	هَادِي haadi	هَدَى hadaa	To guide
مَدَارِس madaaris	مَدْرَسَة madrasah	دِرَاسَة diraasah	مَدْرُوس madrous	دَارِس daaris	دَرَسَ darasa	To study
مُكَبِّرات mukabbiraat	مُكَبِّر mukabbir	تَكْبِير takbiir	مُكَبَّر mukabbar	كَابِر kaabir	كَبُرَ kabura	To be big

Table 3.9. Noun Patterns Examples with Different Letters Order.

In addition to noun patterns of singular subjects, more nouns patterns are used for plural. Most common patterns of those are displayed in Table 3.10. Both noun patterns, singular and plural, are employed in the implementation of the introduced root extraction algorithm alongside the verb patterns, which is explained further in chapter 4 of the thesis.

Singular Pattern	Example	Plural Pattern	Example	meaning
فَعْل fa’l	لَوْن lawn	أَفْعَال af’aal	أَلْوَان alwaan	colour/colours
فَعَل fa’al	وَلَد walad	أَفْعَال af’aal	أَوْلَاد awlaad	boy/boys
فَعَال fa’aal	مَتَاع mataa’	أَفْعِلَة af’ilah	أمتعة amti’ah	luggage/luggage
فـَعْل fa’l	شَهْر shahr	أَفْعُل af’ul	أَشْهُر ashhur	month/months
فُعْلَة fu’lah	قُبْلَة qublah	فُعَل fu’al	قُبَل qubal	kiss/kisses
فَعَل fa’al	جَبَل jabal	فِعَال fi’aal	جِبَال jibaal	mountain/ mountains
فَعْل fa’l	بَيْت bayt	فُعُول fu’oul	بيوت boyout	house/houses
فَعَال fa’aal	غَزَال ghazaal	فِعْلان fi’laan	غِزْلان ghizlan	deer/deers
فَعِل fa’il	تَقِي taqi	أَفْعِلاء af’ilaa’	أَتْقِيَاء atqiyaa’	racious/raciouses
فَعِيل fa’iil	أَمِير amiir	فُعَلاء fu’alaa’	أُمَرَاء umaraa’	prince/princes
مَفْعَل maf’al	مَعْبَد ma’bad	مَفَاعِل mafa’il	مَعَابِد ma’aabid	temple/tembles
مِفْعَال mif’aal	مِفْتَاح miftaah	مَفَاعِيل mafa’iil	مَفَاتِيح mafatiih	key/keys
فَاعِل faa’il	جَانِب jaanib	فَوَاعِل fawaa’il	جَوَانِب jawaanib	side/sides
فَعِيلَة fa’iilah	قَبِيلَة qabiilah	فَعَائِل fa’aa’il	قَبَائِل qabaa’il	tribe/tribes
فَعِيلَة fa’iilah	مَدِينَة madiinah	فُعُل fu’ul	مُدُن mudun	city/cities
فِعَال fi’aal	سِوَار siwaar	أَفَاعِل afa’il	أساور asawir	bracelet/bracelets
فِنْعَال fin’aal	قِنْطَار qintaar	فَعَالِيل fa’aalil	قناطير qanaatir	kantar/kantars

Table 3.10. Plural Noun Patterns Examples.

3.5 Summary

In this chapter, the main aspects of the Arabic Language were briefly introduced. This included the general characteristics of Arabic, alongside its morphological structure and the history correlated to it. At the end, it was elaborated on derivation methods and the derivation patterns of verbs and nouns with different examples for both, noun derivation and verb derivation patterns.

Chapter 4

Roots Extraction

In this chapter a linguistic root extraction approach that is composed of three main phases is presented. In the first phase removal of affixes including prefixes, suffixes and infixes is handled. Prefixes and suffixes are removed depending on the length of the word, while checking its morphological pattern after each deduction to remove infixes. In the second and third phases, the root extraction algorithm is developed further to handle weak, hamzated, eliminated-long-vowel and two-letter geminated words as there is a rationally great amount of irregular Arabic words in texts. Before roots are extracted, they are checked against a predefined list of 3800 tri-literal and 900 quad literal roots. Series of experiments are conducted on a selected data set from Al-Sulaiti's online Arabic corpora [51]. The data set of Al-Sulaiti is gathered to give a sample text material for Arabic teachers, learners and mainly Arabic language researchers. The corpus consists of 842684 words and 14 different categories, which are processed to improve and test the performance of the proposed algorithm. The work presented in this chapter was published in [52].

4.1 Methodology

The proposed root extraction algorithm is composed of three main phases. These phases are processed after a text pre-processing stage where all stop words and vowel marks are removed. In the first phase we focus on eliminating suffixes and prefixes according to the length of the word, while employing a pattern matching process to remove infixes and extract the root of the word. The words are matched against patterns of similar length after every prefix/suffix deletion, to improve the speed of root extraction and avoid removing original letters of the word that are equal to a group of a suffix/prefix letters.

In the second phase, if the word root is still not found, it is decided to remove suffixes and prefixes that are of one letter where the word is more than three letters long. If the word is three letters long, it is then processed depending on it being hamzated, weak, geminated, or a word with eliminated long vowel. Finally, if the word is of two letters, it is processed depending on its being a geminated or a long-vowel-eliminated word. Below is a detailed explanation of the three phases of the algorithm.

4.1.1 *First Phase*

Within this phase, the algorithm is defined to process words according to their length, starting with rules for long words and moving towards shorter words. After every suffix/prefix deletion the word is checked against a list of patterns of the same length, as seen in Table 4.1. If a pattern is matched, the root is extracted and is validated by checking if it exists in a predefined root list of 3800 tri-literal and 900 quad literal roots, otherwise the word is processed through the second phase of the algorithm

Length of Patterns/Roots	Patterns
Length 4	فاعل، فعال، فعيل، فعول، أفعل، مفعل، فعلة، فعلل
Length 5 patterns of tri-literal roots	تفعيل، تفعلة، مفاعل، افعال، متفعل، تفاعل، تفوعل، انفعل، افتعل، مفتعل، فعلان، فعلال فعلاء، مفعال، فواعل، فعيلة، فعائل، أفاعل، مفعول، تفعيل، فاعول، فعلى، فعالة، تفعلة، مفعلة، أفعلة
Length 5 patterns of quad roots	تفعلل، مفعلل، فعللة، فعالل
Length 6 patterns of tri-literal roots	استفعل، مستفعل، افتعال، انفعال، متفاعل، مفاعلة، مفاعيل، أفعلاء، أفاعيل، افعوعل، مفعوعل
Length 6 patterns of quad roots	فعاليل، افعلال، افتعلل، متفعلل
Length 6 or more	استفعال، افعيعلال

Table 4.1. Arabic Patterns and Roots.

The process of this method is described in the steps below, and a corresponding flowchart of this phase is represented in

- First, if the word starts with the letters ‘ال’ then remove them.
- If the word length is equal or greater than six, check for the following prefixes and remove them:

  Prefixes- كال، بال، فال، مال، وال، ولل، است، يست، تست، مست

- If the word length is still greater than or equal to five, remove the following prefixes/suffixes:

  Prefixes- سن، سي، ست، لي، لن، لت، لل

  Suffixes- ون، ات، ان، ين، تن، تم، كن، كم، هن، هم، يا، ني، تي، وا، ما، نا، ية، ها، اء

- If the word is equal or greater than four letters long, remove the following prefixes/suffixes:

  Prefixes- ت، ي، ب، ل

  Suffixes- ت، ة، ه، ا، ي

- If no root was found, the word is then processed through the second phase of the algorithm.

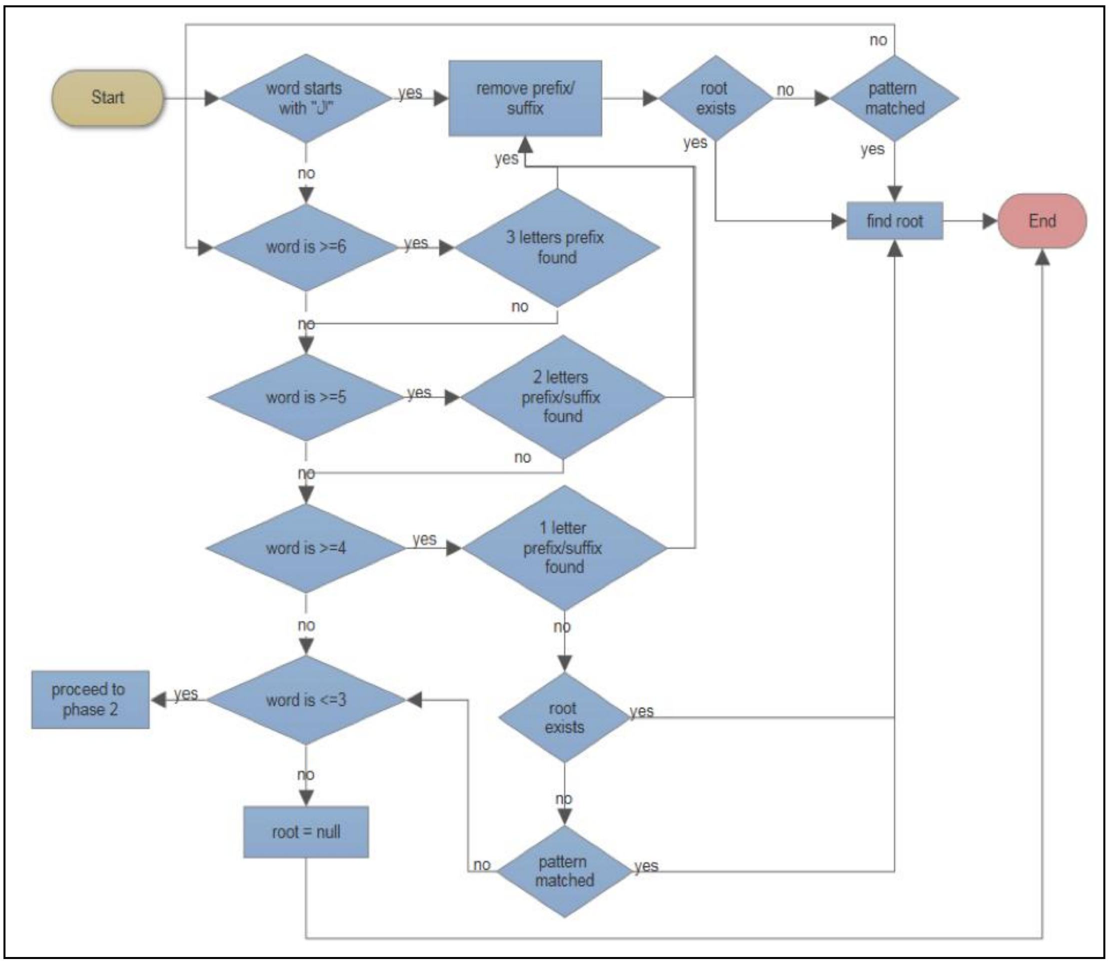


Figure 4.1. Flowchart Representation of Phase 1.

4.1.2 Second Phase

In this phase, the cases of hamzated, weak, geminated and eliminated-long-vowel words are handled.

- If the word contains one of the hamzated letters ‘ؤ’, ‘ئ’ or ‘ء’, such as in the word (يؤكل), or the letter ‘آ’ which expresses the hamzated *alif* ‘أ’ with the long vowel *alif* ‘ا’ as in the word (مآرب), change it to ‘أ’ then validate if it was a root or not. In the case of (يؤكل), the letter ‘ي’ is removed in Phase 1 and the letter ‘ؤ’ is changed to ‘أ’, giving us the correct root (أكل).
- If the word’s second letter is weak, ‘ا’, ‘ي’ or ‘و’, then change it to ‘و’, if the root was not present change it to ‘ي’, if the root was not found change it to ‘ا’. An example of this case is the word (قال) whose root is (قول).

- If the root is still not found, that mean the word is either geminated or an eliminated-long-vowel word with one letter prefix/affix. In this algorithm we remove the prefixes of the letters ‘ي’ and ‘ت’ like in the word (يجن) which is geminated, and the word (تقُم) which is an eliminated-long-vowel word.
- If the root was not found and the word is three letters long, return root was not found, otherwise if the word is two letters long, proceed to Phase 3.

4.1.3 Third Phase

In this phase, words that are two letters long are handled. These words can either be geminated, eliminated-long-vowel or hamzated with an *alif* that is removed from imperative verbs like in (كُل) and (خُذ) of the roots (أكل) and (أخذ).

- The first step here is to double the last letter of the word that is geminated, as doubled verbs roots have the second highest percentage of popularity in Arabic language after consonant verbs [5]. An example of this case is the verbs (صدّ) and (مرّ) of the roots (صدد) and (مدد).
- If the root is still not valid, the word could be an eliminated-long-vowel. Thus we add the long vowel ‘و’ in between, as weak roots of the vowel ‘و’ comes third in the roots popularity list.
- If the root was not found, add the vowel ‘ي’ in the middle of the word.
- Lastly, if the root is still not found, add a hamzated *alif* in the beginning then check for root validation. If the root was not found at this stage, return root was not found.

4.2 Implementation

The Root Extraction method developed is implemented using Java programming language, which have been used in many stemming and document classification application such as Lucene Apache software [53], and Carrot² Classification Software [54] [55]. The implementation process is divided in different stages for each phase of the Root Extraction method. In this section, each stage of implementation is explained separately. These stages are identified as the following:

a. Pre-processing Stage.
b. Affixes Elimination and Pattern Matching Stage.
c. Handling Hamzated, Weak and Geminated Words.
d. Final Stage (Handling Two-Lettered Words).

For each of the stages above, a separate class is defined in the Java code to create a sustainable design, and develop a high standard of coding quality.

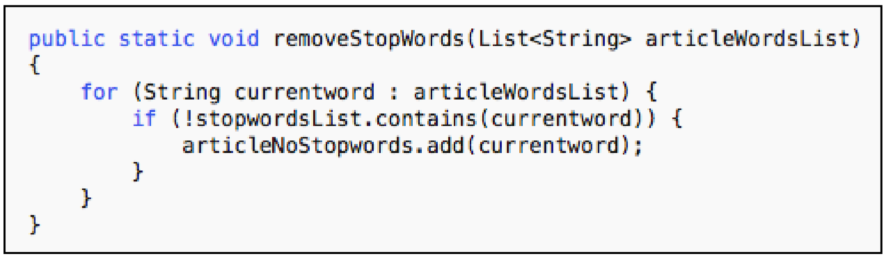
Pre-processing Stage

In this stage, the code is initialized to enable reading numerous text files and process them accordingly for stemming, stop words removal and the root extraction process overall. This task is done in the main class of the code, "RootExtractor.java".

The "main" function in the RootExtractor class is the starting point of the implementation. At first, all files are read one by one, and an ArrayList is created for them by a separate function named "createList". This function accepts each name of the text files as a string to be parsed and processed. Each file is open opened in Java, where all lines are read one by one adding their contents in another ArrayList belonging to that text file.

The next step of this stage is to remove all stop-words which should be eliminated before the root extraction process. As each word is parsed, it is matched against all stop-words that are listed in a different text file using two loaded

ArrayLists. After that, stop-words are removed from each text file's ArrayList as seen in Figure 4.2.



```
public static void removeStopWords(List<String> articleWordsList)
{
    for (String currentword : articleWordsList) {
        if (!stopwordsList.contains(currentword)) {
            articleNoStopwords.add(currentword);
        }
    }
}
```

Figure 4.2. Instantiation of the removeStopWords() Function.

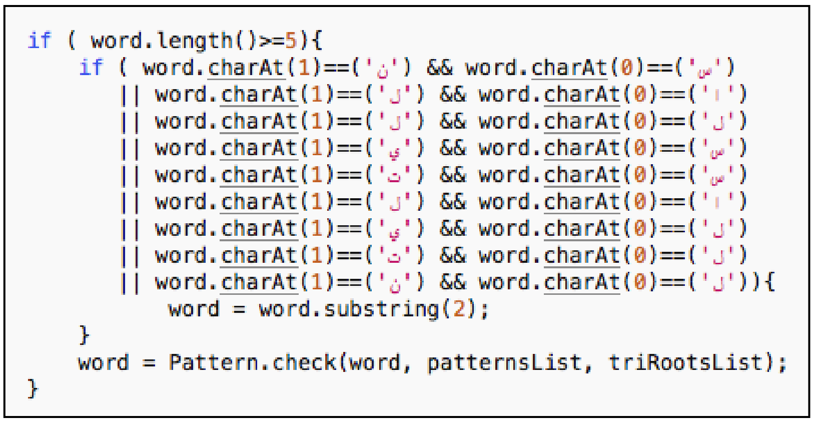
Affixes Elimination and Pattern Matching Stage

In this stage the words are processed according to their length form longer to shorter. Each word is to be checked for specific suffix/prefix removal for each length. The word is then checked if it's a root or not. If the root is found, the root is extracted and returned. Otherwise, the word is checked against a list of defined morphological patterns. If the word matches a pattern, the root is extracted and is checked against list of roots. Finally if the root is found it is returned, otherwise the word is processed further according to its length. For these two main procedures, two classes are implemented, SuffixPrefix.java and Pattern.java. In addition, the class Root.java is implemented to apply the root validation step.

- **SuffixPrefix.java**

  The SuffixPrefix class contains a "remove" function which takes 3 arguments. The first argument is the word which is currently being processed. The second one is the patternList which is to be matched against it. While the third is the rootList, which is employed when applying the root validation step.

  The word is passed through this class after being processed in the main RootExtractor class. After that, the length of the word is checked, and then it is passed to the SuffixPrefix class where suffixes and prefixes are removed accordingly. Several 'if' statements are defined for each word length, starting with the length 6 or more, and then decreasing one level at a time until reaching length 4. As the length and condition are matched, the appropriate suffixes/prefixes are removed. For example, if the length of the word is 5 and the last two characters of it are 'سن', which represents one of the prefixes which should be removed from words of length 5, they are removed. An instantiation of the code to remove suffixes from words of 5 letters is shown in Figure 4.3. After an affix is eliminated, the word is returned to the Pattern class to be checked against the appropriate patterns, and is then passed to the "check" function where the class Root is employed for the root validation process.



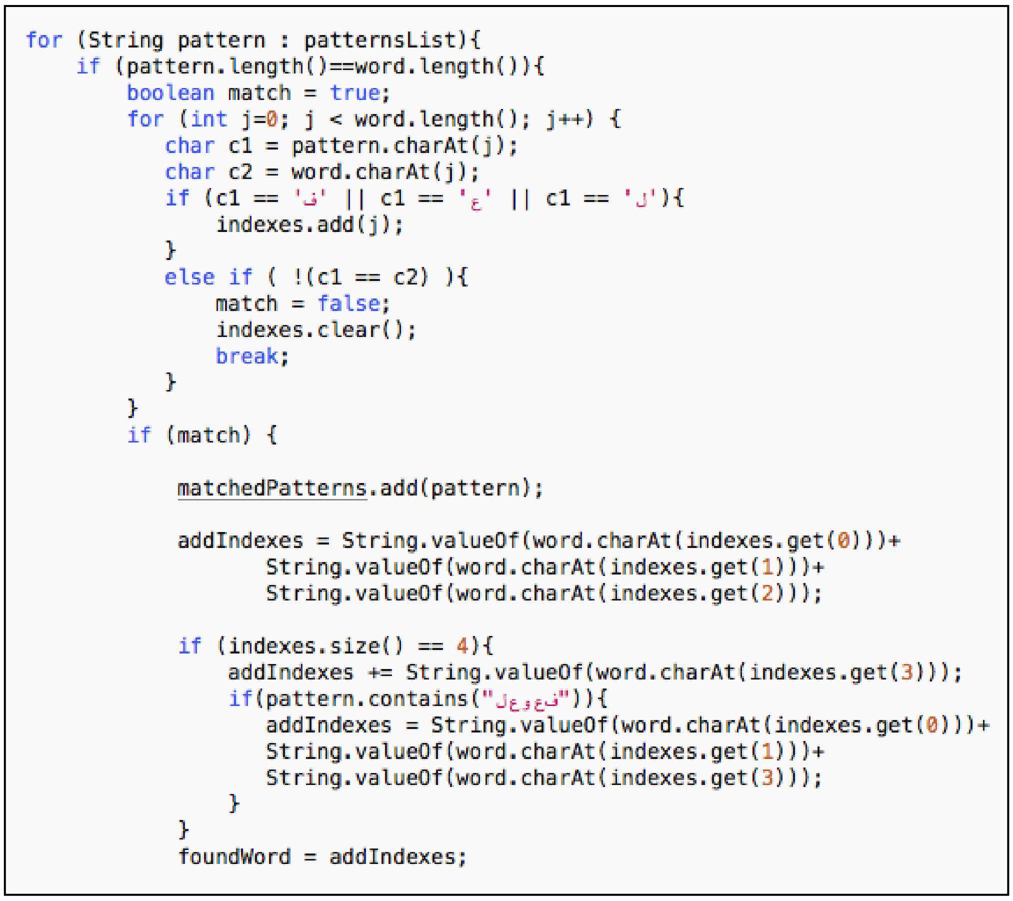
```
if ( word.length()>=5){
    if ( word.charAt(1)==('ن') && word.charAt(0)==('س')
       || word.charAt(1)==('ل') && word.charAt(0)==('ا')
       || word.charAt(1)==('ل') && word.charAt(0)==('ل')
       || word.charAt(1)==('ي') && word.charAt(0)==('س')
       || word.charAt(1)==('ت') && word.charAt(0)==('س')
       || word.charAt(1)==('ل') && word.charAt(0)==('ا')
       || word.charAt(1)==('ي') && word.charAt(0)==('ل')
       || word.charAt(1)==('ت') && word.charAt(0)==('ل')
       || word.charAt(1)==('ن') && word.charAt(0)==('ل')){
           word = word.substring(2);
    }
    word = Pattern.check(word, patternsList, triRootsList);
}
```

Figure 4.3. Code Implementation to Remove Suffixes for 5 Lettered Words.

- **Pattern.java**

In this class, the word is matched against a list of patterns of the same length, where each letter of the word is compared with the letters of the pattern of the same position. If all letters matches, except those of the positions of the three consonant root letters (ف-ع-ل), the root is extracted. In this class, the function CharAt() of the java's String class is used compare the letters of the words against those of the pattern, as shown in Figure 4.4.

As well, this class contains the "check" function which is employed in previous class SuffixPrefix.java, where the word is checked against the lists of roots to validate the root if found.

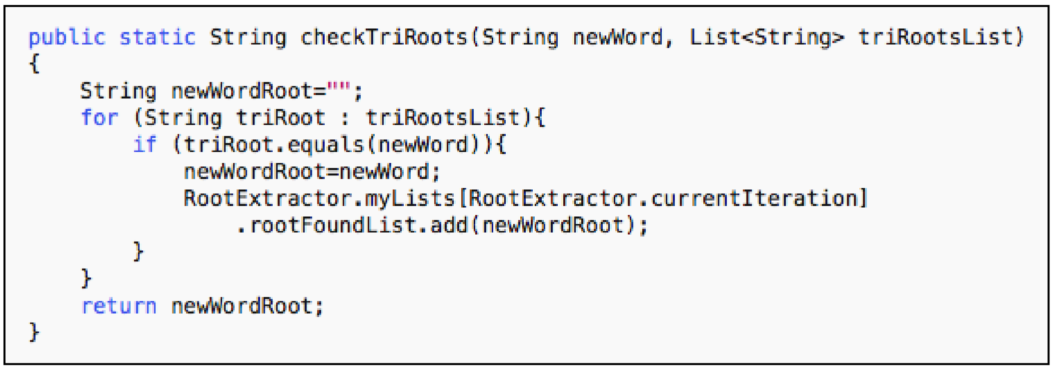


```java
for (String pattern : patternsList){
    if (pattern.length()==word.length()){
        boolean match = true;
        for (int j=0; j < word.length(); j++) {
           char c1 = pattern.charAt(j);
           char c2 = word.charAt(j);
           if (c1 == 'ف' || c1 == 'ع' || c1 == 'ل'){
               indexes.add(j);
           }
           else if ( !(c1 == c2) ){
               match = false;
               indexes.clear();
               break;
           }
        }
        if (match) {

            matchedPatterns.add(pattern);

            addIndexes = String.valueOf(word.charAt(indexes.get(0)))+
                   String.valueOf(word.charAt(indexes.get(1)))+
                   String.valueOf(word.charAt(indexes.get(2)));

            if (indexes.size() == 4){
                addIndexes += String.valueOf(word.charAt(indexes.get(3)));
                if(pattern.contains("فعوعل")){
                   addIndexes = String.valueOf(word.charAt(indexes.get(0)))+
                   String.valueOf(word.charAt(indexes.get(1)))+
                   String.valueOf(word.charAt(indexes.get(3)));
                }
            }
            foundWord = addIndexes;
```

Figure 4.4. Implementation of the Pattern Matching Process.

- **Root.java**

In this class, the word is processed class to match it against the list of TriRoots or Quadroots according to its length. These roots are compiled from two text files into two different ArrayLists, and consist of 3800 tri-literal and 900 quad-literal roots. As the word is processed through this class, it is passed to one of the functions, checkTriRoot or CheckQuadRoot of the Root Class. The implementation of the checkTriRoots() function is shown in Figure 4.5. Both functions receive two arguments, one being the word to be matched, and the tri-literal or quad-literal roots list, in which the word is compared with the root list. If the word and the root match correctly, the root is found is then returned.



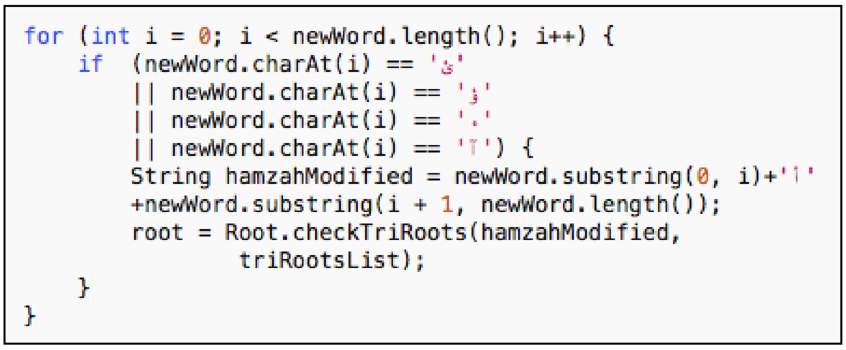
```
public static String checkTriRoots(String newWord, List<String> triRootsList)
{
    String newWordRoot="";
    for (String triRoot : triRootsList){
        if (triRoot.equals(newWord)){
            newWordRoot=newWord;
            RootExtractor.myLists[RootExtractor.currentIteration]
                .rootFoundList.add(newWordRoot);
        }
    }
    return newWordRoot;
}
```

Figure 4.5. Implementation of the checkTriRoots() Function.

Handling Hamzated, Weak and Geminated Words

If this stage is reached and the root is not found, this means that the word either includes a hamzated letter, a geminated or a weak letter. Thus, defined rules are applied while the word is being checked is it was a root or not. That is by using the if-else conditions, done at the end of the main class RootExtractor before the rules of the final stage.

This step is implemented by checking each location of the word for specific characters, such as hamzated letters ‘ؤ’, ‘ئ’ and ‘ء’ and weak letters ‘ا’, ‘ي’ and ‘و’. If they match, the letter is replaced accordingly. Figure 4.6 shows the implementation of the code to replace the hamzated letters to a hamzated *alif*. Then the word is returned to the Root class to be checked is against the defined lists of TriRoots or QuadRoots depending on its length. If the root is found, it is then returned, otherwise the root is set to be not found, or the word is processed further if it was shortened to two-lettered word.



```
for (int i = 0; i < newWord.length(); i++) {
    if  (newWord.charAt(i) == 'ئ'
        || newWord.charAt(i) == 'ؤ'
        || newWord.charAt(i) == 'ء'
        || newWord.charAt(i) == 'آ') {
        String hamzahModified = newWord.substring(0, i)+'أ'
        +newWord.substring(i + 1, newWord.length());
        root = Root.checkTriRoots(hamzahModified,
                triRootsList);
    }
}
```

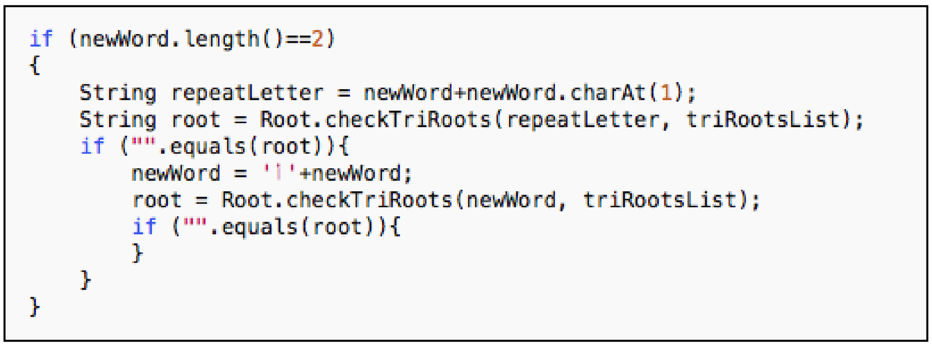
Figure 4.6. Code Implementation for Handling Hamzated Words.

Final Stage (Handling Two-Lettered Words)

The final part of implementation is to develop the code to handle the words of two letters length. Meaning if all stages are completed while the root is still not found, and the processed word is of or has reached the length of two letters, few rules are applied accordingly. The implementation includes the use of if-else statements to apply the rules and to check if the created word is a root or not. If the word is a root then the root is finally extracted. Else, apply the following rule and so on. These rules are defined as the following:

- Double the last letter, which occurs in cases of geminated words.
- Add the letter *waw* - ‘و’ in the middle of the word, for cases of eliminated vowel words.
- Add the letter *yaa* - ‘ي’ in the middle of the word, for cases of eliminated vowel words.
- Add a hamzated *alif* - ‘أ’ in the beginning of the word, for cases of eliminated *alif* of imperative verbs.

If the root is found after applying one of those rules, the root is returned. Otherwise the root is set to be not found. That is implemented at the end of the main class RootExtractor, after the code of handling hamzated, weak and geminated words. Implementation of the code handling geminated and eliminated *alif* words is presented in Figure 4.7.



```
if (newWord.length()==2)
{
    String repeatLetter = newWord+newWord.charAt(1);
    String root = Root.checkTriRoots(repeatLetter, triRootsList);
    if ("".equals(root)){
        newWord = 'أ'+newWord;
        root = Root.checkTriRoots(newWord, triRootsList);
        if ("".equals(root)){
        }
    }
}
```

Figure 4.7. Code Implementation to Handle Geminated and *alif* Eliminated Words.

4.3 Results and Discussion

Data Set

In order to support and test our algorithm, a number of entries are selected from Al-Sulaiti's online Arabic corpora [51]. The data set of Al-Sulaiti is collected to provide a prototype text material for Arabic teachers, new learners and mainly Arabic language researchers and engineers. The corpus consists of 842684 words and 14 different categories. In previous work on Arabic root extraction, most testing methods do not include manual checking to verify if the root of the word was extracted correctly and does actually belong to that word or not. Instead, the roots are defined as correct if the word was shortened to a tri-literal word, or if it did exist in a predefined list of roots. Also, the percentage of the correctly extracted roots is shown to be higher than other compared algorithms within the work despite using a different data set of different amount. Thus, we decided to manually verify the results of the algorithm, selecting several entries making up to 4341 words as the total text, to be compared with Khoja's stemming algorithm result.

Testing and Evaluation Method

Arabic stemming and root extraction research included various different algorithms, but only a few has focused on solving the problem of tri-literal words that are weak, hamzated, geminated or eliminated-long-vowel. Nevertheless, Khoja's stemmer is one of the very well-known Arabic stemming algorithms that also handle these cases. Therefore, we process our data through our root extraction system taking the text input from a text file containing the data set. The same data set is then processed through Khoja's stemming system which is available for download [32]. After that, the results of all processed words and their roots are manually checked, as some roots can be extracted for the words as long as they appear in the roots defined list, but not necessary belong to those words, thus inaccurately extracted. Both the results of the introduced algorithm and the results of Khoja's stemmer are checked and compared for evaluation.

Results and Findings

Using the same collected data set as input to both our root extraction system and Khoja's system, we achieve the results as shown in Table 4.2 and Figure 4.8. It can be seen that Khoja's system extracted 3162 roots out of 4341 words (73%), while our system has extracted 3061 roots (70%). However, not all roots were correctly extracted because of excessive root extraction steps in Khoja's algorithm that leads to extracting roots for Arabicized, Proper noun words, and words that are a combination of a prefix/suffix and a stop word. For example,Khoja's system extracts the root (طلي) for the word (إيطاليا), which means the country Italy and do not have a root in Arabic. It also extracts the root (ولي) for the word (ولا), which is a combination of the prefix *waw* (و) and the stop word (لا). Both words however are defined to have no root when processed through the introduced extraction algorithm. Although both algorithms extracted a number of inaccurate roots, which exist in the Arabic roots list but do not represent the processed word, the number of inaccurate roots extracted ishigher in Khoja's system results (13.7%) than in the proposed root extraction system (5.2%). That is due to extracting roots of Arabicized and Proper noun words as well as failing to extract the correct roots for many tri-literal weak and hamzated roots. Such as extracting the root (ربي) for the word (روايات) where the correct root which is extracted through the new algorithm is (روي), and the root (خرر) for the word (أخر), where the correct root is (أخر). Overall, our system extracted more accurate roots (65%) than Khoja's system (59%) with an improvement rate of 6% (Figure 4.7).

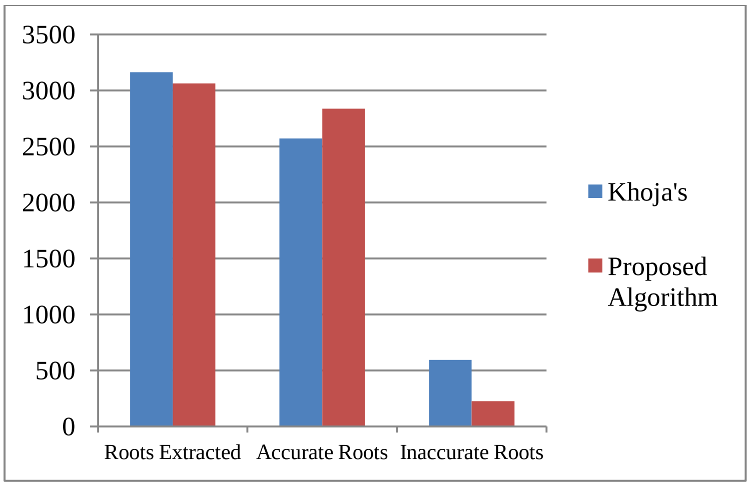


Figure 4.8. Proposed Algorithm vs Khoja's Testing Results

	Proposed Algorithm		**Khoja's Algorithm**	
# Total Roots Extracted	3061	70.5%	3162	72.8%
# Inaccurate Extracted Roots	224	5.2%	593	13.7%
# Accurate Extracted Roots	2837	65.4%	2569	59.2%

Table 4.2. Proposed Algorithm vs Khoja's Testing Results.

4.4 Summary

In this chapter an improved root extraction algorithm for Arabic words, which is based on morphological analysis and linguistic constraints was presented. The algorithm handles the problems of infixes removal by eliminating prefixes and suffixes while checking the word against a predefined list of patterns. The problem of extracting the roots of weak, hamzated, and eliminated-long-vowel words has been handled. As well as the two-letter geminated words, that is by identifying linguistic based rules to replace, eliminate or duplicate certain letters where needed. The experiments and testing were conducted by using thousands of Arabic words gathered from an online Arabic corpus which is collected to aid Arabic language based research. Human judgment was applied to evaluate the results and accuracy of the algorithm. The algorithm is introduced with the aim of supporting Arabic stemming/root extracting tools. The results obtained shows that the root extraction algorithm is promising and is worth being applied in various Arabic language processing programs.

Chapter 5

Documents Classification

It is mentioned previously in this thesis that text classification requires the use of text pre-processing methods and feature identification algorithms to represent the text before processing text classification.

In this chapter, a new approach to identify significant keywords for Arabic corpora is presented. As well, three different text classification methods are implemented using the extracted keywords as the main features for classification. Finally, the classifications results of each method are evaluated and discussed as a whole.

5.1 Feature Selection

5.1.1 Methodology

The feature selection procedure is done by implementing the advanced stemming and root extraction algorithm, as well as Term Frequency-Inverse Document Frequency (TF.IDF) topic identification method [56] Both methods are employed to find features of different data sets representing six different categories. These are culture, economy, international, local, religion and sports. The first step to identify the features is to stem and index each data set. After that, the TF.IDF algorithm is used to calculate the weights of the roots to identify the highly significant keywords for each category. A prototype vector is then built using the keywords selected. The weight for each element in the vector is obtained as the combination of the term frequency TF (*w*,*d*), that is the number of times the word w is repeated in the document *d*, and IDF (*w*), which is the inverse document frequency [57] [58]. The weight of the word w_i in document *d* is called d_i and is obtained using equation (5.1) [57]:

$$d_i = TF(w,d) \times IDF(w) \qquad (5.1)$$

The IDF (w), which is the inverse document frequency of the term, is calculated by applying equation (5.2) [57]:

$$IDF(w) = \log\left(\frac{N}{DF(w)}\right) \qquad (5.2)$$

where N is the total number of documents and DF is the number of documents in which the term has occurred in [57]. As we calculate the TF.IDF value for each term in the data set, we extract the terms with the highest TF.IDF to present the highly significant terms of the topic.

5.1.2 Implementation

In order to implement the TF.IDF algorithm to select topic features, the Root Extraction system is employed. That is to extract the roots of all articles in the category and calculate their TF.IDF values subsequently. To accomplish this task, two functions are built as an extension to the Root Extractor. These are calculateIDF() which calculates the IDF values, and calculate tfidf() which calculates the TF.IDF values. In FiguresFigure 5.1 andFigure 5.2, instantiations of both functions are presented.

```
public static void calculateIdf() {

    int currentCounter = 0;
    HashMap<String, Integer> CountMap_df = new HashMap<>();
    List<String> wordslist = new ArrayList<String>(
            MyLists.allwords_tf.keySet());

    for (int y = 0; y < wordslist.size(); y++) {
        for (int i = 0; i < numberOfFiles; i++) {
            if (myLists[i].rootFoundList.contains(wordslist.get(y))) {
                currentCounter++;
                if (CountMap_df.containsKey(wordslist.get(y))) {
                    CountMap_df.put(wordslist.get(y),
                            CountMap_df.get(wordslist.get(y)) + 1);
                } else {
                    CountMap_df.put(wordslist.get(y), 1);
                }
            }
        }

        float temp = (float) Math.log10((float) numberOfFiles
                / (float) currentCounter);

        MyLists.idf.add((float) temp);
        currentCounter = 0;

    }
}
```

Figure 5.1. Implementation of the calculateIdf() function.

```
public static void calcultetfidf() throws FileNotFoundException,
    UnsupportedEncodingException {

    List<String> wordslist = new ArrayList<String>( MyLists.allwords_tf.keySet());

    for (int y = 0; y < wordslist.size(); y++) {
        String currentword = wordslist.get(y);
        Float idf = MyLists.idf.get(y);
        Integer tf = MyLists.allwords_tf.get(currentword);
        Double tfidf = (double) (tf * idf) ;
        MyLists.tfidf.put(currentword, tfidf);
    }

    ValueComparator bvc =  new ValueComparator(MyLists.tfidf);
    TreeMap<String,Double> sorted_map = new TreeMap<String,Double>(bvc);
    sorted_map.putAll(MyLists.tfidf);

    System.out.print(sorted_map.size()+" roots in ");
    ReadWrite.writeToFile(folderPath, sorted_map, sorted_map.size());
}
```

Figure 5.2. Implementation of the calculate tfidf() function.

5.1.3 Experiments and Results

Until recently, there are no standard Arabic text corpora for data mining and classification research purposes. However, a number of studies are trying to scientifically compile representative training data sets for Arabic text classification, which cover different text topics that can be used in future as a benchmark [59].

Therefore, many works dealing with topic identification or text categorization for Arabic language were conducted out using non representative and small corpora. In order to test this algorithm, a text corpus of 1000 articles which corresponded to thousands of words is selected. The corpus was retrieved from an online Arabic database resource providing thousands of Arabic online newspaper articles (http://www.sourceforge.net/projects/arabiccorpus). The text we selected for testing belongs to the 'Culture and Education' category (المقالات الثقافية) and should extract culture related terms as the highly significant topics.

In our experiment, a data pre-processing step was conducted before the stemming and weighting stage. Every article was processed to remove punctuation marks and digits and eliminate stop words. After that, we implement our root-based stemmer to extract and index the words as roots, reducing the number of indexed terms and to achieve a better result covering the main terms of the category avoiding repetitive listing of words that belongs to the same morpheme. Sequentially, the TF.IDF values are calculated to extract the highly significant terms. As we calculated the TF.IDF values, we extracted the top ten terms to represent the category, as shown in Table 5.1.

TF.IDF	Extracted Term
4318	علم
3891	عمل
3816	عرض
3306	كتب
3182	عرب
3160	شعر
3303	فنن
2334	سرح
2267	ثقف
2156	حدث

Table 5.1. Terms extracted via Root-stemming and TF.IDF.

Subsequently, the extracted terms are compared with the top TF.IDF terms extracted from the same category articles where a light stemmer is implemented to stem the words [30], as shown in Table 5.2.

TF.IDF	Extracted Term
1867	عربية
1308	عربي
1192	عالم
1179	عمل
1063	كتاب
1056	فنية
1053	فن
1042	ثقافة
1030	معرض
948	فنان

Table 5.2. Terms extracted via light stemming and TF.IDF.

It can be seen that the terms represented by roots in our results have a higher TF.IDF value as more than one world relates to the same morpheme. For example, our results list the root (فنن) of the noun (فن) instead of listing more than one word that belongs to the same noun, such as (فنية), (فن) and (فنان). As well as for the words (عربية) and (عربي), the term extracted using our method is عرب. These roots weights higher TF.IDF values than terms extracted without the use of advanced stemming and root extraction methods. Thus, implementing a feature selection algorithm by combining root-based stemming as well as TF.IDF weighting approach gives a result of less indexed terms and a more efficient terms weighting than when using light stemming methods.

5.1.4 Main Feature Selection

Consequent to the results of the root-based TF.IDF feature selection method, it is implemented to define the main features of several Arabic text corpora. Most research on Topic Identification or Text Classification for Arabic language was conducted by utilizing non representative small corpora [60] [61] [62] [63]. This could be a cause of erroneous results and inaccurate evaluation. One of the few studies in the field which was carried out employing a fairly representative corpus is presented in [64]. However, the number of categories included was three only, which is not sufficient to produce accurate results.

Thus, the corpus selected to aid the text classification phase of this research, was retrieved from an online Arabic database resource, which provides thousands of Arabic online newspaper articles (http://www.sourceforge.net/projects/arabiccorpus). The text corpora selected contains six different categories and nearly 9000 articles, and was employed in few studies including [30], while [4] employed part of it. Those categories are Culture, Economy, International, Local, Religion, and Sports. In Table 5.3. Number of Documents, the number of articles for each category is presented.

Category	**Number of Documents**
Culture	2782
Economy	3468
Local	2035
International	3596
Religion	3860
Sports	4550

Table 5.3. Number of Documents for each Category.

In order to select the main features for each category, the training database which represents 80% of each category are employed. The database is first pre-processed to remove stop-words, punctuations and unnecessary diacritics, while replacing the important ones. After that, it is processed through the root-based TF.IDF feature selection system where optimal features are selected for each category, according to their TF.IDF scores. In TablesTable 5.4 -Table 5.9, the top ten features for each category alongside their scores and examples of possible derived words are shown.

TF.IDF	Words Examples	Related Words	Term
20411.05	مكتبة، كتاب ، كاتب، كتب، مكتوب	Writing, Library	كتب
18847.66	معرض، عروض، يعرض، استعرض	Exhibition	عرض
15669.13	شاعر، شعراء، شِعر، شعرية	Poetry	شعر
14964.31	تشكيلي، تشكيلية، أشكال	Fine Art	شكل
13631.92	تمثيل، تمثيلية، ممثل، ممثلون	Play, Act, Actors	مثل
12048.44	عربي، عربية، عروبة، عرب	Arab, Arabian	عرب
11757.18	كائن، كائنات، مكون، تكون	Creation	كون
11171.34	فنون، فن، فنان، فني	Art, Artist	فنن
10910.20	كثير، كَثُرَ، تكاثر، استكثر	Increment	كثر
10890.13	تعبير، معبر، عبٌر	Expressing	عبر

Table 5.4. Features Selected for the 'Culture-ثقافية' Category.

TF.IDF	Words Examples	Related Words	Term
24681.27	تداول، مداولة، دول	Country, Stock Exchange	دول
20177.52	خلال، تخلل	Involved	خلل
19988.14	اقتصاد، اقتصادي، اقتصادية	Economy	قصد
19953.71	منتجات، تنتج، منتجة	Product	نتج
18404.06	أسوق، سوّق، أسواق، سوق	Marketing, Market	سوق
17815.44	مالي، أموال، تمويل	Money, Finantial	مول
15956.08	مصانع، صنع، يصنع، مصنوعات	Factory, Making	صنع
15918.21	أسهم، ساهم، مساهمة، مساهمون	Stock, Stock holder	سهم
15824.37	قطاع، استقطاع، قطع، انقطاع	Sector	قطع
15206.12	مشروع، مشاريع، مشروعات	Project	شرع

Table 5.5. Features Selected for the 'Economy-اقتصادية' Category.

TF.IDF	Words Examples	Related Words	Term
13787.63	رئيس، رئاسة، رؤساء، يرأس	Leader, Presidency	رأس
12724.58	وحدة، توحيد، موحد	Unity, Unite	وحد
11552.32	حاكم، حكومة، محكمة، أحكام	Rules, Government	حكم
11208.67	أقل، أقال، أقيل، إقالة، استقالة	Quit, Fired	أقل
8429.17	أعراق، عرق، عراق، عريق	Race, Origins, Ancient	عرق
8205.78	حقق، تحقيق، محقق، حقيقة	Investigate, Truth	حقق
7592.35	دولي، دول، تداول، مداولة	Country, Exchange	دول
7415.71	ولاية، ولي، استيلاء، تولى	State, Guardian	ولي
6674.06	مسؤول، مسؤولية، مسائل	Responsibility	سأل
6426.66	ناخب، انتخاب، انتخب	Elections, Elected	نخب

Table 5.6. Features Selected for the 'International-عالمية' Category.

TF.IDF	Words Examples	Related Words	Term
24033.76	طالب، مطالبة، طلاب، طلبات	Student, Request	طلب
21529.82	دراسة، مدرسة، تدريس، مدرسون	Study, School Teaching	درس
18284.75	نطاق، ناطق، منطقة، مناطق	Area	نطق
17886.96	حميد، يحمد، محمود، حامد	Thanking	حمد
16375.33	تعلية، علو، عليه، أعلى	Rise, Promotion	علي
16245.06	تركيز، مركز، مراكز	Centre, Focus	ركز
15855.13	خلال، اختلال، مخل، تخلل	Involved	خلل
15094.73	قام، قيم، مقاومة، قيام	Stand, Resistance	قوم
14356.54	مشروع، تشريع، شرعية	Project	شرع
14050.21	محاضرة، محاضر، حضور، محضور	Talk, Guest	حضر

Table 5.7. Features Selected for the 'Local-محليات' Category.

TF.IDF	Words Examples	Related Words	Term
52252.21	قال، قوله، قيل، قول	Saying, Article	قول
51342.675	تعالى، عليه، علاء، علو، أعلى	Rise, Higher	علي
43563.39	ولاية، ولي، ولاء، توليه	State, Guardian	ولي
38164.648	رسول، رسالة، مرسلين، رسل	Prophet, Send, Message	رسل
37477.15	وكّل، توكيل، وكالة	Empower	وكل
34618.99	أقل، أقال، أقيل، إقالة	Quit, Fired	أقل
32371.37	تكون، تكوين، كيان، كان	Being, Creation	كون
31368.11	صلاة، مصلّون، مصلّى، صلّى	Pray, Prayer	صلي
30993.07	نفوس، تنفس، أنفاس، نفس	Breath, Soul	نفس
24864.95	كثير، كَثُرَ، كثرة، استكثر	Increment	كثر

Table 5.8. Features Selected for the 'Religion-دينية' Category.

TF.IDF	Words Examples	Related Words	Term
37578.00	فريق، فِرَق، فرق، فروقات	Team	فرق
36589.12	ملعب، لاعب، لعبة، ملاعب	Field, Players	لعب
31424.21	منتخب، ناخب، منتخبات	National Team	نخب
26906.35	بطولة، أبطال، بطولات، بطل	Championship	بطل
25217.00	واحد، توحيد، موحد، وحدة	Unity, United	وحد
23098.08	حمْد، محمد، حامد، يحمد	Greatful	حمد
21980.21	سباق، مسابقة، متسابق	Race, Competition	سبق
20441.41	ثاني، ثنائي، ثناء، أثنى	Dual	ثني
19032.99	هدف، أهداف، هداف	Goal	هدف
18814.56	مركز، مراكز، يركز، تركيز	Centre, Position, Focus	ركز

Table 5.9. Features Selected for the 'Sports-رياضية' Category.

After selecting the top ten features for each category, they are set as vectors and are implemented in documents classification. In the next sections, both training and testing database, which was not included in feature selection, are processed through different classification methods. At first, each document in the database is processed through the root-based extractor, calculating the document’s terms frequencies, and then employing those measures to classify the document through different classification methods.

5.2 Terms Frequency-based Classification

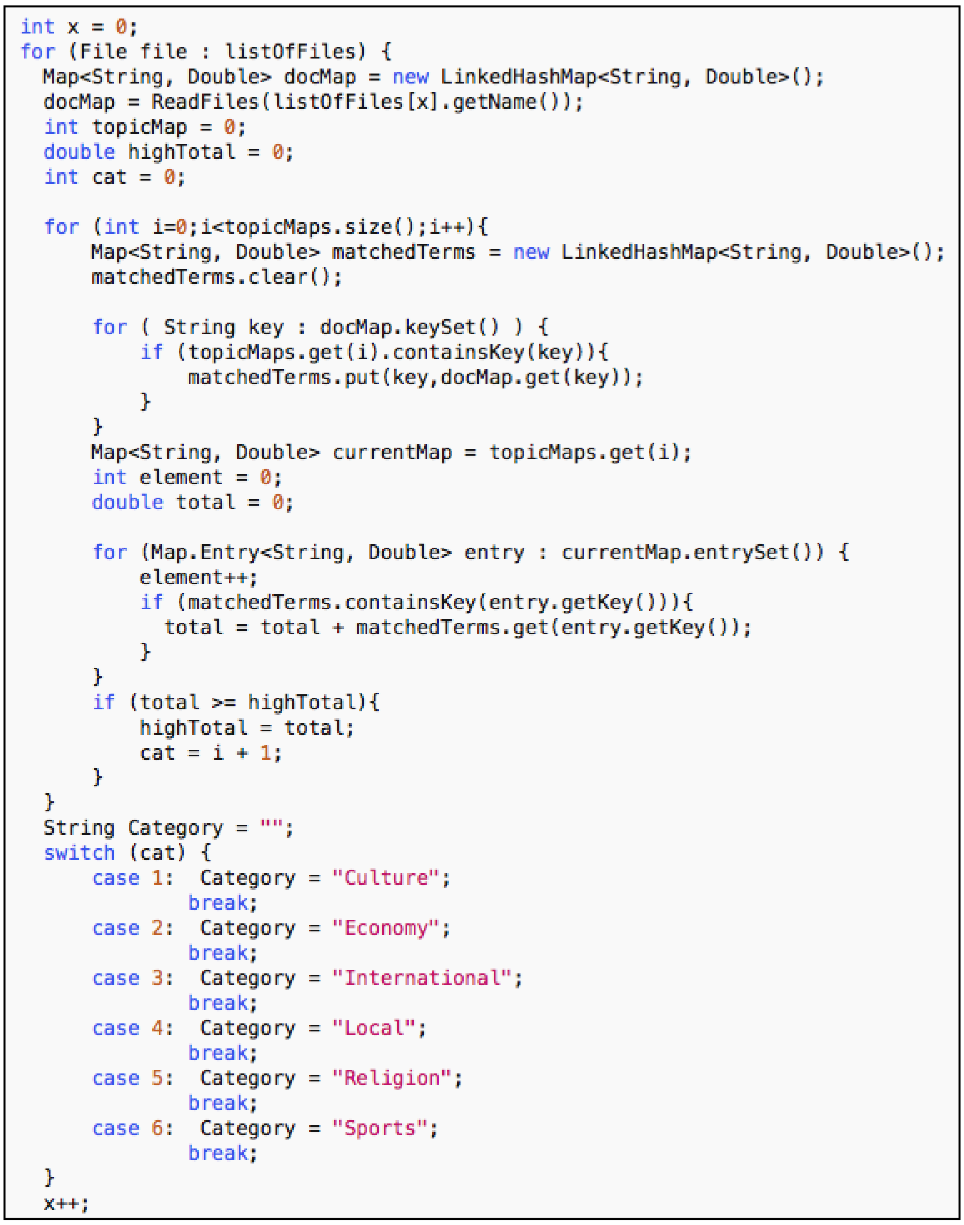
5.2.1 Methodology

Terms Presence and Term Frequency have been a significant measure in documents classification techniques. In this section, a document classification approach is introduced, which is completely based on term weighting and terms presence within each document and category documents as a whole. In this method the topic vectors selected through the TF.IDF method, as listed in previous section, are employed to classify each article in the database. That is done by pre-processing each document in each category, and then processing it to extract the roots. Meanwhile, all the terms in an article matching those in the topic vectors, and their frequencies are calculated. Subsequently, the document topic is found depending on the frequencies total found for each topic vector. In other words, the document topic with the highest frequencies sum is selected.

5.2.2 Implementation

In order to implement the Term-Frequency-based Classifier, the class TF.IDFClassifier.java was built while utilizing the output files of the Root Extraction system. These include the roots lists extracted from each article in the database alongside their frequencies. As well, each topic vector selected through the TF.IDF system is utilized. In the main function of the class, each vector is added as a topicMap in a Maps ArrayList, ArrayList<Map>(). The category folder is also initiated at the beginning and all its roots files are added to a File[] type of list named listOfFiles. As well, the variables topicMap, highTotal and cat, are initiated to define the category of the highest total. Next, a LinkedHashMap<String, Double>() named matchedTerms is created. That is to add the terms and their frequencies of the article's roots, which matches the topicMap terms.

After that, the total of frequencies for matched terms is calculated for the topicMap, and is set as the highTotal at first. Subsequently, the frequencies total is calculated for the next topicMap, and is set to the highTotal if is higher than the previous one, and so on. At last, the category is selected depending on the highest frequencies total, for each article in the category. In Figure 5.3, an instantiation of the main part of the Terms Frequency-based Classifier code is presented.



```
int x = 0;
for (File file : listOfFiles) {
  Map<String, Double> docMap = new LinkedHashMap<String, Double>();
  docMap = ReadFiles(listOfFiles[x].getName());
  int topicMap = 0;
  double highTotal = 0;
  int cat = 0;

  for (int i=0;i<topicMaps.size();i++){
      Map<String, Double> matchedTerms = new LinkedHashMap<String, Double>();
      matchedTerms.clear();

      for ( String key : docMap.keySet() ) {
          if (topicMaps.get(i).containsKey(key)){
              matchedTerms.put(key,docMap.get(key));
          }
      }
      Map<String, Double> currentMap = topicMaps.get(i);
      int element = 0;
      double total = 0;

      for (Map.Entry<String, Double> entry : currentMap.entrySet()) {
          element++;
          if (matchedTerms.containsKey(entry.getKey())){
            total = total + matchedTerms.get(entry.getKey());
          }
      }
      if (total >= highTotal){
          highTotal = total;
          cat = i + 1;
      }
  }
  String Category = "";
  switch (cat) {
      case 1:  Category = "Culture";
              break;
      case 2:  Category = "Economy";
              break;
      case 3:  Category = "International";
              break;
      case 4:  Category = "Local";
              break;
      case 5:  Category = "Religion";
              break;
      case 6:  Category = "Sports";
              break;
  }
  x++;
```

Figure 5.3. Implementation of the Terms Frequency-based Classifier.

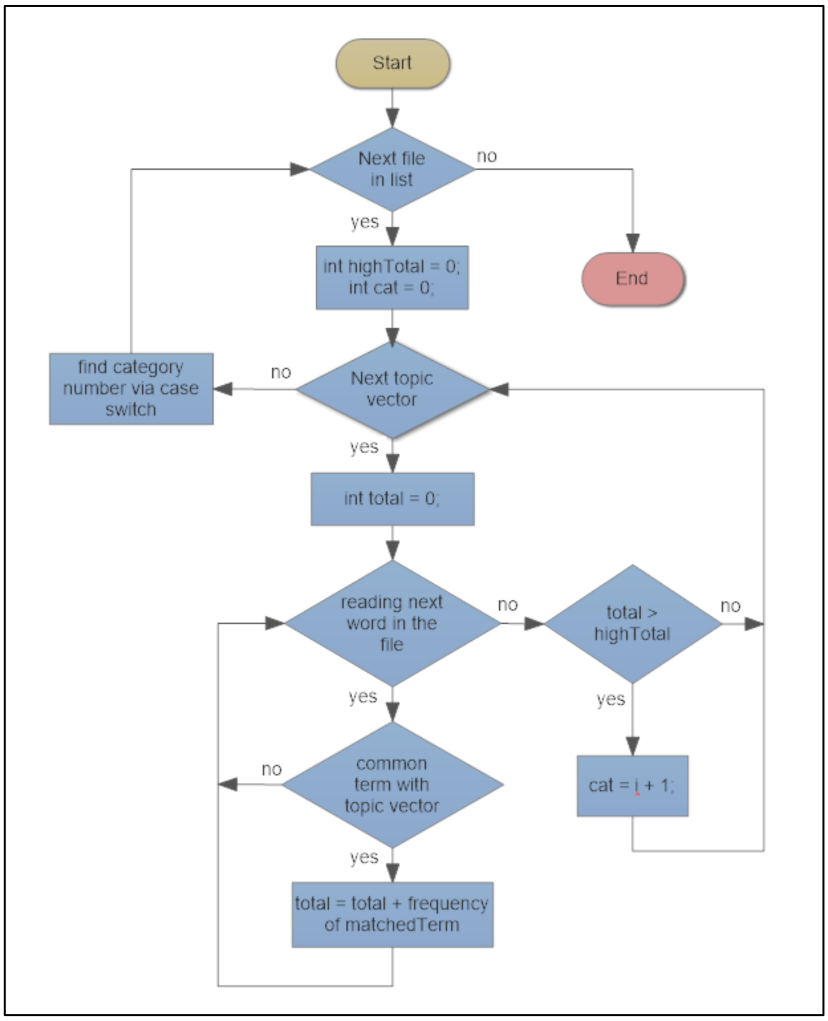


Figure 5.4. Flowchart Representation of the Terms Frequency-based classifier Implementation.

5.2.3 Results

After the implementation is done, documents of both training database which represents 80% articles of each category, and the testing database which represents 20%, are processed for classification. The execution time here was fairly rapid, as it took about 15 minutes to classify all files. In Table 5.10, the final results of the Terms Frequency-based classification method are presented. The accuracy of the classification results for each category is calculated as the percentage of articles that are classified to the correct category. The classification results of each classification method are discussed further in section 5.5 of this chapter.

Category		Culture	Economy	International	Local	Religion	Sports
Training Database	**Classification Accuracy**	58.2%	59.8%	73.7%	46.6%	61.8%	77.4%
Testing Database		60.6%	60.7%	75.7%	51.8%	79.7%	77.6%
Overall Results		59.4%	60.3%	74.7%	49.2%	70.8%	77.5%

Table 5.10. Terms Frequency-based Classification Results.

5.3 Artificial Neural Networks Classification

5.3.1 Introduction

Neural networks are the powerful tools used for forecasting of recent developments in artificial intelligence research and many information retrieval and classification applications [65] [66]. These involve non-linear models that may be used for mapping of past and future trends and time series data, and for revealing the hidden relationships and structures that govern them. The tools are used in several applied fields, for example economics, computer sciences, and medicine. They are used in the analysis of the relationships among financial and economic phenomena, generating time-series and optimization, and forecasting and filtration [67].

ANNs are considered as an electronic model based on a neural structure similar to the human brain. This modelling involves a less technical way of generating solutions, much as the brain does on the basis of experience. ANN is a non-linear self-adaptive data driven method. It takes vector ($y_j \ldots y_k$) as input and is a type of real function. The output is usually a function, mostly a sigmoid function i.e. tangent hyperbolic or logistic function. These types of functions (multilayer perceptron) consist of combinations of weighted sums of the functions parallel to the neurons. Cascade-forward and feedforward networks are particularly applicable in approximation functions when all inputs and outputs are known. The Neural network training parameters are:

- The initial weights and biases randomly between -1 and +1
- Training parameters learning rule Back-propagation
- Adaptive learning rate is 0.001
- Momentum constant is 0.9
- Acceptable mean-squared error is 0.001
- Performance function: mean square error (MSE)

There are several types of neural networks that work effectively and efficiently to execute the process of the research.

5.3.2 Methodology

In ANN, the nodes are seen as ‘artificial neurons’, where each artificial neuron is a computational model inspired by natural neurons [68]. Biological natural neurons, as seen in Figure 5.5, receive signals through *synapses* that are located on the *dendrites* or *membrane* of the neuron. When the signals received are strong enough (surpass a certain *threshold*), the neuron is *activated* then emits a signal through the *axon*. This signal might be sent to another *synapse*, and might activate other neurons.

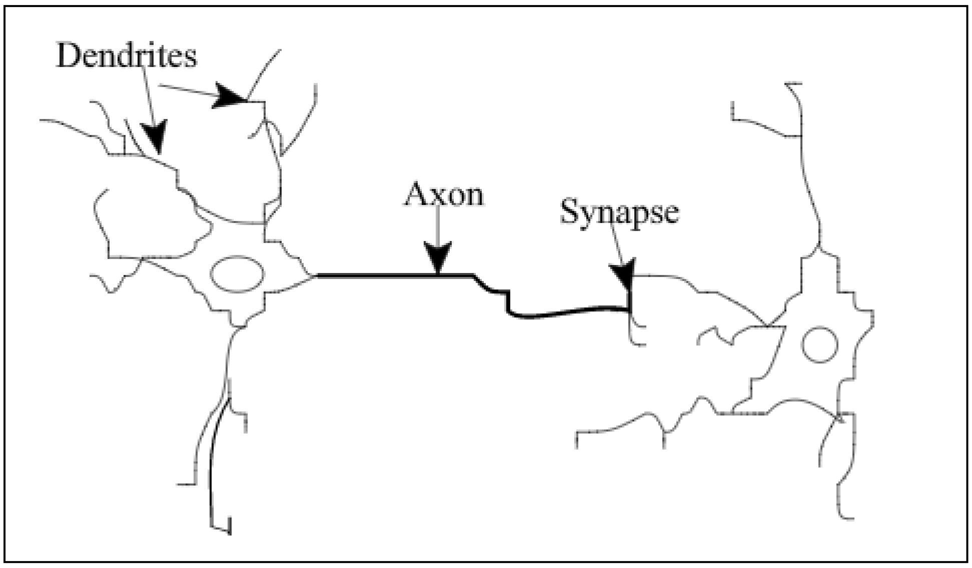


Figure 5.5. Biological Neuron Model.

The complexity of real neurons is significantly simplified when modeling artificial neurons, as seen in Figure 5.6. These basically consist of *inputs* like synapses, which are multiplied by *weights,* the strength of the respective signals, and then computed by a mathematical function which determines the *activation* of the neuron. Another function, which may be the identity, computes the *output* of the artificial neuron, in some cases in dependance of a certain *threshold*. ANNs combine artificial neurons in order to process information.

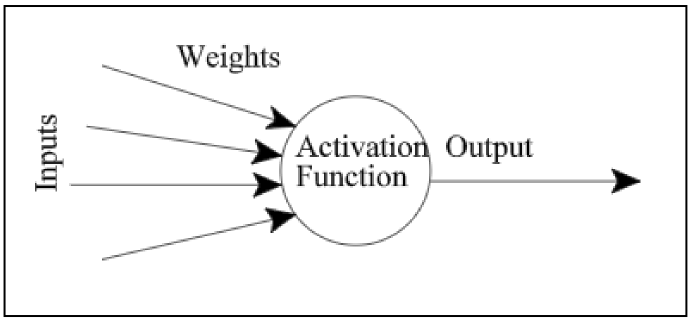


Figure 5.6. An Artificial Neuron

The Backpropagation Algorithm

The backpropagation algorithm is used in layered feed-forward Artificial Neural Networks. Backpropagation is a multi-layer feed forward, supervised learning network based on gradient descent learning rule. Where the algorithm is provided with examples of the inputs and outputs that the network is required to compute, and then the error, difference between actual and expected results, is calculated. The idea of the backpropagation algorithm is to reduce this error, until the Artificial Neural Network *learns* the training data. Figure 5.7 shows a diagram of the feed forward ANN.

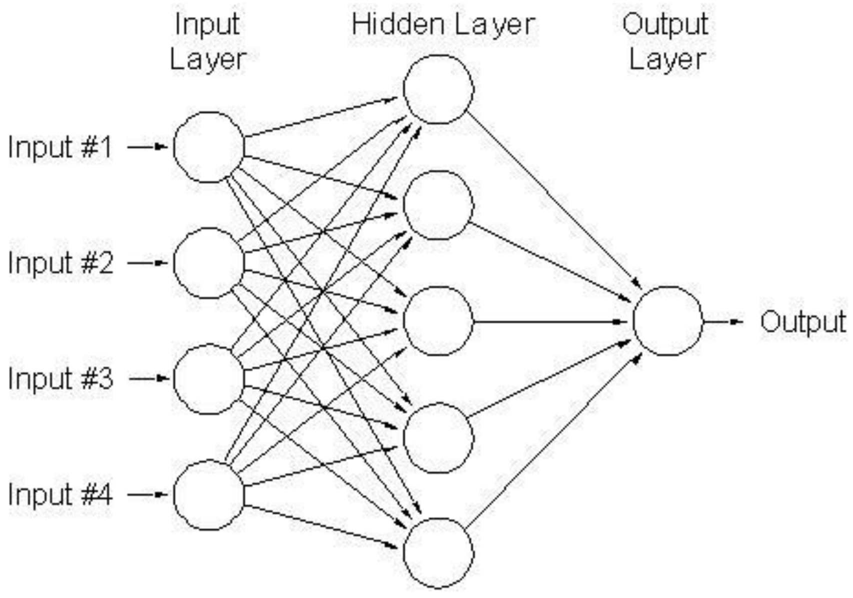


Figure 5.7. An Artificial Neural Network Structure.

The activation function of the artificial neurons in ANNs implementing the backpropagation algorithm is a weighted sum of the inputs $x_{i,}$ multiplied by their respective weights w_{ji} , as in equation 5.3 below [68]:

$$A_j\left(\overline{x}, \overline{w}\right) = \sum_{i=0}^{n} x_i w_{ji} \tag{5.3}$$

$$O_j\left(\overline{x}, \overline{w}\right) = \frac{1}{1 + e^{A_j\left(\overline{x}, \overline{w}\right)}} \tag{5.4}$$

Since the error is the difference between the actual and the desired output, the error depends on the weights, and it is needed to adjust the weights in order to minimize the error. The error function for the output of each neuron can be defined as in equation 5.5 as follows [68]:

$$E_j\left(\overline{x}, \overline{w}, d\right) = \left(O_j\left(\overline{x}, \overline{w}\right) - d_j\right)^2 \tag{5.5}$$

The backpropagation algorithm calculates how the error depends on the output, inputs, and weights as in equation 5.6 below [68]:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{5.6}$$

The adjustment of each weight (Δwji) will be the negative of a constant eta (η) multiplied by the dependence of the “wji” previous weight on the error of the network.

First, it is essential to calculate how much the error depends on the output as in the following equation [68]:

$$\frac{\partial E}{\partial O_j} = 2\left(O_j - d_j\right) \tag{5.7}$$

After that, the output value that depends on the activation, which in turn depends on the weights, is calculated by equation 5.8 as follows [68]:

$$\frac{\partial O_j}{\partial w_{ji}} = \frac{\partial O_j}{\partial A_j}\frac{\partial A_j}{\partial w_{ji}} = O_j(1-O_j)x_i \tag{5.8}$$

Therefore, the adjustment to each weight will be calculated as in the following equation [68]:

$$\Delta w_{ji} = -2\eta\left(O_j - d_j\right)O_j(1-O_j)x_i \tag{5.9}$$

In order to adjust v_{ik}, the weights of a previous layer, It is required to calculate how the error depends not on the weight, but on the input from the previous layer i.e. replacing w by x, as shown in equation 5.10 below [68]:

$$\Delta v_{ik} = -\eta\frac{\partial E}{\partial v_{ik}} = -\eta\frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial v_{ik}} \tag{5.10}$$

where

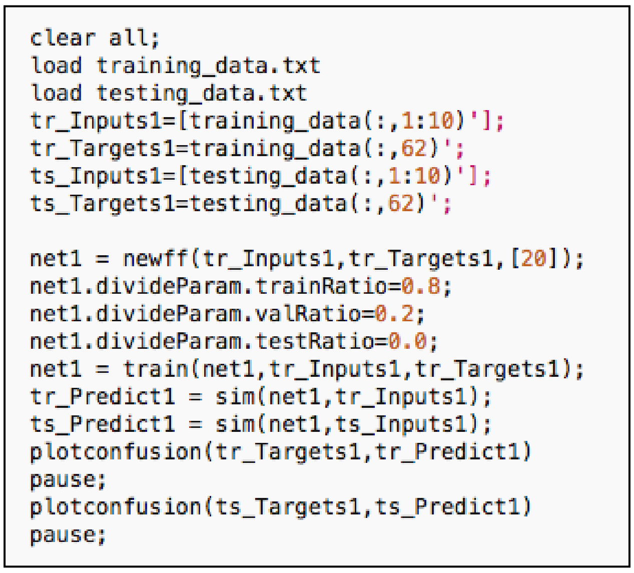
$$\frac{\partial E}{\partial w_{ji}} = 2\left(O_j - d_j\right)O_j(1-O_j)w_{ji} \tag{5.11}$$

and

$$\frac{\partial x_i}{\partial v_{ik}} = x_i(1-x_i)v_{ik} \tag{5.12}$$

5.3.3 Implementation

Both the ANN and SVM classification methods are implemented in MATLAB technical computing software, which have been used in developing and implementing many text mining applications [69]. As well, it includes a great amount of algorithms and methods that are highly used in text mining, which are already programmed as functions and toolboxes. The frequencies of all top terms for each article in the training and testing databases, representing 80% and 20% of the database respectively, were previously generated as an output to the Terms Frequency-based Classifier. After that, those frequencies are converted to a matrix database to be ready for classification in MATLAB, derived from MATrix LABoratory [69]. That is because it was originally developed as a programming language used to manipulate and operate with matrices. In Figure 5.7 it can be seen how the ANN algorithm is implemented to classify the articles of the first category as an example. The topology of the implemented ANN is a single hidden layer with 12 neurons, the activation function for the hidden layer is tangent-sigmoid and the output is linear, while the number of training epochs is 100. As well, it is shown where the files are processed and how the results are plotted.



```
clear all;
load training_data.txt
load testing_data.txt
tr_Inputs1=[training_data(:,1:10)'];
tr_Targets1=training_data(:,62)';
ts_Inputs1=[testing_data(:,1:10)'];
ts_Targets1=testing_data(:,62)';

net1 = newff(tr_Inputs1,tr_Targets1,[20]);
net1.divideParam.trainRatio=0.8;
net1.divideParam.valRatio=0.2;
net1.divideParam.testRatio=0.0;
net1 = train(net1,tr_Inputs1,tr_Targets1);
tr_Predict1 = sim(net1,tr_Inputs1);
ts_Predict1 = sim(net1,ts_Inputs1);
plotconfusion(tr_Targets1,tr_Predict1)
pause;
plotconfusion(ts_Targets1,ts_Predict1)
pause;
```

Figure 5.7. Instantiation of the ANN classification code.

5.3.4 Results

Following the implementation of the ANN classifier, documents of both training and testing databases are processed for classification. The execution time here was quite fast as that of the Terms Frequency-based classification, as it took about 15 to 20 minutes to classify all files. Plotted results for categories of both databases can be seen in Appendix A: at the end of this thesis. In Table 5.11, the final results of the ANN classification method are shown, which are discussed in more details in section 5.5 of this chapter.

Category		**Culture**	**Economy**	**International**	**Local**	**Religion**	**Sports**
Training Database	**Classification Accuracy**	91.7%	90.5%	91.5%	91.3%	91.5%	85.0%
Testing Database		88.7%	89.4%	90.8%	88.8%	89.9%	83.9%
Overall Results		90.2%	90%	91.2%	90.5%	90.7%	84.5%

Table 5.11. ANN Classification Results.

5.4 Support Vector Machine Classification

5.4.1 Methodology

SVM theory was developed by Vladimir Vapnik in 1995. It is considered as one of the most important breakthroughs in machine learning field and can be applied in classification and regression [70] [71] [72]. In modelling the SVM, the main goal is to select the optimal hyperplane in high dimensional space ensuring that the upper bound of generalization error is minimal. SVM can only directly deal with linear samples but mapping the original space into a higher dimensional space can make the analysis of nonlinear sample possible [73]. For example if the data point(xi, yi), was given randomly and independently generated from an unknown function, the approximate function form by SVM is as in equation (5.11) :

$$g(x) = w\emptyset(x) + b \qquad (5.11)$$

where $\emptyset(x)$ is the feature and nonlinear mapped from the input space x. w and b are both coefficients and can be estimated by minimizing the regularized risk function.

$$R(C) = C\frac{1}{N}\sum_{i=1}^{N} L(d_i, y_i) + \frac{1}{2}\|w\|^2 \qquad (5.12)$$

$$L(d, y) = \begin{cases} |d - y| - \varepsilon & |d - y| \geq \varepsilon, \\ 0 & otherwise, \end{cases} \qquad (5.13)$$

In equations (5.12) and (5.13) above, both C and ε are prescribed parameters. C is called the regularization constant, while ε is referred to as the regularization constant. $L(d, y)$ is the intensive loss function and the term $c\frac{1}{N}\sum_{i=1}^{N} L(d_i, y_i)$ is the empirical error, while $\frac{1}{2}\|w\|^2$ indicates the flatness of the function. The trade-off between the empirical risk and flatness of the model is measured by C. Since introducing positive slack variables ζ and $\zeta*$ the equation (5.13) transformed to equation (5.14) as the following:

$$R(w, \zeta, \zeta*) = \frac{1}{2} ww^T + C \times (\sum_{i=1}^{N}(\zeta, \zeta*)) \qquad (5.14)$$

Subject to:

$$w\emptyset(x_i) + b_i - d_i \leq \varepsilon + \zeta *_i \qquad (5.15)$$

$$d_i - w\emptyset(x_i) - b_i \leq \varepsilon + \zeta_i \qquad (5.16)$$

$$\zeta_i, \dot{\zeta_\iota} \geq 0 \qquad (5.17)$$

The decision function (kernel function) comes up finally after the Lagrange multipliers are introduced and optimality constraints exploited. The below function is the form of kernel function:

$$f(x) = \sum_i^l (\alpha_i - \dot{\alpha}_i) K(x_i, x_j) + b \qquad (5.18)$$

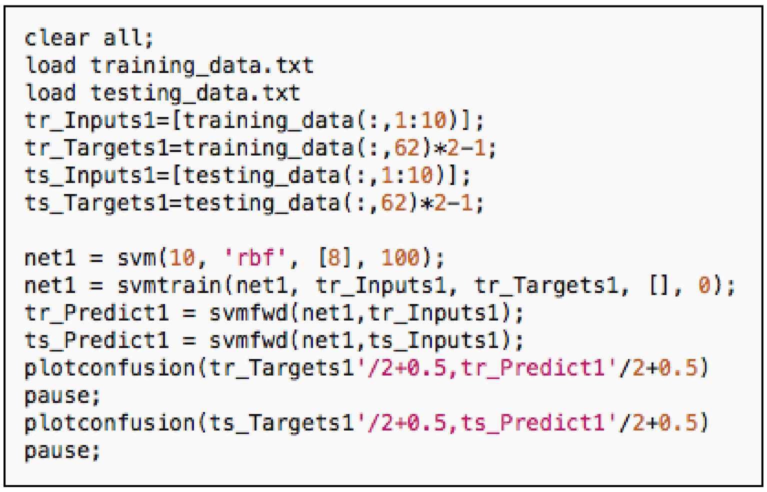
where $\dot{\alpha}_i$ are Lagrange multipliers. The satisfy the equalities $\alpha_i \times \dot{\alpha}_i = 0, \alpha_i \geq 0, \dot{\alpha}_i \geq 0$. The kernel value is the same with the inner product of two vectors x_i and x_j in the feature space $\emptyset(x_i)$ and $\emptyset(x_j)$. The most popular kernel function is Radial Basis Function (RBF) as in equation (5.19): .

$$K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right) \qquad (5.19)$$

Theoretical background, geometric interpretation, unique solution and mathematical tractability are the main advantages which has made SVM attract researchers and investors interest and be applied to many applications in different fields such prediction financial time series.

5.4.2 *Implementation*

As mentioned previously in section 5.3.3, the SVM classification method was developed and implemented in MATLAB computing software in which the matrix files of the training and testing databases, representing 80% and 20% of the database repsectly, are processed. In Figure 5.8, it is shown how the SVM classification is implemented to classify the articles of the first category for instance, where the kernel function is Radial Basis Function. In addition, it can be seen where the files are processed and how the results are plotted.



```
clear all;
load training_data.txt
load testing_data.txt
tr_Inputs1=[training_data(:,1:10)];
tr_Targets1=training_data(:,62)*2-1;
ts_Inputs1=[testing_data(:,1:10)];
ts_Targets1=testing_data(:,62)*2-1;

net1 = svm(10, 'rbf', [8], 100);
net1 = svmtrain(net1, tr_Inputs1, tr_Targets1, [], 0);
tr_Predict1 = svmfwd(net1,tr_Inputs1);
ts_Predict1 = svmfwd(net1,ts_Inputs1);
plotconfusion(tr_Targets1'/2+0.5,tr_Predict1'/2+0.5)
pause;
plotconfusion(ts_Targets1'/2+0.5,ts_Predict1'/2+0.5)
pause;
```

Figure 5.8. Instantiation of the SVM classification code.

5.4.3 *Results*

Subsequent to the implementation of the SVM classifier, documents of both training and testing databases are processed for classification. The execution time here was fairly slow compared to that of the Terms Frequency-based and ANN classification. It took about 3 to 4 hours to classify all files, and around 30 minutes to classify each category. Plotted results for categories of both databases can be seen in Appendix D: at the end of this thesis. In Table 5.12, the final results of the SVM classification method are shown, which are discussed further in section 5.5 of this chapter.

Category		Culture	Economy	International	Local	Religion	Sports
Training Database	Classification Accuracy	95.7%	95.1%	91.9%	95.0%	94.4%	96.5%
Testing Database		87.1%	88.0%	83.1%	89.3%	89.4%	86.7%
Overall Results		91.4%	91.6%	87.5%	92.2%	92%	91.6%

Table 5.12. SVM Classification Results.

5.5 Overall Results Discussion

Category		Culture	Economy	International	Local	Religion	Sports	Average Accuracy
TF-based Classification Accuracy	Training Database	58.2%	59.8%	73.7%	51.8%	61.8%	77.4%	63.78%
	Testing Database	60.6%	60.7%	75.7%	46.6%	79.7%	77.6%	66.82%
	Overall Results	59.4%	60.3%	74.7%	49.2%	70.8%	77.5%	65.32%
ANN Classification Accuracy	Training Database	91.7%	90.5%	91.5%	91.3%	91.5%	85.0%	90.25%
	Testing Database	88.7%	89.4%	90.8%	88.8%	89.9%	83.9%	88.58%
	Overall Results	90.2%	90%	91.2%	90.5%	90.7%	84.5%	89.42%
SVM Classification Accuracy	Training Database	95.7%	95.1%	91.9%	95.0%	94.4%	96.5%	94.77%
	Testing Database	87.1%	88.0%	83.1%	89.3%	89.4%	86.7%	87.27%
	Overall Results	91.4%	91.6%	87.5%	92.2%	92%	91.6%	91%

Table 5.13. Overall Results of each Classification Method.

Each identified classification method was tested over both the training and testing database, representing 80% and 20% correspondingly.

As it can be seen in Table 5.13, the Terms Frequency-based classification scores the least in both databases, of an average accuracy rate of 64% and 67% respectively. As well, the average accuracy of the testing database is found to be 3% higher than that of the training database. However, both the ANN and SVM classification methods show a high accuracy scores ranging between 85%-95%. Meanwhile, the average accuracy of the TF-based Classification is around 65%, which 20% less accurate.

In both the ANN and SVM classification methods, it is noted that the accuracy rate of the training database is higher than that of the testing database. However, the SVM classification of the training database achieves around 5% higher accuracy rate than that of the ANN classification method. Overall, SVM classification methods show superior performance over the TF-IDF and ANN classification methods, achieving an accuracy rate of 91%. Moreover, in a different study by Abbas, Smaili and Berkani [30], different classification methods including SVM were evaluated while using light stemming. In their study, SVM shows a superior result over the other methods and an accuracy of up to 97%. However, the database which was used for testing is different from the selected database in this research, thus it is not ideal to compare it with the SVM classification using root-based stemming which is applied here.

In addition to the results reported above, the classifiers were tested using the confusion matrix, the result are presented in Appendix C. The confusion matrix, also known as an error matrix is a specific table layout that allows visualization of the performance of a classification algorithm. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa).

The following example, given by Wikipedia, explains the concept. If a classification system has been trained to distinguish between cats and dogs, a confusion matrix will summarize the results of testing the algorithm for further inspection. Assuming a sample of 13 animals — 8 cats, and 6 dogs, the resulting confusion matrix could look like the Table 5.14.

		Predicted	
		Cat	Dog
Actual class	Cat	5	3
	Dog	2	4

Table 5.14. Example of Confusion Matrix Table.

In this confusion matrix, of the 8 actual cats, the system predicted that three were dogs, and of the 6 dogs, it predicted 4 correctly and and 2 were cats. 2 actual dogs were predicted as cats and 3 actual cats were predicted as dogs. We can see from the matrix that the system in question has trouble distinguishing between cats and dogs. All correct guesses are located in the diagonal of the table, so it's easy to visually inspect the table for errors, as they will be represented by values outside the diagonal.

5.6 Summary

In this chapter, a new approach to identify significant keywords for Arabic corpora was presented. The feature selection procedure, which is required to extract features for each topic, was accomplished by implementing the advanced root extraction algorithm, as well as the Term Frequency/Inverse Document Frequency (TF.IDF) topic identification method. Then, features of the database corpora representing six different categories were extracted.

Subsequently, selected features were set as vectors for each topic, and were employed as the main features for document classification. Both training and testing database, which was not included in feature selection, were processed through different classification methods. The classification methods introduced included the Terms Frequency-based classification method, the ANN method, and the SVM method. The Terms Frequency-based classification scored the least of an average accuracy rate of 64% and 67% respectively. Nevertheless, both the ANN and SVM classification methods show a high accuracy scores ranging between 85%-95%. Meanwhile, the average accuracy of the TF-based Classification is around 65%, which 20% less accurate.

Chapter 6

Conclusions and Future work

In this chapter, the stages of the work of this thesis are illustrated in brief, and a number of methods that may improve the system in the future are presented.

6.1 Conclusions

In this thesis, an improved root extraction algorithm for Arabic words which is based on morphological analysis and linguistic constraints was presented. The algorithm handles the problems of infixes removal by eliminating prefixes and suffixes while checking the word against a predefined list of patterns. The problem of extracting the roots of weak, hamzated, and eliminated-long-vowel words has been handled. As well as the two-letter geminated words, that is by identifying linguistic based rules to replace, eliminate or duplicate certain letters where needed. As well, a new approach to select feature keywords for Arabic corpora was introduced, and different classification methods utilizing the selected features and the root extraction method for text processing were evaluated. These include Terms Frequency-based method, Artificial Neural Networks (ANN), and Support Vector Machine (SVM).

6.1.1 The Root Extraction Algorithm

The root extraction algorithm is constructed following two main stages. The first stage is to process the text to remove affixes from the text, considering prefixes, suffixes and infixes. Simultaneously, the word is checked against its morphological pattern depending on its length after each deletion. If there was a match between the morphological pattern and the world, the root will be extracted. Else, the process continues to the second stage of the root extraction, where the algorithm is developed further to handle different types of words. The second stage includes two phases, where in the first, the cases of hamzated, weak, geminated and eliminated-long-vowel words are handled. In the last phase, words that are two letters long are handled. These words can either be geminated, eliminated-long-vowel or hamzated with an *alif* that is removed from imperative verbs like in (كُل) and (خُذ) of the roots (أكل) and (أخذ). After the roots are extracted, they are verified by checking them against a list of predefined roots, containing 3800 trilateral and 900 literal roots. The performance of the algorithm is then tested by performing a number of experiments.

The experiments and testing were conducted by using thousands of Arabic words gathered from an online Arabic corpus which is collected to aid Arabic language based research. Human judgment was applied to evaluate the results and accuracy of the algorithm, where the list of all words and their extracted roots was checked manually to check which roots are extracted correctly belonging to the original world, and which are not. The algorithm is introduced with the aim of supporting Arabic stemming/root extracting tools. The results obtained shows that the root extraction algorithm is promising and is worth being applied in various Arabic language processing programs.

6.1.2 Feature Selection and Classification

In addition, a new approach to identify significant keywords for Arabic corpora was presented. The feature selection procedure, which is required to extract features for each topic, was accomplished by implementing the advanced root extraction algorithm, as well as the Term Frequency/Inverse Document Frequency (TF.IDF) topic identification method. Then, features of the database corpora representing six different categories were extracted.

Subsequently, selected features were set as vectors for each topic, and were employed as the main features for document classification. The classification system is trained on six different categories: culture, economy, international, local, religion and sports. From each category the top ten frequent terms are selected as features. Both training and testing database, were processed through different classification methods. The classification methods introduced included the Terms Frequency-based classification method, the ANN method, and the SVM method. The Terms Frequency-based classification scored the least of an average accuracy rate of 64% and 67% respectively. The other two main non-parametric classifiers of Artificial Neural Networks (ANN) and Support Vector Machine (SVM) are tested where the system is trained on 80% of the collected database. Testing the classification methods has been done on the remaining 20% of the documents. The results of ANN and SVM are compared to the standard method used for text classification, the terms frequency-based method. The results obtained indicate that ANN and SVM have better accuracy, with scores ranging between 85-95%, compared to the standard TF-based classification method, with rates ranging between 64-67%. The proposed classification method proves the ability to categorize the Arabic text documents into the appropriate categories with a high precision rate while selecting the top features of each category and employing the root extraction algorithm in the text preprocessing stage.

6.2 Future Work

There are several possible methods and techniques that can be applied in the future to develop and improve what has been done in this thesis. In regards to the root extraction method, it can be improved by developing a friendly GUI interface which will allow the system to be used for different educational and research proposes. As well, the root extraction algorithm can be extended to involve type of word detection and Arabic text analysis such as tagging each part of the word to identify the genre of number of subjects, gender, and verbs tenses or types of nouns if the word was a noun. As well, the root extraction algorithm can be applied in various web mining and text tagging applications to highlight main topics of articles for example, or to support generating metadata for online resources.

In regards to the feature selection and classification methods, it could be improved further by selecting the top features of the topics manually, depending on the user's interest and personal view of the topic/topics. Selecting highly important terms of the topic as the topic's feature by the user, would enhance the documents classification outcome. Also, many classification algorithms can be implemented to employ the root-based stemming and root features, and be evaluated and improved accordingly.

References

[1] G. Kanaan, R. Al-Shalabi, S. Ghwanmeh and H. Al-Ma'adeed, “A Comparision of Text-Classification Techniques Applied to Arabic Text,” *Journal of the American Society for Information Science and Technology,* vol. 60, no. 9, pp. 1836-1844, 2009.

[2] G. Guo, H. Wang, D. Bell, Y. Bi and K. Greer, “A KNN model based approach and its application in text categorization,” in *Proceedings of the Computational Linguistics and Intelligent Text Processing, 5th International Conference*, Berlin, 2004.

[3] M. K. Saad and W. Ashour, “Arabic Morphological Tools for Text Mining,” in *Proceedings of the 6th International Conference on Electrical and Computer Systems (EECS'10),* Lefke, North Cyprus, 2010.

[4] A. Ghareb, A. R. Hamdan and A. Abu Bakar, “Text Associative Classification Approach for Mining Arabic Data Set,” in *Proceedings of the 4th Conference on Data Mining and Optimization (DMO)*, Langkawi, Malaysia, 2012.

[5] L. S. Larkey, L. Ballesteros and M. E. Connell, “Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis,” in *Proc. SIGIR '02*, Tampere, Finland, 2002.

[6] J. A. Haywood and H. M. Nahmad, A new Arabic grammar of written language, London: Lund Humphries, 1998.

[7] A. Alzamil, “The relationship between roots and patterns: new classification for Arabic language roots (العلاقة بين الجذور والأوزان الصرفية: تصنيف جديد لجذور اللغة العربية),” in *Proc. of the Int. Symposium on Computers & Arabic Language*, Riyadh, KSA, 2007.

[8] M. A. Otair, “Comparative Analysis of Arabic Stemming Algorithms,” *International Journal of Managing Information Technology (IJMIT),* vol. 5, no. 2, 2013.

[9] R. Alshalabi, “Pattern-based stemmer for finding Arabic roots,” *Information Technology,* vol. 4, no. 1, pp. 38-43, 2005.

[10] M. Y. Al-Nashashibi, D. Neagu and A. A. Yaghi, “An improved root extraction technique for Arabic words,” in *Proceedings of the 2nd International Conference on Computer Technology and Development (ICCTD 2010)*, Cairo, Egypt, 2010.

[11] Y. Kadri and J. Y. Nie, “Effective stemming for Arabic Information Retrieval,” in *Proc.*

of the International Conference at the British Computer Society, London, October, 2006.

[12] G. Kanaan, R. Al-Shalabi, M. Ababneh and A. Al-Nobani, “Building an effective rule-based light stemmer for Arabic language to improve search effectiveness,” in *International Conference on Innovations in Information Technology, IIT,* December, 2008.

[13] S. Ghwanmeh, R. Al-Shalabi, G. Kanaan and S. Rabab'ah, “Enhanced algorithm for extracting the root of Arabic words,” in *Proc. of the 6th IEEE International Conference on Computer Graphics and Visualization*, 2009.

[14] A. Brahmi, A. Ech-Cherif and A. Benyettou, “Arabic texts analysis for topic modeling evaluation,” *Information Retrieval,* vol. 15, no. 1, pp. 33-53, 2012.

[15] R. Al-Shalabi, G. Kanaan and H. Muaidi, “New approach for extracting Arabic roots,” Alexandria, Egypt, 2003.

[16] H. Al-Serhan and A. Ayesh, “A triliteral word roots extraction using Neural Network for Arabic,” in *Proceedings of the 2006 International Conference on Computer Engineering and Systems,* November, 2006.

[17] S. Khoja, “Stemming Arabic Text,” Lancaster University, Lancaster, U.K., 1999.

[18] M. Swalha and E. Atwell, “Comparative evaluation of Arabic language morphological analyzers and stemmers,” in *Proceedings of the 22nd International Conference on Computational Linguistics (COLING),* 2008.

[19] E. M. Saad, M. H. Awadalla and A. Alajmi, “Arabic Verb Pattern Extraction,” in *Proceedings of the 10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010),* Kuala Lumpur, Malaysia, 2010.

[20] H. Al-Ameed, “A proposed new model using a light stemmer for increasing the success of search in Arabic terms,” PhD Thesis, University of Bradford, Bradford, UK, 2006.

[21] M. A. B. Ar-Rhazi, Mukhtar us-Sihah, Beirut: Librairie du Liban, 1986.

[22] H. Bayyomee, K. Kolfat and A. Al-Shafe'e, Lexicon for Arabic Verbs Morphology, Cairo: Dar Ilias Modern Publishing Company, 1989.

[23] K. Darwish and D. W. Oward, “Adapting Morphology for Arabic Information Retrieval,” in *Arabic Computational Morphology*, Springer, 2007, pp. 245-262.

[24] K. Beesley, “Arabic Finite-State Morphological Analysis and Generation,” in *Proceedings of the International Conference on Computational Linguistics (COLING-96)*, 1996.

[25] K. Beesley, T. Buckwalter and S. Newton, “Two-Level Finite-State Analysis of Arabic Morphology,” in *Proceedings in the Seminar on Bilingual Computing in Arabic and English*, Cambridge, UK, 1989.

[26] G. Kiraz, “Arabic Computational Morphology in the West,” in *Proceedingd of the 6th International Conference an Exhibition on Multi-lingual Computing*, Cambridge, UK, 1998.

[27] A. Abdul-Al-Aal, An-Nahw Ashamil (النحو الشامل), Cairo: Maktabat Annahda Al-Masriya, 1987.

[28] K. Darwish, “Building a Shallow Arabic Morphological Analyzer in One Day,” in *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Philadelphia, PA, 2002.

[29] M. Abbas and K. Smaili, “Comparison of Topic Identification Methods for Arabic Language,” in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2005.

[30] M. Abbas, K. Smaili and D. Berkani, “Evaluation of Topic Identification Methods on Arabic Corpora,” *Journal of Digital Information Management*, vol. 9, no. 5, pp. 185-192, October 2011.

[31] M. Antoine and O. Zaiane, “Text document categorization by term association,” in *Proceedings of the IEEE International Conference on Data Mining (ICDM '02)*, Maebashi City, Japan, 2002.

[32] S. Khoja and R. Garside, “Stemming Arabic text,” Lancaster, UK, 1999.

[33] K. Brustad, M. Al Batal and A. Al Tonsi, Alif Baa: Introduction to Arabic Letters and Sounds, Georgetown University Press, 2010.

[34] J. Mace, Beginner's Arabic Script, McGraw-Hill/Contemporary, 2000.

[35] A. Al-Rajhi, التطبيق الصرفي, Dar Alnahdhah Alarabiyah, 2004.

[36] S. Eldin, “Development of a computer-based Arabic lexicon,” in *Proceedings of the International Symposium on Computers and Arabic Language (ISCAL)*, Riyadh, KSA,

2007.

[37] R. Duwairi, M. Al-Refai and N. Khasawneh, “Feature Reduction Techniques for Arabic Text Categorization,” *Journal of the American Society for Information Science and Technology,* vol. 60, no. 11, p. 2347–2352, 2009.

[38] A. Ibn Manẓūr, Lisān Al-‘arab, Cairo, Egypt: Matba‘at Muṣṭafa al-Baabī al-Ḥalabī, 1988.

[39] J. A.-D. As-Suyuti, Al-’ašbāh wa An-naẓā’ir fī An-naḥw, T. ‘. Saad, Ed., Cairo, Egypt: Maktabaht al-Kulliyyāt al-’Azhariyyah, 1975.

[40] A. a.-Q. Az-Zajjājī, Al-jumal, M. Ibn ’Abī Šanab, Ed., Beirut, Lebanon: Dār al-Kutub al-‘Ilmiyyah, 1986.

[41] A. Ibn al-’Anbārī, Al-’inṣāf fī Masā’il il-xilāf bayna An-naḥawiyyīn, M. M. Jawdah, Ed., Cairo, Egypt: Maktabaht al-Khanji, 2002.

[42] U. A. Ibn Jinni, Al-xaṣā’iṣ, M. An-Najjār, Ed., Cairo, Egypt: Dār al-Kutub al-Miṣriyyah, 1952.

[43] A. A. Wafi, Fiqh Allughah, Egypt: Nahdhat Misr Establishment, 2004.

[44] A.-f. Abaadi, Al-qamous Almuheet, 8 ed., Beirut, Lebanon: Al-arisaalah Establishment for Printing, Publishing and Distripution, 2005.

[45] I. Anees, Min Asrar Allughah, Egypt: Maktabat Al-Anjlo Al-Misriyyah, 1978.

[46] A.-X. A. Al-Farahidi, Muxtṣar Kitāb Al-‘ayn, H. H. Hammūdi, Ed., Masqat, Oman: Al-Maktabah aš-Šarqiyyah, 1998.

[47] A. Ameen, Al-Ishtiqaq fi Allughati Al-Arabiyyah, Cairo, Egypt: The Translation and Publishing Committee, 1956.

[48] A. Matloob, Annaht Fi Al-Lughat Al-Arabiyyah, Lebanon: Maktabat Lubnan, 2002.

[49] A. Bin Faris, Mu’jam Maqayees Al-Lughah, A. Haroon, Ed., Cairo, Egypt: Dar Al-Fikr, 1979.

[50] M. Al-batal, Alkitab fi Ta’allum al-‘Arabiyyah, George town University, 2004.

[51] L. Al-Sulaiti, “Latifa Al-Sulaiti's Website: Arabic Online Corpus,” 2009.

[52] A. Alsaad and M. Abbod, “Arabic Text Root Extraction via Morphological Analysis and Linguistic Constraints,” in *Proceedings of the 16th UKSim-AMSS International Conference on Computer Modelling and Simulation*, Cambridge, UK, 2014.

[53] M. McCandless, E. Hatcher and O. Gospodnetić, Lucene in Action, 2 ed., Manning, 2010.

[54] S. Osiński and D. Weiss, “Carrot2: Design of a Flexible and Efficient Web Information Retrieval Framework,” in *Proceedings of the third International Atlantic Web Intelligence Conference (AWIC 2005)*, Łodź, Poland, 2005.

[55] S. Demir, E. A. Sezer and H. Sever, “Modifications for the Cluster Content Discovery and the Cluster Label Induction Phases of the Lingo Algorithm,” *International Journal of Computer Theory and Engineering*, vol. 6, no. 2, pp. 86-90, 2014.

[56] C. D. Manning, P. Raghavan and S. Hinrich, Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2008.

[57] T. Joachims, “A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization,” Pittsburgh, 1996.

[58] G. Salton, Developments in Automatic Text Retrieval, vol. Science 253, 1991, pp. 974-979.

[59] S. Al-Harbi, A. Almuhareb, A. Al-Thubaity, M. S. Khorsheed and A. Al-Rajeh, “Automatic Arabic Text Classification,” in *9es Journees Internationales d'Analyse Statistique des Donnees Textuelles (JADT)*, Lyon, France, 2009.

[60] M. El-Kourdi, A. Bensaid and T. Rachidi, “Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm,” in *Proceedings of the Workshop on Computational Approaches to Araboc Script-based Languages*, Geneve, Switzerland, 2004.

[61] H. Sawaf, J. Zaplo and H. Ney, “Statistical Classification Methods for Arabic News Articles,” in *Proceedings of the Arabic Natural Language Processing in ALC 2001*, Toulouse, France, 2001.

[62] A. El-Halees, “Mining Arabic Association Rules for Text Classification,” in *Proceedings of the 1st International Conference of Mathematical Sciences*, Al-Azhar University of Gaza, Palestine, 2006.

[63] M. M. Syiam, Z. T. Fayed and M. B. Habib, “An Intelligent System for Arabic Text Categorization,” in *IJICIS 6*, 2006.

[64] R. Duwairi, M. Al-Refai and N. Khasawneh, “Stemming Versus Light Stemming as

Feature Selection Techniques for Arabic Text Categorization," in *Proceedings of the 4th International Conference on Innovations in Information Technology*, Dubai, UAE, 2007.

[65] R. Moraes, J. F. Valiati and W. P. G. Neto, "Document-level sentiment classification: An empirical comparison between SVM and ANN," *Expert Systems with Applications International Journal*, vol. 40, no. 2, p. 621–633, 2013.

[66] N. Sharma and T. Gedeon, "Artificial Neural Network Classification Models for Stress in Reading," in *Neural Information Processing*, Berlin, Springer Berlin Heidelberg, 2012, pp. 388-395.

[67] Jian, Mao and Mohiuddin, Neural Networks: A tutorials, IEEE, 2016.

[68] R. Rojas, Neural Networks: A Systematic Introduction, Berlin: Springer, 1996.

[69] R. E. Banchs, Text Mining with MATLAB, 1 ed., New York: Springer-Verlag New York, 2013.

[70] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learning,* vol. 20, pp. 273-297, 1995.

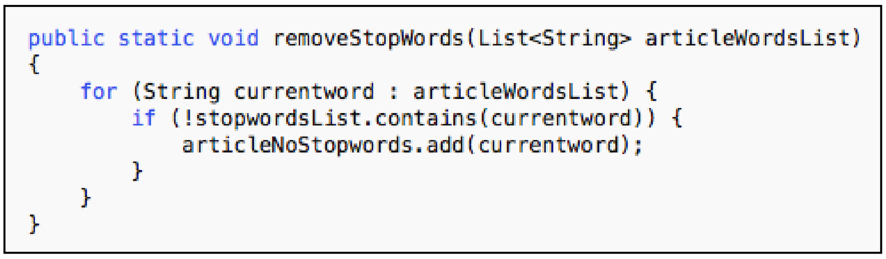
[71] J. Shawe-Taylor and N. Cristianini, Support Vector Machines and other kernel-based learning methods, Cambridge: Cambridge University Press, 2000.

[72] Y. Bi, S. Kapoor and R. Bhatia, Intelligent Systems and Applications: Extended and Selected Results from the SAI Intelligent Systems Conference (IntelliSys) 2015., Switzerland: Springer International Publishing, 2016.

[73] M. Pontil and A. Verri, "Properties of Support Vector Machines," *Neural Comput.,* vol. 10, pp. 955-974, 1998.

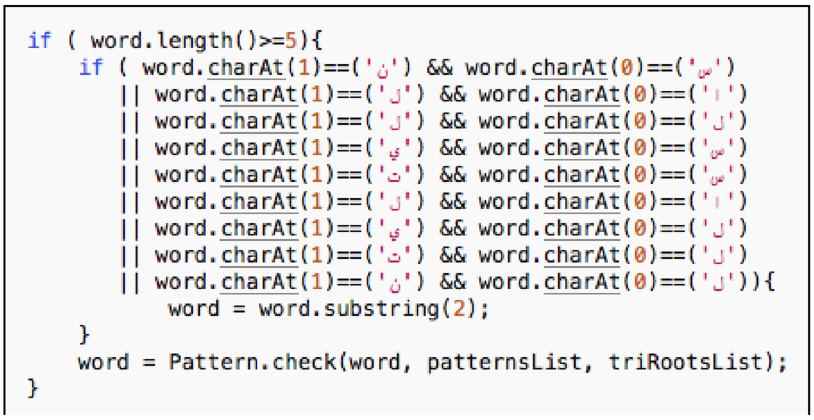
[74] K. Seymore and R. Rosenfeld, "Using Story Topics for Language Model Adaptation," in *Proceeding of the European Conference on Speech Communication and Technology*, 1997.

Appendix A: Java Implementation Codes



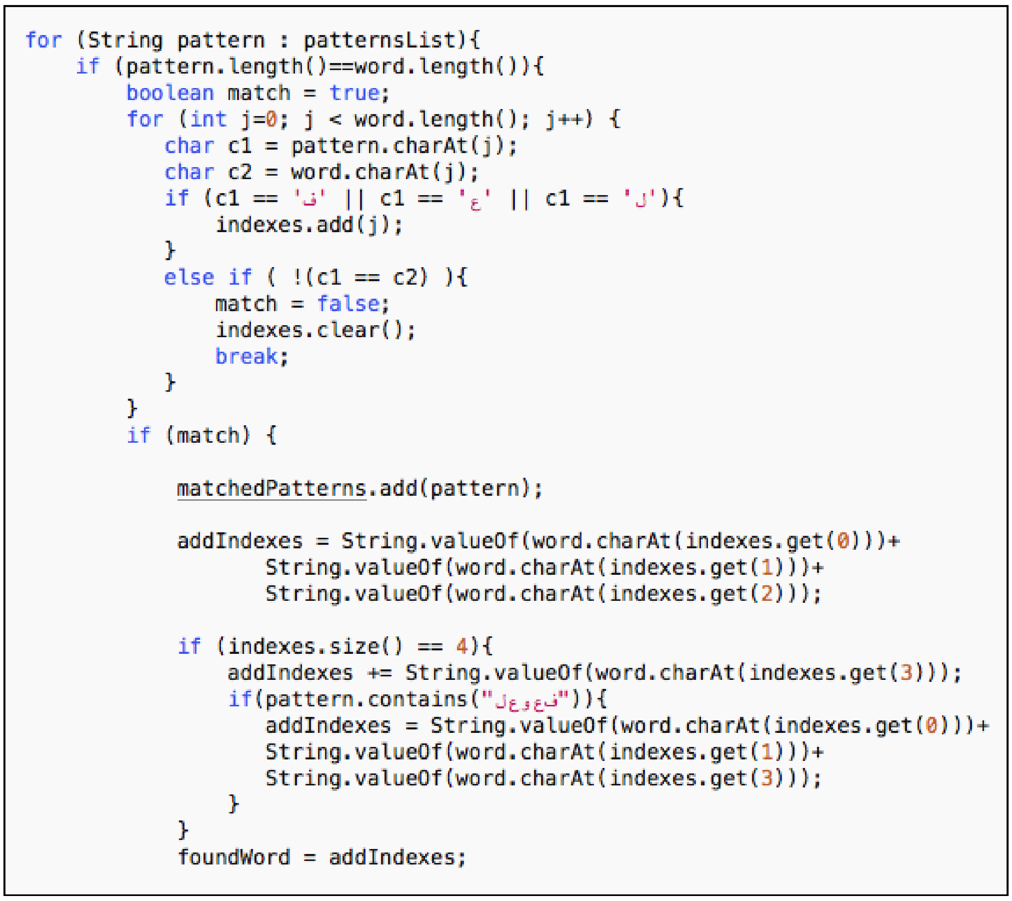
```java
public static void removeStopWords(List<String> articleWordsList)
{
    for (String currentword : articleWordsList) {
        if (!stopwordsList.contains(currentword)) {
            articleNoStopwords.add(currentword);
        }
    }
}
```

Figure A.1. Instantiation of the removeStopWords() Function.



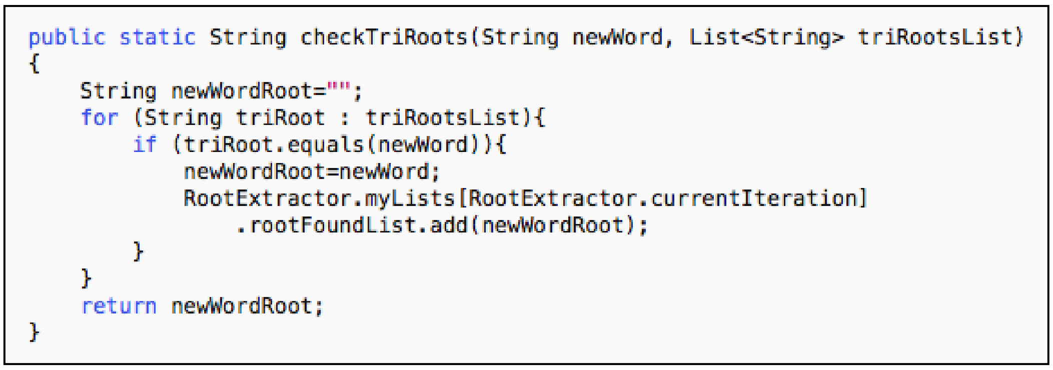
```java
if ( word.length()>=5){
    if ( word.charAt(1)==('ن') && word.charAt(0)==('س')
       || word.charAt(1)==('ل') && word.charAt(0)==('ا')
       || word.charAt(1)==('ل') && word.charAt(0)==('ل')
       || word.charAt(1)==('ي') && word.charAt(0)==('س')
       || word.charAt(1)==('ت') && word.charAt(0)==('س')
       || word.charAt(1)==('ل') && word.charAt(0)==('ا')
       || word.charAt(1)==('ي') && word.charAt(0)==('ل')
       || word.charAt(1)==('ت') && word.charAt(0)==('ل')
       || word.charAt(1)==('ن') && word.charAt(0)==('ل')){
         word = word.substring(2);
    }
    word = Pattern.check(word, patternsList, triRootsList);
}
```

Figure A.2. Code Implementation to Remove Suffixes for 5 Lettered Words.



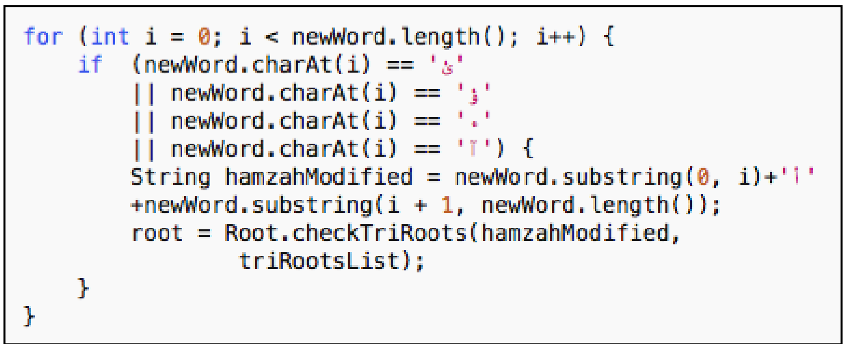
```java
for (String pattern : patternsList){
    if (pattern.length()==word.length()){
        boolean match = true;
        for (int j=0; j < word.length(); j++) {
           char c1 = pattern.charAt(j);
           char c2 = word.charAt(j);
           if (c1 == 'ف' || c1 == 'ع' || c1 == 'ل'){
               indexes.add(j);
           }
           else if ( !(c1 == c2) ){
               match = false;
               indexes.clear();
               break;
           }
        }
        if (match) {

            matchedPatterns.add(pattern);

            addIndexes = String.valueOf(word.charAt(indexes.get(0)))+
                   String.valueOf(word.charAt(indexes.get(1)))+
                   String.valueOf(word.charAt(indexes.get(2)));

            if (indexes.size() == 4){
                addIndexes += String.valueOf(word.charAt(indexes.get(3)));
                if(pattern.contains("فعوعل")){
                   addIndexes = String.valueOf(word.charAt(indexes.get(0)))+
                   String.valueOf(word.charAt(indexes.get(1)))+
                   String.valueOf(word.charAt(indexes.get(3)));
                }
            }
            foundWord = addIndexes;
```

Figure A.3. Implementation of the Pattern Matching Process.



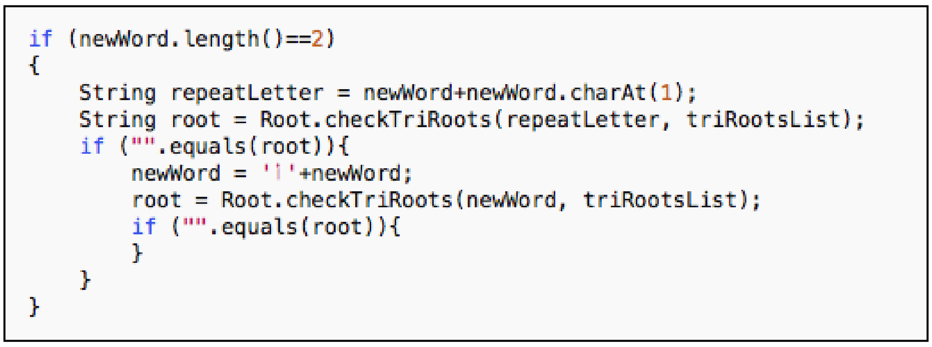
```java
public static String checkTriRoots(String newWord, List<String> triRootsList)
{
    String newWordRoot="";
    for (String triRoot : triRootsList){
        if (triRoot.equals(newWord)){
            newWordRoot=newWord;
            RootExtractor.myLists[RootExtractor.currentIteration]
                .rootFoundList.add(newWordRoot);
        }
    }
    return newWordRoot;
}
```

Figure A.4. Implementation of the checkTriRoots() Function.



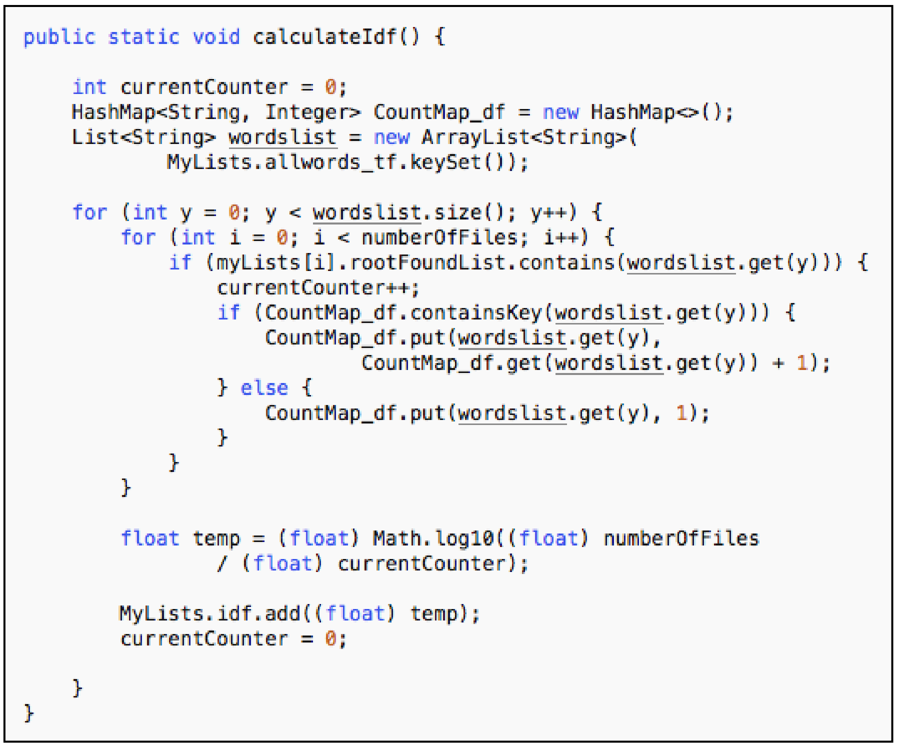
```java
for (int i = 0; i < newWord.length(); i++) {
    if  (newWord.charAt(i) == 'ئ'
        || newWord.charAt(i) == 'ؤ'
        || newWord.charAt(i) == 'ء'
        || newWord.charAt(i) == 'آ') {
        String hamzahModified = newWord.substring(0, i)+'أ'
        +newWord.substring(i + 1, newWord.length());
        root = Root.checkTriRoots(hamzahModified,
                triRootsList);
    }
}
```

Figure A.5. Code Implementation for Handling Hamzated Words.



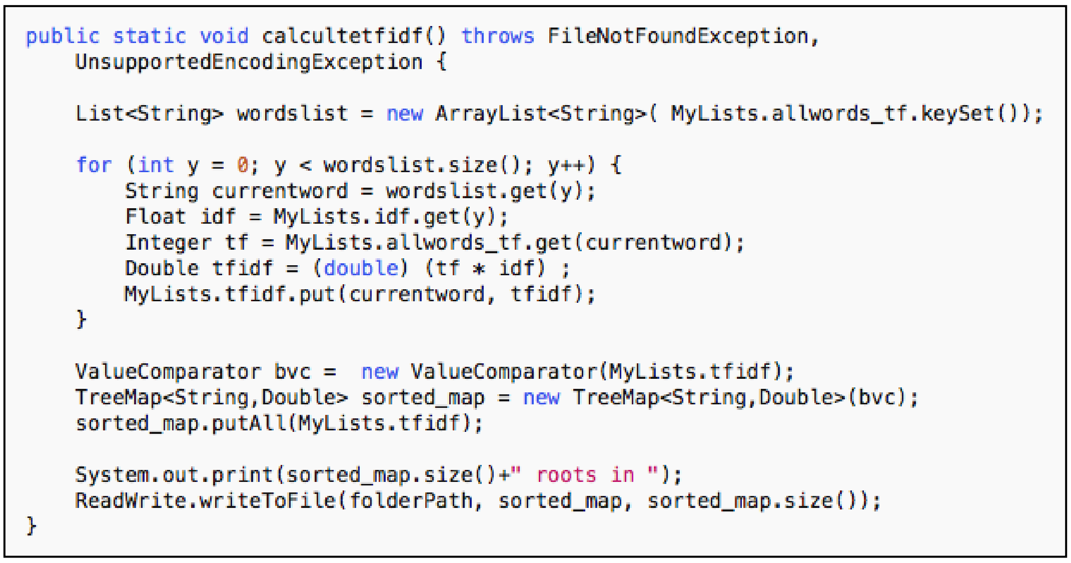
```java
if (newWord.length()==2)
{
    String repeatLetter = newWord+newWord.charAt(1);
    String root = Root.checkTriRoots(repeatLetter, triRootsList);
    if ("".equals(root)){
        newWord = 'أ'+newWord;
        root = Root.checkTriRoots(newWord, triRootsList);
        if ("".equals(root)){
        }
    }
}
```

Figure A.6. Code Implementation to Handle Geminated and *alif* Eliminated Words.



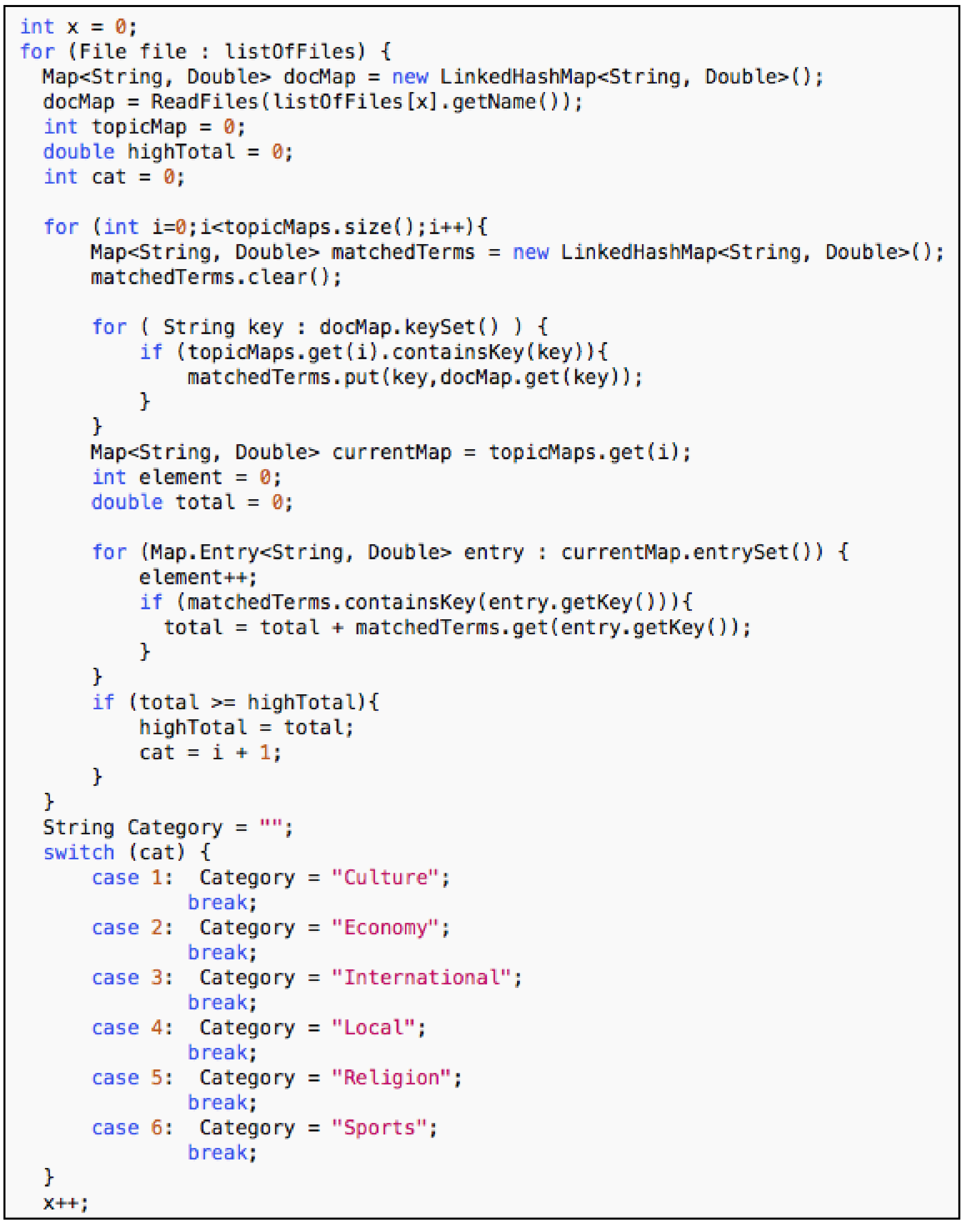
```java
public static void calculateIdf() {

    int currentCounter = 0;
    HashMap<String, Integer> CountMap_df = new HashMap<>();
    List<String> wordslist = new ArrayList<String>(
            MyLists.allwords_tf.keySet());

    for (int y = 0; y < wordslist.size(); y++) {
        for (int i = 0; i < numberOfFiles; i++) {
            if (myLists[i].rootFoundList.contains(wordslist.get(y))) {
                currentCounter++;
                if (CountMap_df.containsKey(wordslist.get(y))) {
                    CountMap_df.put(wordslist.get(y),
                            CountMap_df.get(wordslist.get(y)) + 1);
                } else {
                    CountMap_df.put(wordslist.get(y), 1);
                }
            }
        }

        float temp = (float) Math.log10((float) numberOfFiles
                / (float) currentCounter);

        MyLists.idf.add((float) temp);
        currentCounter = 0;

    }
}
```

Figure A.7. Implementation of the calculateIdf() function.



```java
public static void calcultetfidf() throws FileNotFoundException,
    UnsupportedEncodingException {

    List<String> wordslist = new ArrayList<String>( MyLists.allwords_tf.keySet());

    for (int y = 0; y < wordslist.size(); y++) {
        String currentword = wordslist.get(y);
        Float idf = MyLists.idf.get(y);
        Integer tf = MyLists.allwords_tf.get(currentword);
        Double tfidf = (double) (tf * idf) ;
        MyLists.tfidf.put(currentword, tfidf);
    }

    ValueComparator bvc =  new ValueComparator(MyLists.tfidf);
    TreeMap<String,Double> sorted_map = new TreeMap<String,Double>(bvc);
    sorted_map.putAll(MyLists.tfidf);

    System.out.print(sorted_map.size()+" roots in ");
    ReadWrite.writeToFile(folderPath, sorted_map, sorted_map.size());
}
```

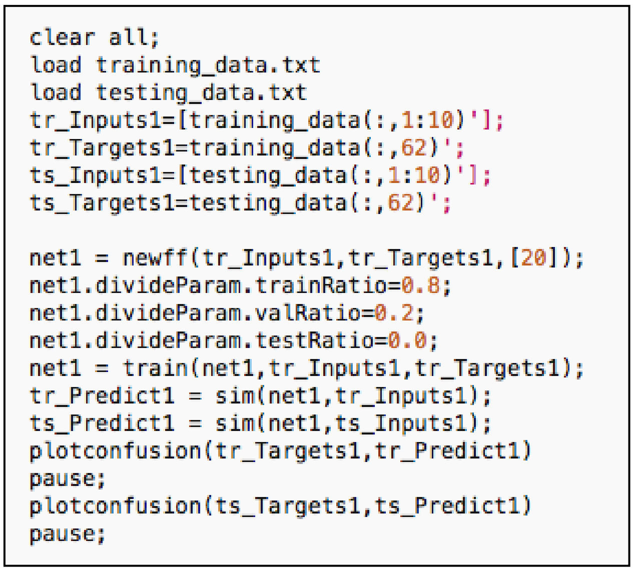
Figure A.8. Implementation of the calculate tfidf() function.



```java
int x = 0;
for (File file : listOfFiles) {
  Map<String, Double> docMap = new LinkedHashMap<String, Double>();
  docMap = ReadFiles(listOfFiles[x].getName());
  int topicMap = 0;
  double highTotal = 0;
  int cat = 0;

  for (int i=0;i<topicMaps.size();i++){
      Map<String, Double> matchedTerms = new LinkedHashMap<String, Double>();
      matchedTerms.clear();

      for ( String key : docMap.keySet() ) {
          if (topicMaps.get(i).containsKey(key)){
              matchedTerms.put(key,docMap.get(key));
          }
      }
      Map<String, Double> currentMap = topicMaps.get(i);
      int element = 0;
      double total = 0;

      for (Map.Entry<String, Double> entry : currentMap.entrySet()) {
          element++;
          if (matchedTerms.containsKey(entry.getKey())){
            total = total + matchedTerms.get(entry.getKey());
          }
      }
      if (total >= highTotal){
          highTotal = total;
          cat = i + 1;
      }
  }
  String Category = "";
  switch (cat) {
      case 1:  Category = "Culture";
               break;
      case 2:  Category = "Economy";
               break;
      case 3:  Category = "International";
               break;
      case 4:  Category = "Local";
               break;
      case 5:  Category = "Religion";
               break;
      case 6:  Category = "Sports";
               break;
  }
  x++;
```

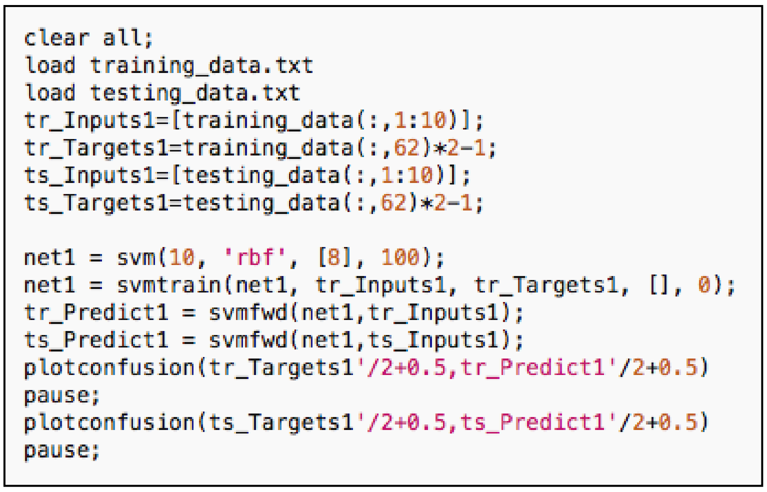
Figure A.9. Implementation of the Terms Frequency-based Classifier.

Appendix B: Matlab Implementation Codes



```
clear all;
load training_data.txt
load testing_data.txt
tr_Inputs1=[training_data(:,1:10)'];
tr_Targets1=training_data(:,62)';
ts_Inputs1=[testing_data(:,1:10)'];
ts_Targets1=testing_data(:,62)';

net1 = newff(tr_Inputs1,tr_Targets1,[20]);
net1.divideParam.trainRatio=0.8;
net1.divideParam.valRatio=0.2;
net1.divideParam.testRatio=0.0;
net1 = train(net1,tr_Inputs1,tr_Targets1);
tr_Predict1 = sim(net1,tr_Inputs1);
ts_Predict1 = sim(net1,ts_Inputs1);
plotconfusion(tr_Targets1,tr_Predict1)
pause;
plotconfusion(ts_Targets1,ts_Predict1)
pause;
```

Figure B.1. Instantiation of the ANN classification code.



```
clear all;
load training_data.txt
load testing_data.txt
tr_Inputs1=[training_data(:,1:10)];
tr_Targets1=training_data(:,62)*2-1;
ts_Inputs1=[testing_data(:,1:10)];
ts_Targets1=testing_data(:,62)*2-1;

net1 = svm(10, 'rbf', [8], 100);
net1 = svmtrain(net1, tr_Inputs1, tr_Targets1, [], 0);
tr_Predict1 = svmfwd(net1,tr_Inputs1);
ts_Predict1 = svmfwd(net1,ts_Inputs1);
plotconfusion(tr_Targets1'/2+0.5,tr_Predict1'/2+0.5)
pause;
plotconfusion(ts_Targets1'/2+0.5,ts_Predict1'/2+0.5)
pause;
```

Figure B.2. Instantiation of the SVM classification code.

Appendix C: ANN Classification Results

C.1 Training Database

In the confusion matrix presented in Figure C.1 below, the first two diagonal cells show the number and percentage of correct classifications by the trained network. It can be seen that 13261 of the documents which do not belong to the Culture category are classified as not related. On the other hand, 1620 of the documents representing 10% of the database which belong to the Culture category are classified as related. That gives an accuracy rate of 91.7% as shown in the bottom right cell of the matrix. However, the other two diagonal cells, the number and percentage of incorrect classification is shown, giving inaccuracy percentage of 8.3%.

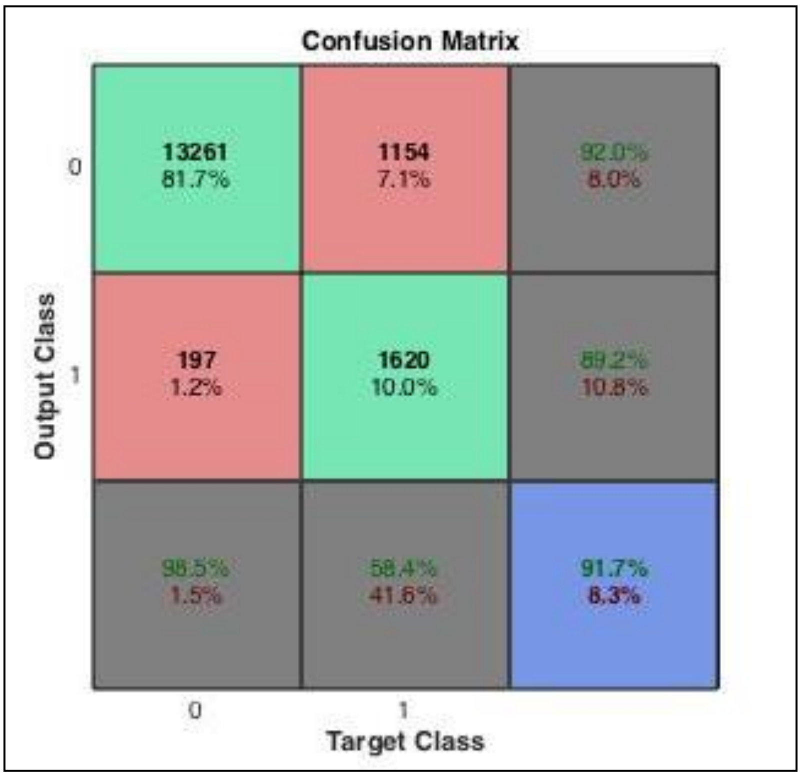


Figure C.1. ANN Classification Results of the 'Culture' Category Training Database.

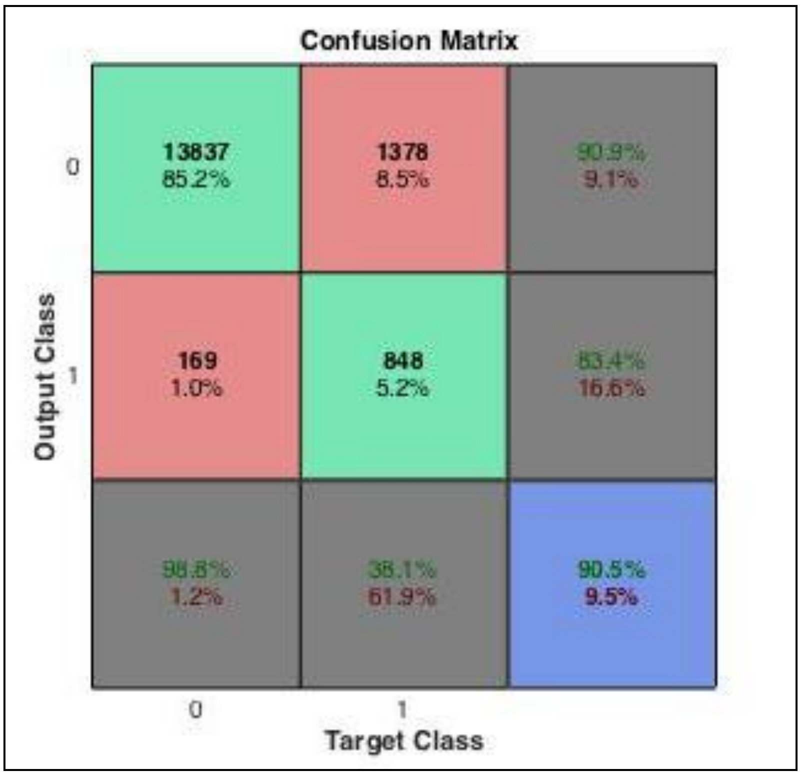


Figure C.2. ANN Classification Results of the 'Economy'

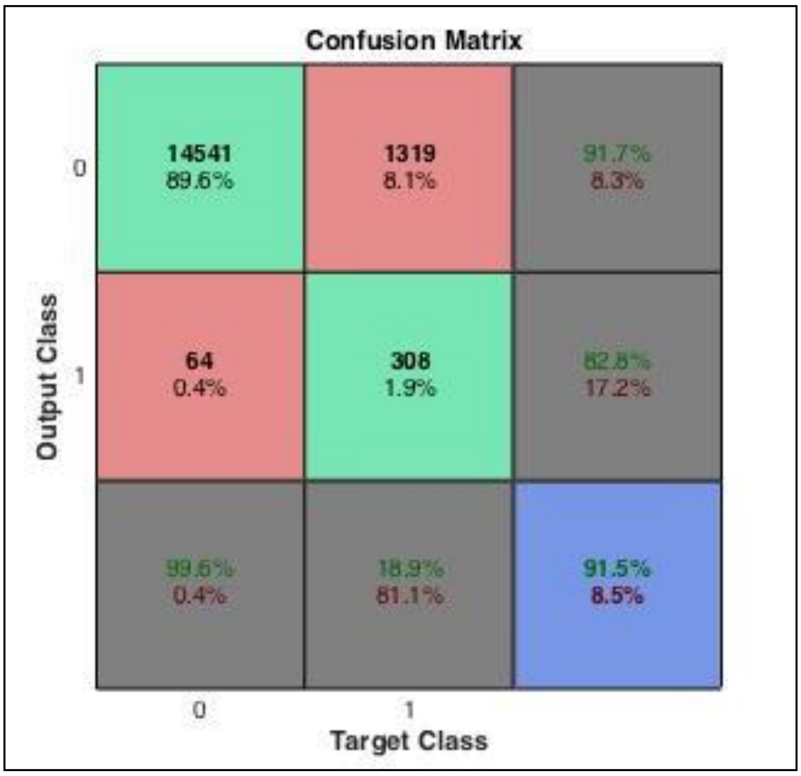


Figure C.3. ANN Classification Results of the 'International' Category Training Database.

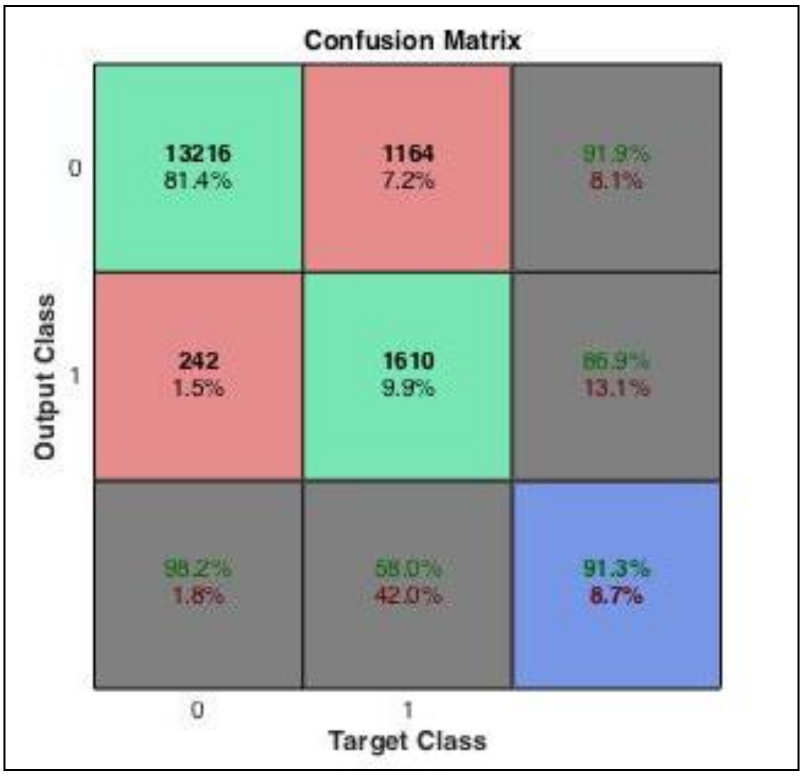


Figure C.4. ANN Classification Results of the 'Local' Category Training Database.

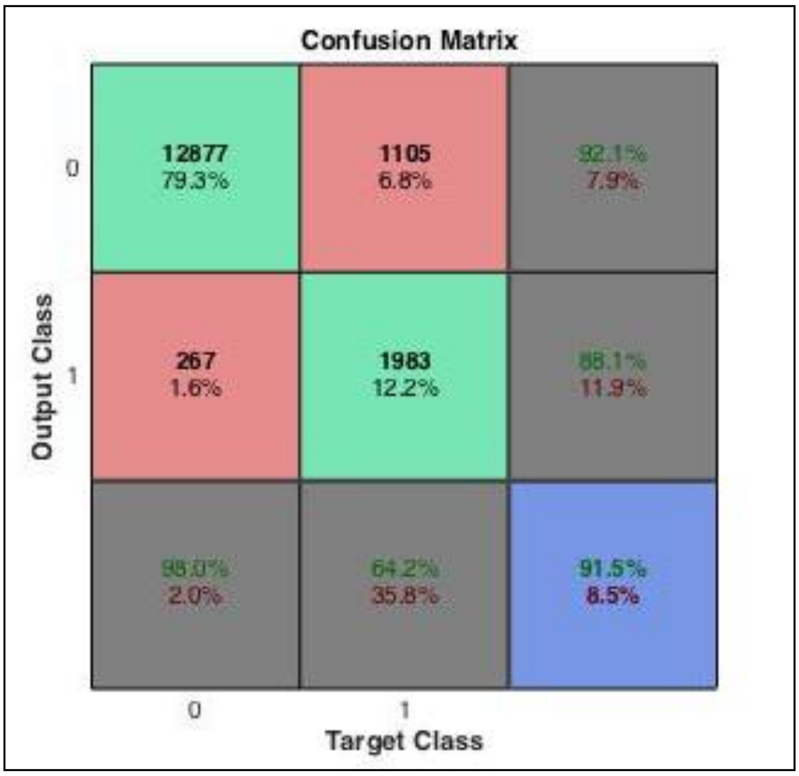


Figure C.5. ANN Classification Results of the 'Religion' Category Training Database.

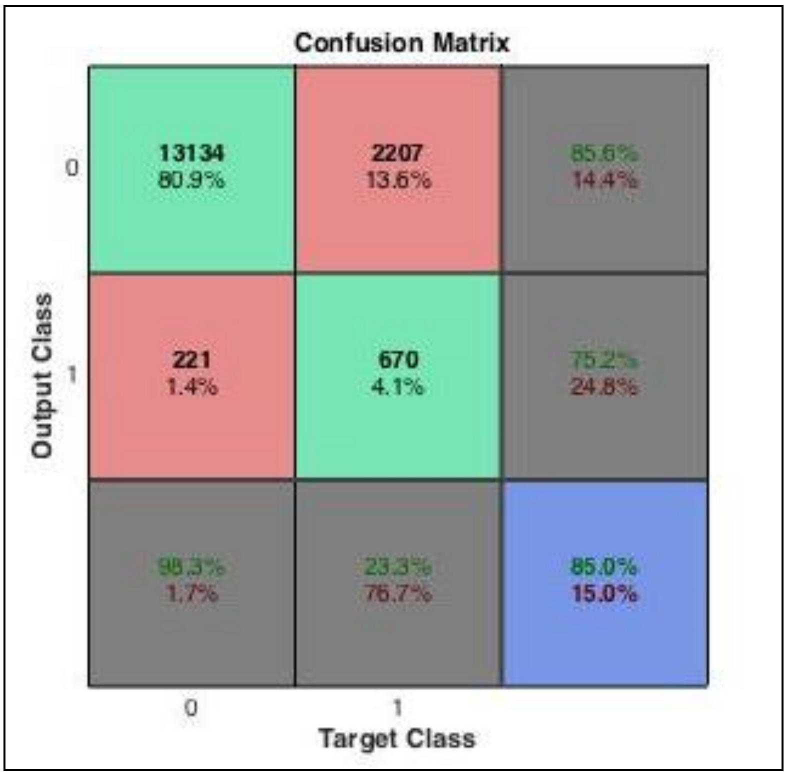


Figure C.6. ANN Classification Results of the 'Sports' Category Training Database.

C.2 Testing Database

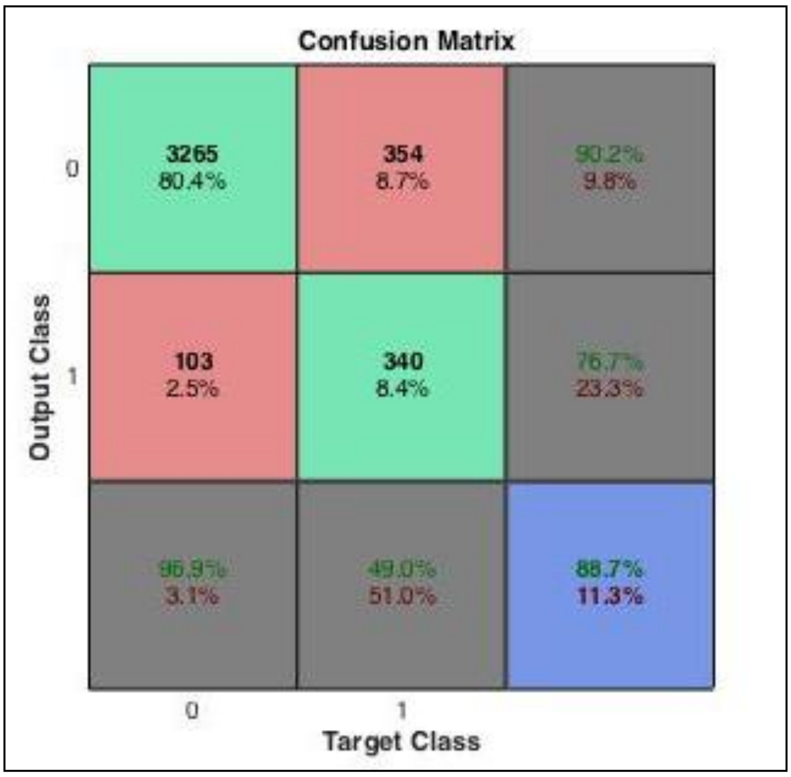


Figure C.7. ANN Classification Results of the 'Culture' Category Testing Database.

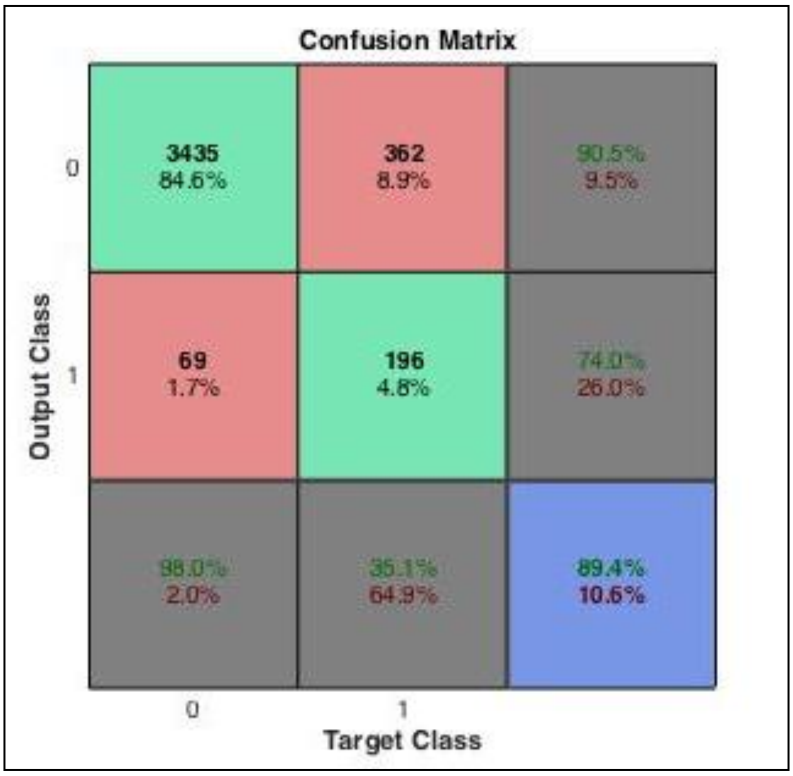


Figure C.8. ANN Classification Results of the 'Economy' Category Testing Database.

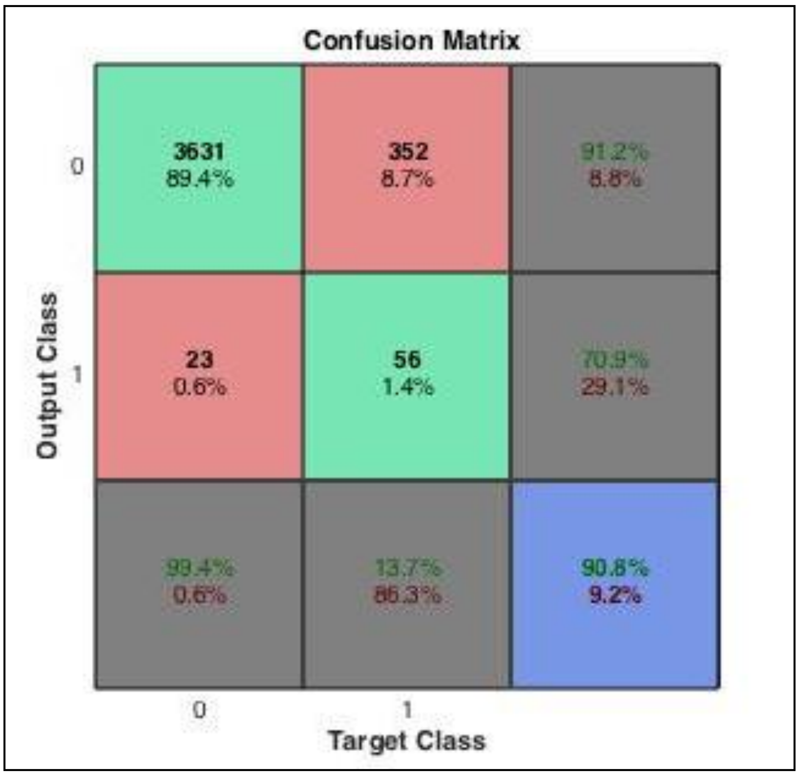


Figure C.9. ANN Classification Results of the 'International' Category Testing Database.

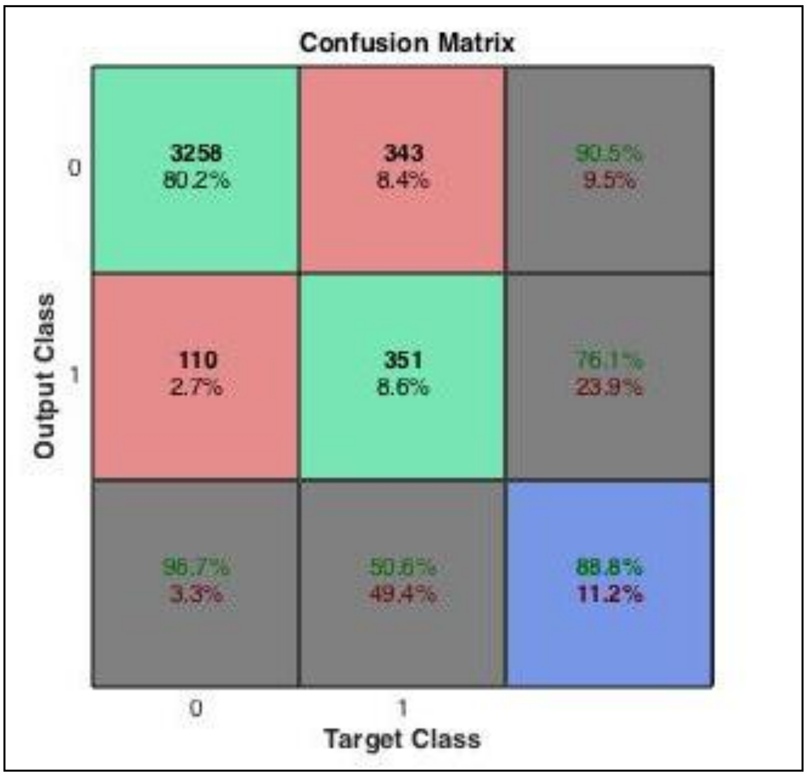


Figure C.10. ANN Classification Results of the 'Local' Category Testing Database.

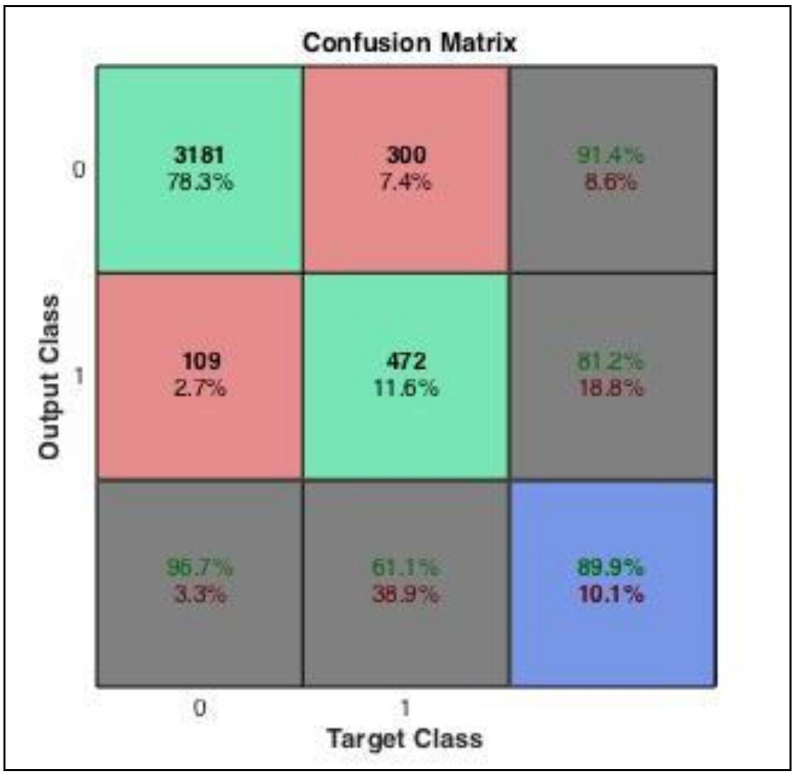


Figure C.11. ANN Classification Results of the 'Religion' Category Testing Database.

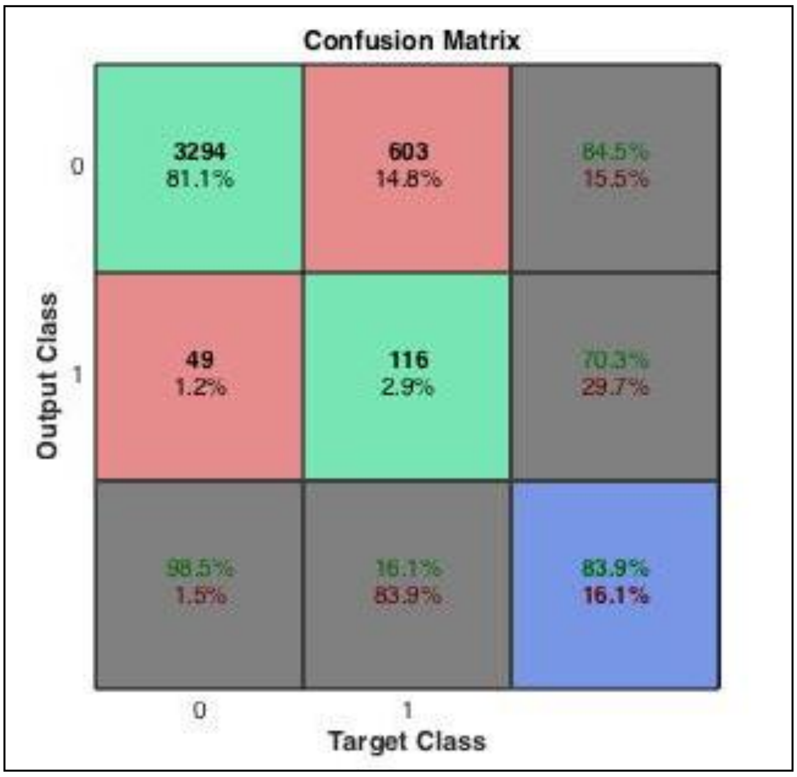


Figure C.12. ANN Classification Results of the 'Sports' Category Testing Database.

Appendix D: SVM Classification Results

D.1 Training Database

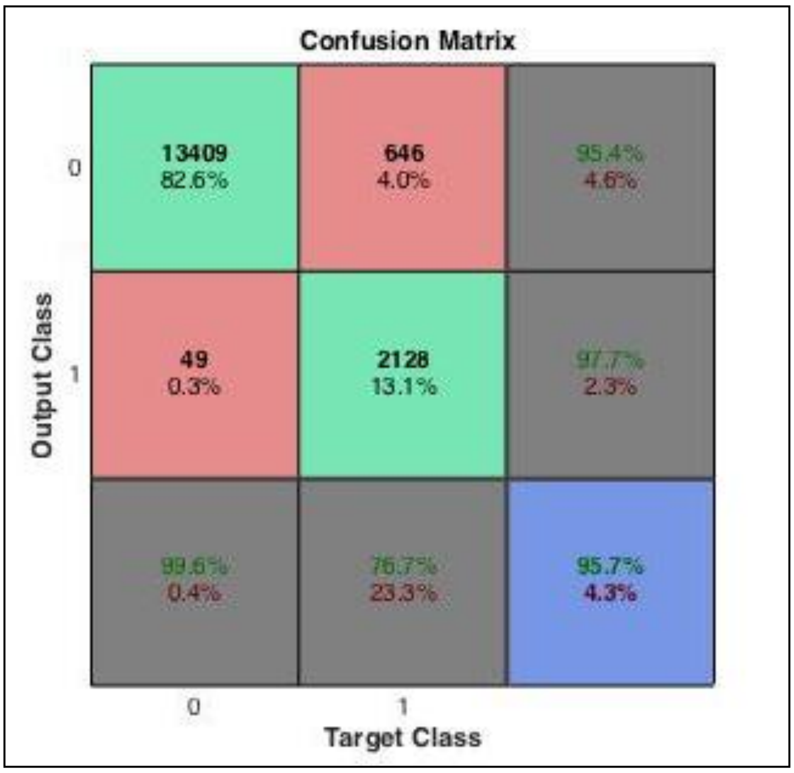


Figure D.1. SVM Classification Results of the 'Culture' Category Training Database.

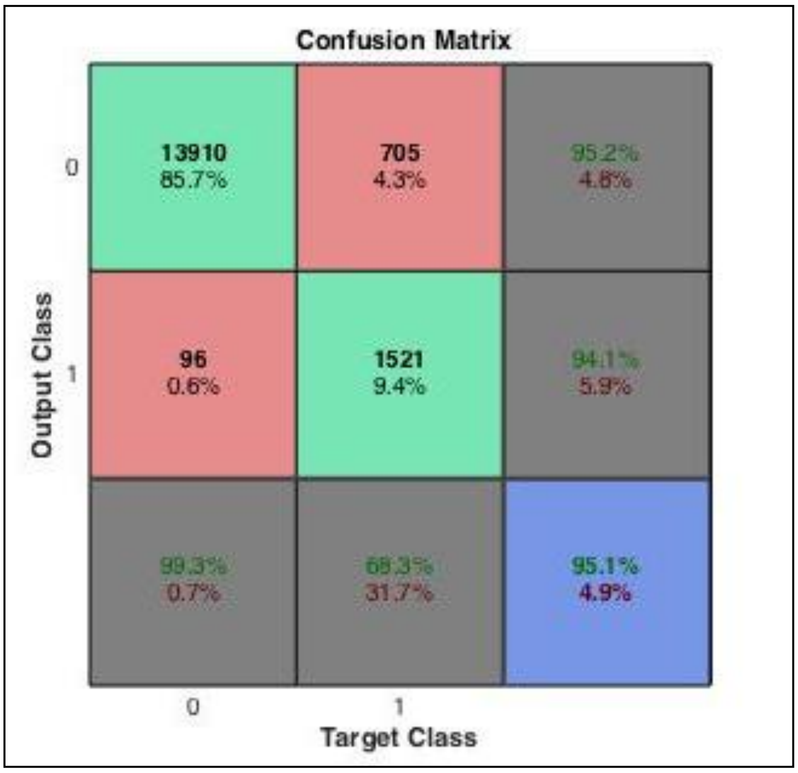


Figure D.2. SVM Classification Results of the 'Economy' Category Training Database.

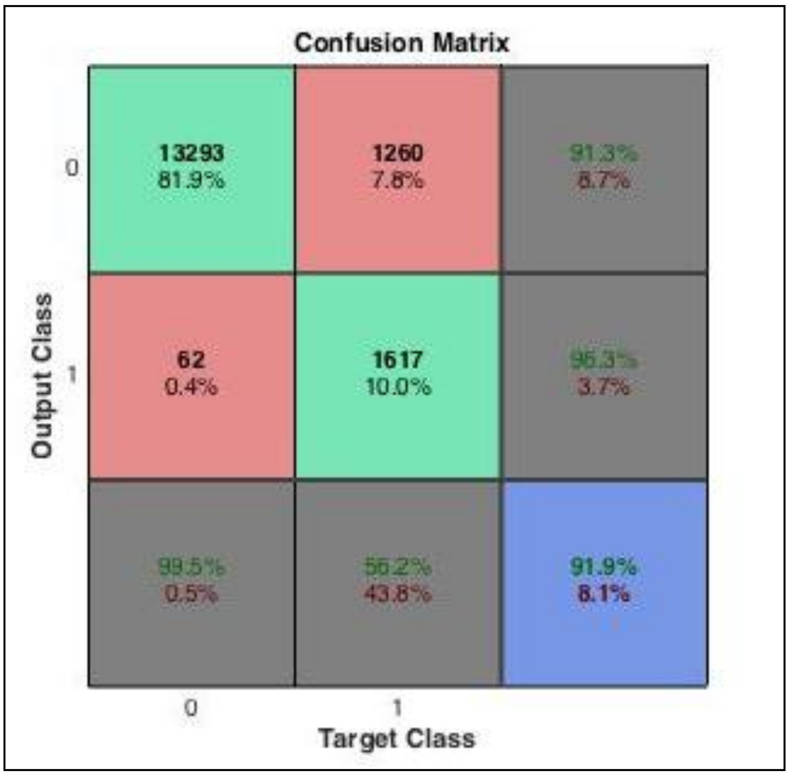


Figure D.3. SVM Classification Results of the 'International' Category Training Database.

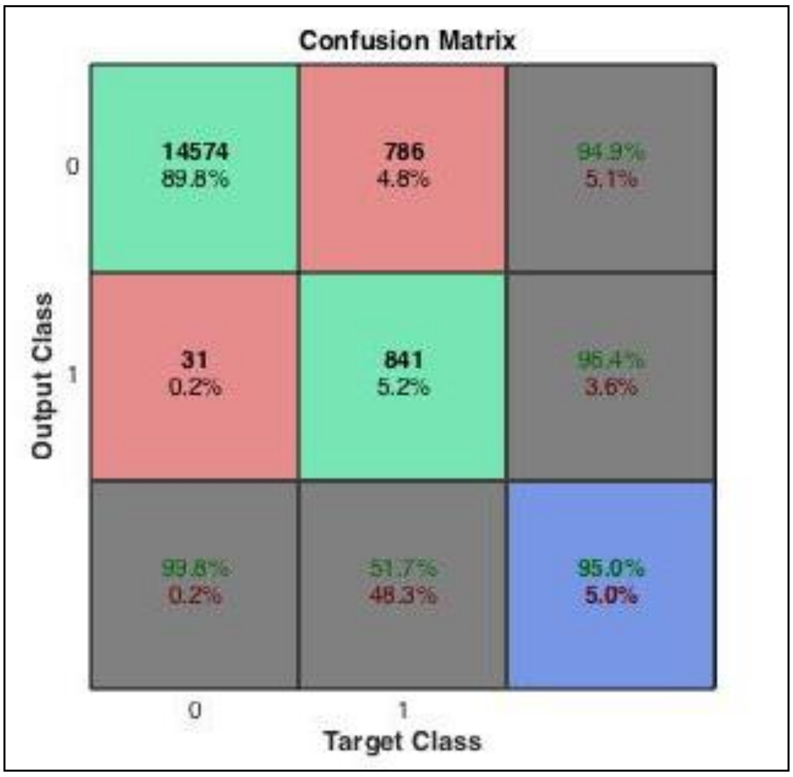


Figure D.4. SVM Classification Results of the 'Local' Category Training Database.

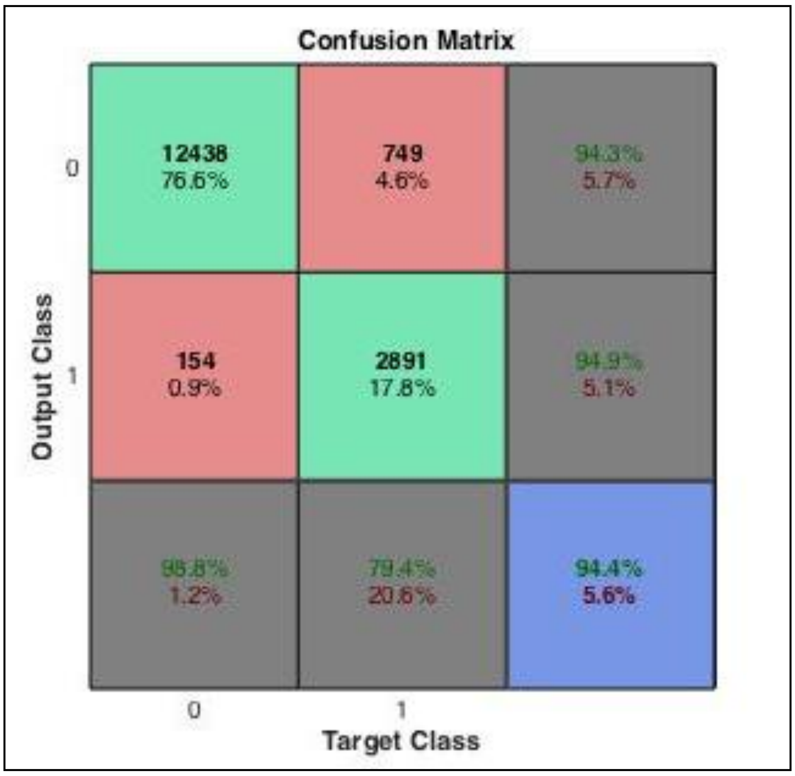


Figure D.5. SVM Classification Results of the 'Religion' Category Training Database.

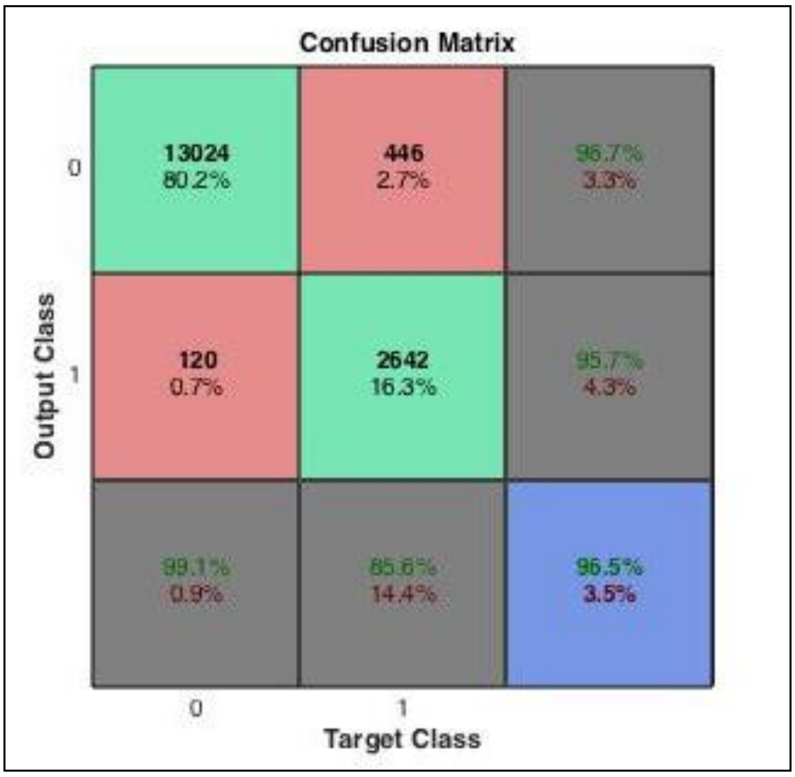


Figure D.6. SVM Classification Results of the 'Sports' Category Training Database.

D.2 Testing Database

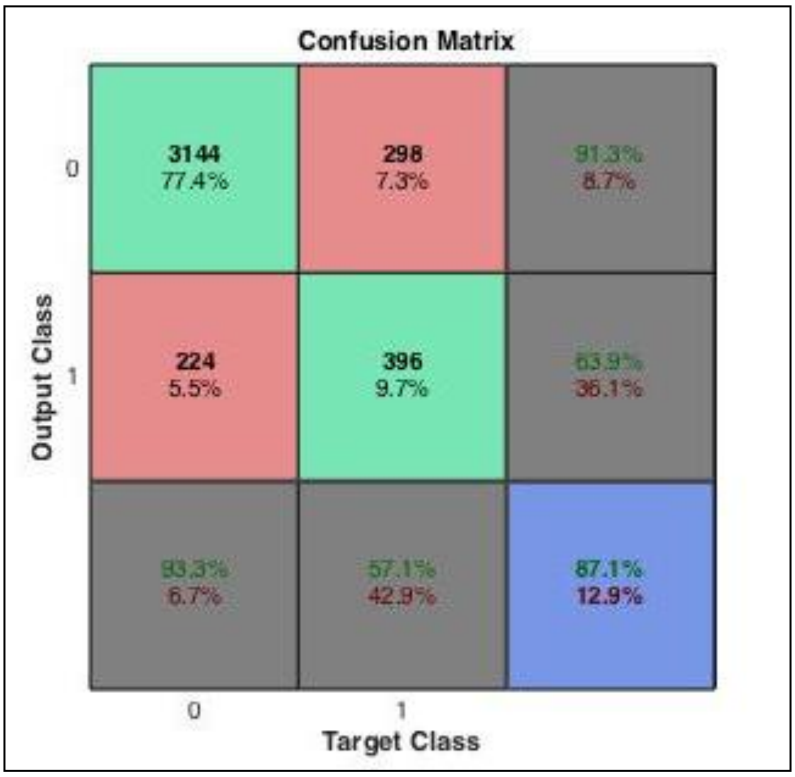


Figure D.7 SVM Classification Results of the 'Culture' Category Testing Database.

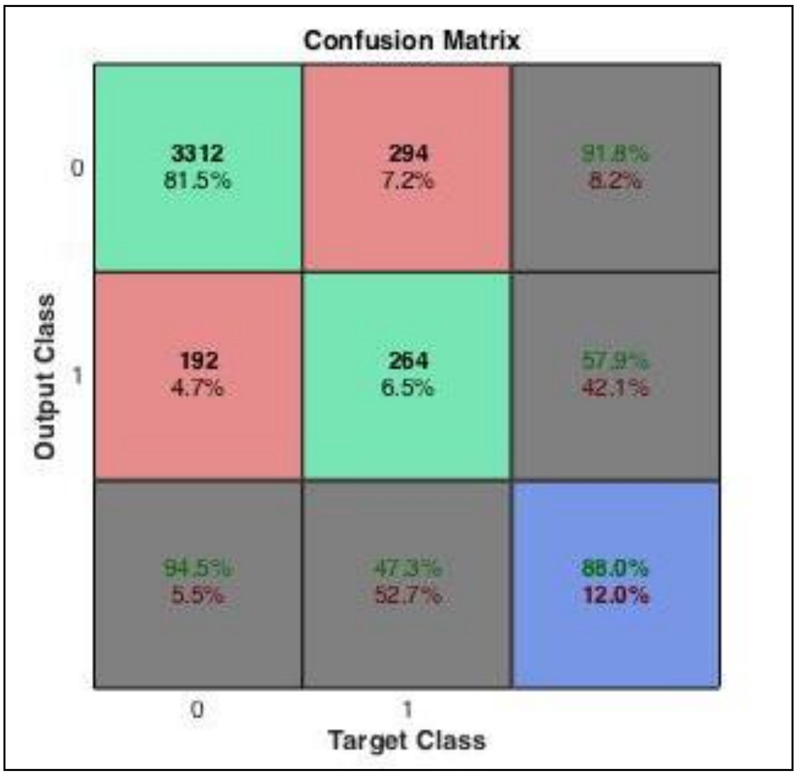


Figure D.8. SVM Classification Results of the 'Economy' Category Testing Database.

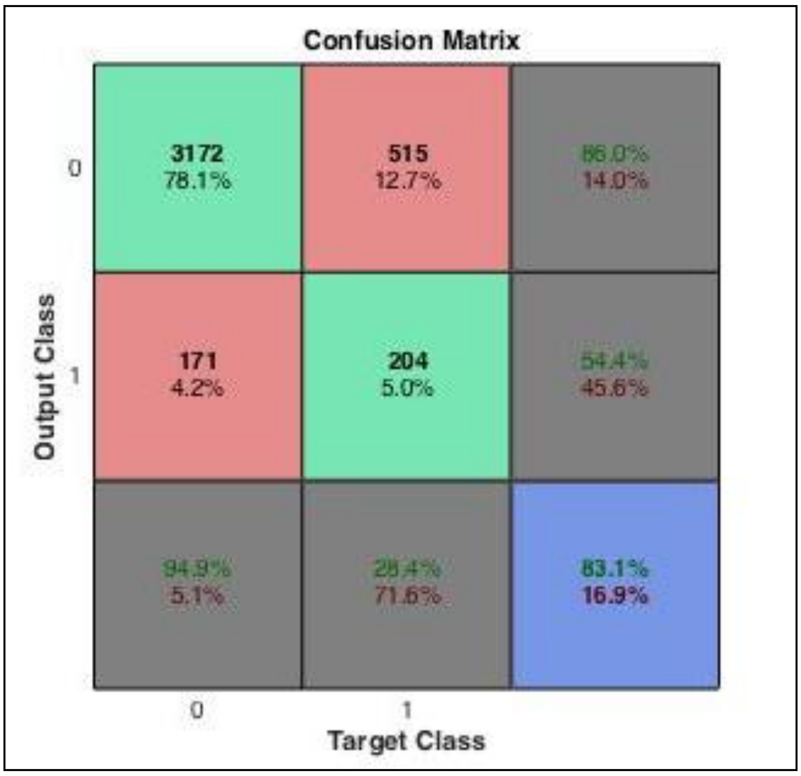


Figure D.9. SVM Classification Results of the 'International' Category Testing Database.

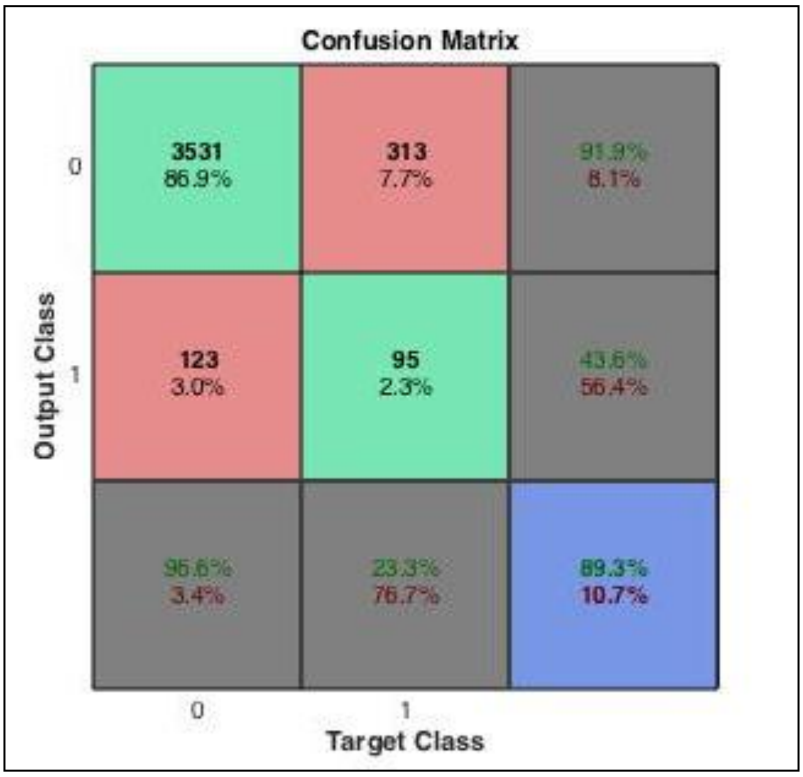


Figure D.10. SVM Classification Results of the 'Local' Category Testing Database.

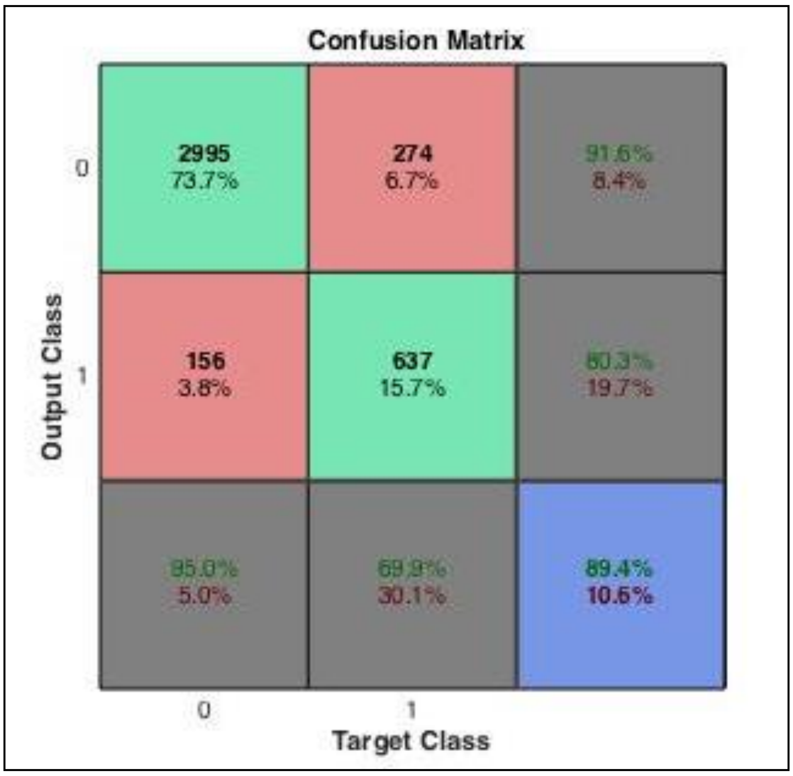


Figure D.11. SVM Classification Results of the 'Religion' Category Testing Database.

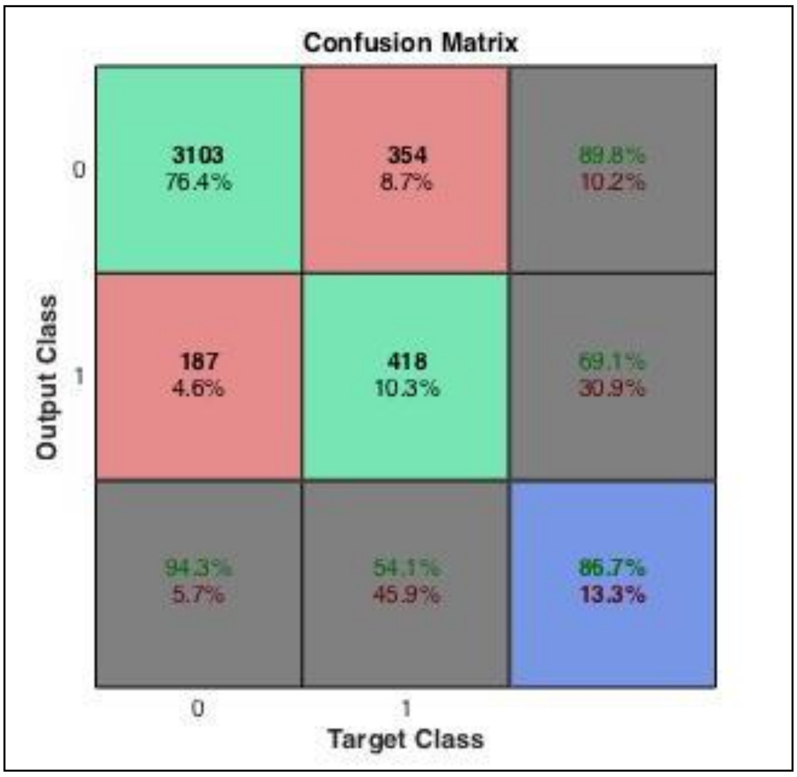


Figure D.12. SVM Classification Results of the 'Sports' Category Testing Database.