ARABIC LANGUAGE PROCESSING FOR TEXT CLASSIFICATION

Contributions to Arabic Root Extraction Techniques, Building An Arabic Corpus, and to Arabic Text Classification Techniques

by

May Yacoub Adib AL-NASHASHIBI

Thesis submitted for the degree of

Doctor of Philosophy in Computer Science

School of Computing, Informatics and Media

University of Bradford

2012

Abstract

Keywords: Arabic language, Root extraction techniques, Corpus development, Text labeling, Term weighting methods, Document representation, Single-label text classification.

The impact and dynamics of Internet-based resources for Arabic-speaking users is increasing in significance, depth and breadth at highest pace than ever, and thus requires updated mechanisms for computational processing of Arabic texts. Arabic is a complex language and as such requires in depth investigation for analysis and improvement of available automatic processing techniques such as root extraction methods or text classification techniques, and for developing text collections that are already labeled, whether with single or multiple labels.

This thesis proposes new ideas and methods to improve available automatic processing techniques for Arabic texts. Any automatic processing technique would require data in order to be used and critically reviewed and assessed, and here an attempt to develop a labeled Arabic corpus is also proposed. This thesis is composed of three parts: 1- Arabic corpus development, 2- proposing, improving and implementing root extraction techniques, and 3- proposing and investigating the effect of different pre-processing methods on single-labeled text classification methods for Arabic.

This thesis first develops an Arabic corpus that is prepared to be used here for testing root extraction methods as well as single-label text classification techniques. It also enhances a rule-based root extraction method by handling irregular cases (that appear in about 34% of texts). It proposes and implements two expanded algorithms as well as an adjustment for a weight-based method. It also includes the algorithm that handles irregular cases to all and compares the performances of these proposed methods with original ones. This thesis thus develops a root extraction system that handles foreign Arabized words by constructing a list of about 7,000 foreign words. The outcome of the technique with best accuracy results in extracting the correct stem and root for respective words in texts, which is an enhanced rule-

based method, is used in the third part of this thesis. This thesis finally proposes and implements a variant term frequency inverse document frequency weighting method, and investigates the effect of using different choices of features in document representation on single-label text classification performance (words, stems or roots as well as including to these choices their respective phrases). This thesis applies forty seven classifiers on all proposed representations and compares their performances. One challenge for researchers in Arabic text processing is that reported root extraction techniques in literature are either not accessible or require a long time to be reproduced while labeled benchmark Arabic text corpus is not fully available online. Also, by now few machine learning techniques were investigated on Arabic where usual preprocessing steps before classification were chosen. Such challenges are addressed in this thesis by developing a new labeled Arabic text corpus for extended applications of computational techniques.

Results of investigated issues here show that proposing and implementing an algorithm that handles irregular words in Arabic did improve the performance of all implemented root extraction techniques. The performance of the algorithm that handles such irregular cases is evaluated in terms of accuracy improvement and execution time. Its efficiency is investigated with different document lengths and empirically is found to be linear in time for document lengths less than about 8,000. The rule-based technique is improved the highest among implemented root extraction methods when including the irregular cases handling algorithm. This thesis validates that choosing roots or stems instead of words in documents representations indeed improves single-label classification performance significantly for most used classifiers. However, the effect of extending such representations with their respective phrases on single-label text classification performance shows that it has no significant improvement. Many classifiers were not yet tested for Arabic such as the ripple-down rule classifier. The outcome of comparing the classifiers' performances concludes that the Bayesian network classifier performance is significantly the best in terms of accuracy, training time, and root mean square error values for all proposed and implemented representations.

Dedicated to my Mother and the beloved memory of my Father

Mother, your dedication, unconditional love and the ethics that you patiently embedded in me while growing up have motivated me to arrive where I am.

Father, although you passed away, yet your free spirit, dignity, decency, integrity, and diligence to provide us with better opportunities than you had shall always inspire me to be my best.

Acknowledgements

I would like first to thank the School of Computing, Informatics and Media especially the Department of Computing at the University of Bradford for the high standard facilities provided for the students and the friendly atmosphere among the faculty and research students. I would like to express my deepest appreciation to my supervisor Prof. Daniel Neagu whose encouragement, support, advice, and fruitful discussions have guided me throughout my study. Prof. Neagu's support has made me not just enthusiastically working to develop a solid background on the subject, but also enjoying the process while doing so. I am grateful to Dr. Ali Yaghi for his support and guidance in the first phase of my research. His deep understanding of the Arabic language and experience has helped me a lot. I would like also to thank Petra University, Amman (Jordan) for partially financing my PhD study. My deep gratitude goes to the ex-vice president for academic affairs at Petra University Prof. Nizar El-Rayyes for his support throughout my study. Special thanks go to my colleagues, Basic Sciences Department at Petra University especially to the head of department Dr. F. Badawi and also Mr. I. Suweid for their encouragements. I would like to express my appreciation to my cousin Eng. Samia Al-Shahabi for her warm welcome and hospitality throughout the period of my study in UK. I would like also to thank all those who participated in the questionnaire I distributed for my research especially my uncle Mr. Mazin Al-Shahabi. My thanks go to Miss Rona Wilson for her cooperation when ever needed. I would like to thank all my friends especially Reema, Basima, Yasmina, Asma'a, and Ghadeer for their support and encouragement. Special thanks go to my friend Fatima for her help in printing the foreign Arabized words list. Finally my gratitude goes to my sister Rula and brothers Adib and Mohammad for their unlimited support, encouragement and love.

List of Peer Reviewed Publications and Contributions

1- M. Y. Al-Nashashibi, A. Yaghi, and D. Neagu, "An improved root extraction technique for Arabic words" in proceedings of *2nd International conference on Computer Technology and Development ICCTD 2010*, S. Mahmoud and Z. Lian (Eds.), pp. 264-269, 2-4 November, Cairo, Egypt, 2010. IEEE Explore.

2- M. Y. Al-Nashashibi, A. Yaghi, and D. Neagu, "Stemming Techniques for Arabic Words: A Comparative Study" in proceedings of *2nd International conference on Computer Technology and Development ICCTD 2010*, S. Mahmoud and Z. Lian (Eds.), pp. 270-276, 2-4 November, Cairo, Egypt, 2010. IEEE Explore.

3- M. Y. Al-Nashashibi, A. Yaghi, and D. Neagu, "An improved root extraction technique for Arabic words" *in Computational Linguistics in The Netherlands CLIN 2010, Poster session,* 4 February, Utrecht University, Utrecht, The Netherland, 2010.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
ARFF	Attribute Relation File Format
BN	Bayesian Network
DF	Document Frequency
DM	Data Mining
DR	Dimensionality Reduction
EM	Expectation Maximization
FSS	Feature Subset Selection
IB	Information Bottleneck
IE	Information Extraction
IG	Information Gain
IR	Information Retrieval
KDD	Knowledge Discovery from Databases
KDT	Knowledge Discovery from Text
K-NN	K Nearest Network
LC	Label Cardinality
LD	Label Density
LDC	Linguistic Data Consortium
LSI	Latent Semantic Indexing
MI	Mutual Information
ML	Machine Learning
MSA	Modern Standard Arabic
NB	Naïve Bayes
NLP	Natural Language Processing
OR	Odds Ratio
RBF	Radial Basis Function
SMO	Sequential Minimal Optimization
SOM	Self Organizing Map
SVM	Support Vector Machine
TC	Text Classification
TFIDF	Term Frequency Inverse Document Frequency
TM	Text Mining
VSM	Vector Space Model
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1: Introduction

1.1 Introduction

The early 90's represented a turning point for research in automatic Text Classification (TC) due to two factors [34]. The first was when the Internet became free to be accessed by everybody, anywhere and anytime. The second was the vast development of hardware capabilities as well as the increased number of required special purpose systems. Furthermore, the continuously increasing number of internet users whose mother-tongue is not necessarily English urged researchers to investigate new methods or improve existing ones in order to process and organize the immense volume of online data [147]. Usually such textual data were manually labeled to specific categories by human experts [20], which is an expensive and time consuming process.

Arabic language is among the top ten languages (7th place) used in the Web. Also, for Arabic users an internet penetration[1] was found to be 17.5% and a growth in internet usage was 2,297.7% between the years 2000 and 2009 (further statistics and details of number of web sites, internet users in the Arab world are shown in appendix I)[2,3]. Such statistics emphasize the importance of applying Text Mining (TM) approaches especially TC to Arabic. However, Arabic language is a complex language and as such requires in depth investigation in terms of applying or improving available automatic processing techniques such as Natural Language

[1] From: Internet World Stats, url: http://www.internetworldstats.com/stats7.htm [1/6/2010]
[2] From url: http://www.labnol.org/internet/blogging/he-total-number-of-websites-on-earth/2257/ [1/6/2010]
[3] From Royal Pingdom, url: http://royal.pingdom.com/2010/01/22/internet-2009-innumbers/ [1/6/2010]

Processing (NLP), and/or TC. Among NLP processes that require investigation for Arabic are morphological analysis and Machine Learning (ML) methods for TC.

Much research has been conducted for the development, improvement of Arabic light stemmers (i.e. outputs stems only) or morphological analyzers (i.e. outputs roots) according to the level of analysis required. Such applications were concentrated mostly for Information Retrieval (IR) whether by building stemmers that handle inflectional or derivational morphology. Examples of commercial morphological systems are Sakhr's[4], Xerox's[5], and MORPHO3's[6] morphological analyzers. Reported Morphological analysis [28] systems for Arabic can be categorized into either systems that were implemented by individuals so as to be used partially in their academic research, or systems that were implemented by commercial institutes or companies as part of the market's needs for Arabic applications such as search engines. Since work in this thesis is within the first category, emphasis here will be on displaying its respective techniques. These techniques can be further subcategorized into: 1- Rule-based techniques as in [1], [18]; 2- Lexical-based techniques as in [56], [88]; and 3- Others as in [30], [15], [36], [124] and [17]. However, such techniques are not freely accessible on line for other researchers to use and compare except for Khoja's [111] morphological analyzer[7] or Buckwalter's stemmer[8] [41].

Although [26] there are many available Arabic corpora, yet there is no online bench mark large Arabic text corpus that is freely accessible for researchers to use for testing root extraction methods as well as ML methods.

[4] Sakhr's morphological processor can be found at: http://www.sakhr.com
[5] Xerox's morphological processor can be found at: http://www.xrce.xerox.com/Research-Development/Historical-projects/Linguistic-Demos/Arabic-Morphological-Analysis-and-Generation
[6] Morpho3's morphological processor can be found at: http://www.rdi-eg.com/technologies/Morpho.aspx
[7] This analyzer was not accwssible to author at period of implementing root extraction techniques.
[8] This stemmer's performance was lower than other reported extraction techniques [159] so not tested here.

There are many [92] ML methods that are used for classification or regression. This thesis concentrates on classification methods only. Most of these methods are not implemented so far for Arabic TC. This thesis's aim is to investigate and compare the performance of many TC methods for Arabic and test the effect of using different choices of features in document representation on TC performance by first improving and comparing the results of two root extraction methods.

1.2 Thesis Contributions

This thesis focuses on exploring different preprocessing methods and investigating their effect on TC performance. More specifically, this thesis focuses on investigating and improving root extraction methods. Its first contribution is that it improves two existing root extraction techniques, namely a rule-based method that extracts triliteral and quadriliteral roots and a weight-based one that extracts only triliteral roots. The improvements are performed through: 1- proposing and implementing an algorithm that handles irregular cases, 2- collecting a list of foreign Arabized words (aforementioned rule-based and weight-based root extraction methods do not handle), 3- proposing and adding a simple method to handle quadriliteral roots for the weight-based technique, and 4- investigating changing the weight options for letters in the weight-based technique on its performance. The importance of handling irregular and foreign Arabized cases comes from the fact that irregular words consist of *weak*[9], *two-letter geminated, hamzated*, and *eliminated-long-vowel words* that are available in about 34% in texts[10], whereas foreign Arabized words are available in about 11% in texts[10]. This thesis compares between these techniques in terms of their accuracy and execution time by testing their performances using a single-labeled Arabic corpus that is developed here.

[9] We use here Haywood and Nahmad 1998 terminology for describing Arabic irregular forms.
[10] Percentage values presented here are gathered from 40 texts chosen arbitrarily in the collection as is described in Chapter 4.

The second contribution of this study is that it investigates the effect of using the outputs of the best reported accuracy root extraction technique (i.e. normalized words, stems, and roots) as well as including other feature choices (their respective phrases) in document representation on single-labeled TC performance using also the developed Arabic corpus.

This chapter presents an overview of the research problem as well as its rationale, and the organization of this study. Also, it presents a brief outline of this thesis.

1.3 Motivation and Objectives

With the vast and expanding use of the internet especially in the Arab world and the enormous number of websites/pages that are provided in Arabic, it has become a necessity to have online tools/search engines that automatically classify/retrieve/translate Arabic documents/web pages according to the needs of the respective users (whether individuals or companies such as online newspapers and journals). Generally, available search engines that provide such services for Arabic are limited and although the literature provides much research work that specifically concentrated on more effective morphological analysis methods and their effect on IR efficiency yet more research work is required to investigate and improve the accuracy of these methods and their effect on TC performance or effective translation.

Despite the recent research studies that investigated the effect of using roots/stems as alternatives of words in Vector Space Model (VSM) [155] for Arabic on TC performance such as in [148] or [154], yet there is no consensus regarding which of features root, stem or word should be used to provide best TC results. Also, the effect of using other feature choices on TC performance for Arabic was not investigated. Furthermore, there is neither available Arabic benchmark corpus nor an

online corpus that is freely accessible except the recently developed small Arabic contemporary corpus by University of Leeds (about 420 text documents). Thus, in literature, research works use different text collections (mostly collected on ad-hoc basis and usually small in size) for TC which makes it difficult to compare the results of such works to reach a conclusion.

Since this thesis's aim is to test the effect of using different choices of features in document representation on TC performance for Arabic, then a need for developing an Arabic corpus to be tested for both root extraction and TC methods, implementing and improving root extraction techniques performances in order to obtain roots and stems for respective words accurately. This is performed here by improving two root extraction methods and taking the outputs of best performing method to be used for feature choices. Furthermore, it proposes and implements a new variant of Term Frequency Inverse Document Frequency (TFIDF) weighting method and uses it to all investigated representations mentioned above in order to emphasize term presence among categories. It also, implements various TC methods using representations that use either words, stems, roots and/or their respective phrases and compares their performances. This work identifies the appropriate text collections, tools and procedures to satisfy the objectives mentioned above.

1.4 Research Questions

In order to satisfy the objectives stated in this thesis, the focus here is on the following four research questions:

- ***Research Question (1)***

What are the steps to develop an Arabic corpus from two different small collections to be manually classified as single-labeled corpus among eight classes?

- ***Research Question (2)***

What are the available root extraction methods to be implemented in this research? What are their disadvantages and how to improve and compare their performance in order to obtain the most correctly outputted stems and/or roots for respective words using the developed Arabic corpus?

- ***Research Question (3)***

How varying feature choices in Vector Space Model representation of corpus will affect the performance of various text classification methods as well as proposing and implementing a variant of TFIDF term weighting? If there is an improvement in text classification performance, would it be statistically significant?

- ***Research Question (4)***

Which classifiers applied to various representations of Arabic corpus have the best performance? Are the results obtained for such classifiers in agreement with previously reported studies?

1.5 Research Approach and Methodology

In order to achieve the objectives of this research, the current study and contributions are constructed in the following three phases:

1- To address the first research question, two labeled Arabic text collections are gathered and after investigating their characteristics, their labeling is unified in order to have a comparatively large corpus. This lead to the development of a single-labeled Arabic corpus: described in Chapter 3.

2- To address the second research question, two available root extraction methods are implemented and their advantages and disadvantages as well as performances are investigated. This lead to the improvement of these two root extraction techniques, proposing and implementing an adjustment method as well as two expanded methods for the second root extraction method, and the validation

of such improvements/investigations by implementing these techniques using the developed Arabic corpus and finally the proposal to develop a root extraction system that handles foreign Arabized words: described in Chapter 4.

3- To address the third and forth research questions, the author found that not many choices of features in document representation have been investigated. However, for some choices that are investigated, no report of significance testing is present to conclusively affirm which has better effect on TC performance. This is reached after studying what has been investigated in literature for Arabic TC in terms of feature choices, term weighting methods, and tested classifiers. Thus, the investigation of the effect of using various features in document representation when using a proposed variant TFIDF term weighing method on single-labeled TC performance is presnted in this thesis. Various classifiers of all types for Arabic are tested and the comparison and critical analysis of their performances for Arabic TC is presented: described in Chapters 5 and 6.

The first phase involves the following: **1-** acquiring two text collections (the first one is collected by the author from various online newspapers, magazines and personal websites where she aimed to cover different geographical regions along classes specified, and the second was downloaded from University of Leeds website [25]), **2-** investigating their characteristics and unifying their labeling under eight general domains.

The second phase consists in: **1-** the development and improvement of two available root extraction techniques by extending their algorithms to handle irregular words, **2-** proposing and implementing an adjustment method as well as two expanded methods of the second root extraction method that handles specific cases of quadriliteral roots. The validation of implemented algorithms was by using two

criteria: accuracy and execution time. After critically evaluating such algorithms a final root extraction system is proposed that would handle foreign Arabized words.

The third phase involves applying the proposed variant of TFIDF method as well as representing features of documents in the corpus by either normalized words, stems, roots, or a hybrid of words and word phrases, stems and stem phrases, roots and root phrases. Then, such representations' effect on single-labeled TC performance is investigated using the WEKA tool [181]. This is performed by: **1-** applying a feature subset selection technique for such representations in order to see if varying the number of best selected features would improve TC performance, **2-** performing significance testing to verify whether indeed improvement/degradation in TC performance among representations is evident, **3-** comparing between the performance of forty seven classifiers applied on above representations is presented whether between different classifiers of the same type or between classifiers among different types. The tested classifiers are categorized in WEKA tool among eight types: **a) Nine Rule** learners where six different classifiers are tested among this category, **b) Thirteen Tree** learners where eleven different classifiers are tested among this category, **c) Eight Bayes-based** learners where six different classifiers are tested among this category, **d) Seven Function** learners where five different classifiers are tested among this category, **e) Two Miscellaneous** learners where two classifiers are tested among this category, **f) Twenty eight Meta** learners where seventeen different classifiers are tested among this category, **h) Four Lazy** learners, and finally **i) Four Multi-Instance (MI)** learners. No classifiers are tested among those last two categories since either the classifiers had poor performance or are not applicable. Thus, only six types of classifiers are investigated here.

1.6 Thesis Organization

This thesis is composed of six chapters (excluding this one) as shown in Figure 1 where:

- ***Chapter 2:*** critically reviews literature from the two main disciplines of this research: morphological analysis methods and text classification methods for Arabic as well as a brief description of available Arabic corpora.
- ***Chapter 3:*** focuses on describing the two text collections, the texts' respective different classes, their characteristics, and the methodology of unifying their classes.
- ***Chapter 4:*** presents our contributions towards improving two different approaches for extracting roots for inputted words. It also presents the methods for evaluating these approaches' performance, and proposes a root extraction system that incorporates the best features among such methods and handles foreign Arabized words.
- ***Chapter 5:*** discusses the design of proposed text representations, proposes a variant TFIDF term weighing and implements these proposals for single-labeled text classification Also, forty seven classifiers performances are presented, compared, and tested for significance on implemented representations.
- ***Chapter 6:*** critically evaluates and compares the performance of used classifiers on implemented document representations.
- ***Chapter 7:*** presents the major conclusions of the research as well as any research contributions. Then, the limitations of this research are discussed along with recommended future work.

1.7 Summary

This chapter presents the overall scope of this research by providing background information, introducing it and research aims. It then introduces the research approach and methodology and finally it outlines the thesis structure. In the next chapter a detailed literature review is presented.

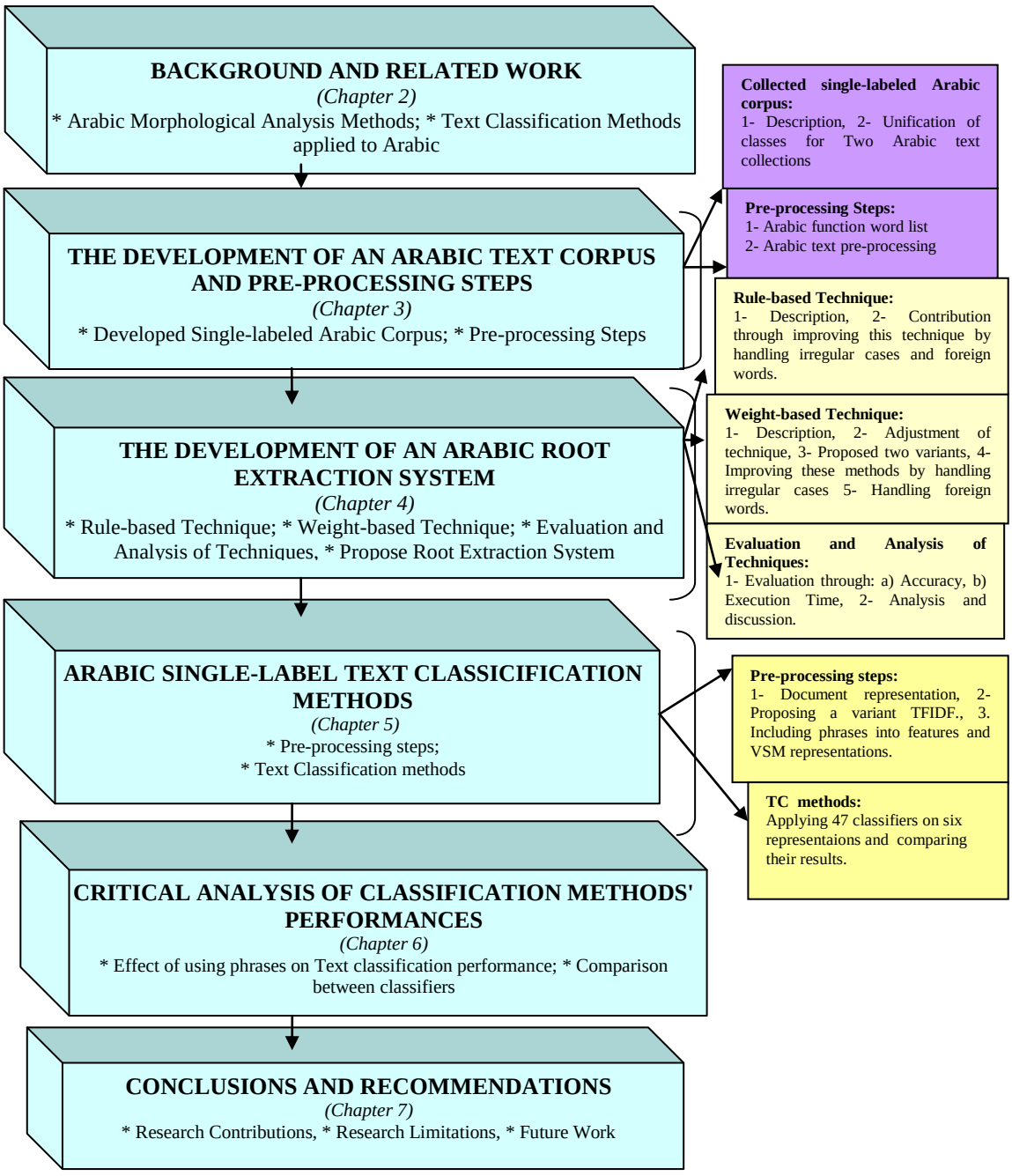


Figure 1: The structure of the thesis and steps followed in the research method

Chapter2: Background and Related Work

2.1 Introduction

This chapter presents a brief review of the literature on the application of two major research areas for Arabic. The two areas are morphological analysis and TC as illustrated in Figure 2. Since the aim of this thesis is twofold, then this chapter first briefly presents different techniques for stemming Arabic words, and decides which of these techniques will be used and improved in order to provide more accurate results. This chapter then discusses briefly applied TC methods for Arabic and compares their respective results with those for English. However, since a collection of Arabic texts, preferably large in size, is required to evaluate the performance of both root extraction and TC methods, this chapter first briefly presents available Arabic text corpora and which to use (if applicable) or develop in this thesis.

The remainder of this chapter is organized as follows: next a brief review of available Arabic text corpora. Section 2.3 discusses and classifies briefly available Arabic root extraction techniques in the literature. Section 2.4 describes briefly applied single-label TC techniques on Arabic. Finally, summary is presented in Section 2.5.

2.2 Arabic Text Corpora

Newspaper articles available online are the common and frequent source for obtaining Arabic texts. Available Arabic corpora can be found for example at Linguistic Data Consortium (LDC)[11] or European Languages Resources Association (ELRA)[12] websites. Further information on such corpora can be found in [26].

[11] LDC, University of Pennsylvania, USA, LDC website: http://ldc.upenn.edu/ [last accessed 1/5/2011]
[12] ELRA website: http://www.elra.info/ [last accessed 1/5/2011].

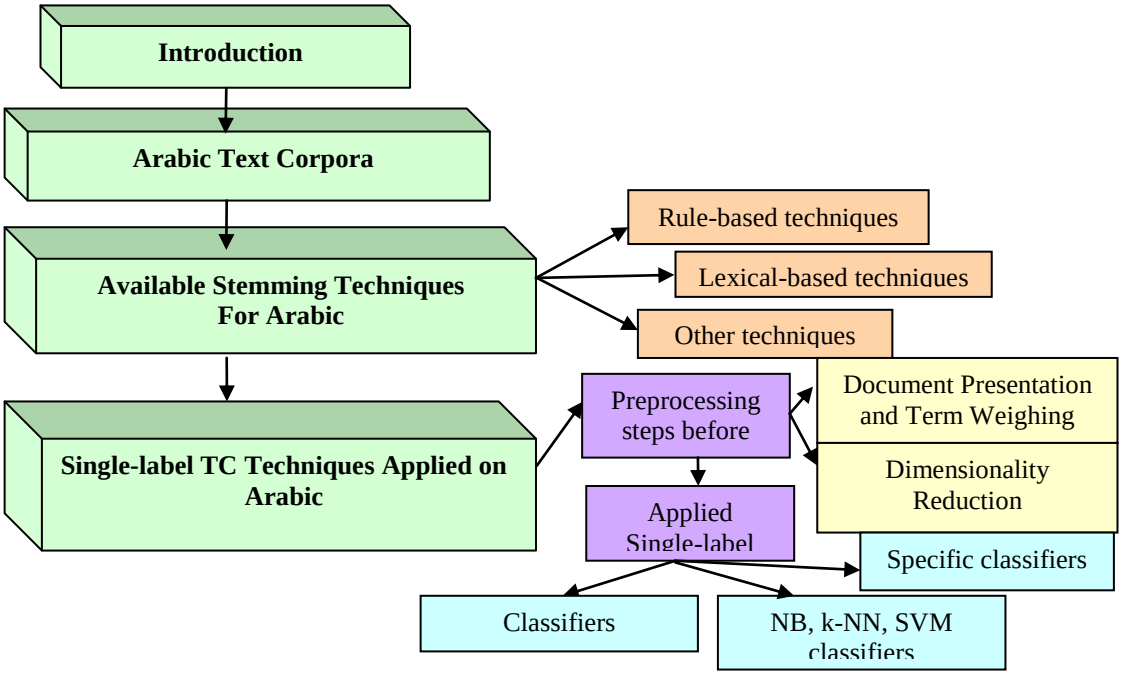


Figure 2: The Structure of Chapter 2

For example ELRA has two Arabic corpora in Modern Standard Arabic (MSA) form. The first is An-Nahar newspaper text single-label corpus that contains about 270,000 texts and about 144 million words under 5 classes. The second is Al-Hayat Arabic single-label corpus under 7 classes. On the other hand, LDC has larger scale Arabic text corpora such as Arabic Newswire Part 1 corpus. This corpus, although without any labels, contains 383,872 texts, about 76 million words and about 666,094 unique words. Another example in LDC is the Arabic Giga word corpus with its 1st, 2nd, 3rd and 4th editions. Its 4th edition has 2,716,995 texts under one of three labels ('story', 'multi' (contains a series of unrelated blur[13]), and 'other' labels). All previous Arabic corpora are not freely accessible to researchers as the English RCV1-v2 [152], [127] or the Reuters-21578 [126] corpora. Thus, there is no available online benchmark freely accessible Arabic corpus, whether single or multi labeled. As such, two small text collections that are used in this thesis are described and arranged in one corpus as is thoroughly explained in Chapter 3. The first text

[13] as stated in: http://catalog.ldc.upenn.edu/LDC2003T12

collection was gathered from different websites and categorized into eight general domains. The second text collection was downloaded from University of Leeds website on January 2010 [25], originally categorized into eight other general domains.

Thus, next a brief description of methods in literature used for choosing general domains for manual classification of texts.

In 1996, Sinclair [165] proposed 35 domains to present texts. Later on Sharoff [163] categorized texts into eight general domains since Sinclair's categorization list, in his opinion, provides a too fine-grained list. The eight general domains are: 1- *natural sciences* (mathematics, biology, physics, chemistry ... etc), 2- *applied sciences* (agriculture, medicine, ecology, engineering, computing, transport .. etc), 3- *social sciences* (law, history, philosophy, psychology, language, education .. etc), 4- *politics* (inner, world), 5- *commerce* (finance, industry), 6- *life* (general domain e.g. fiction, conversation .. etc), 7- *arts* (visual literature, architecture, performing), 8- *leisure* (sports, travel, entertainment, fashion .. etc). However, Eibeed [64] suggested that Arabic articles be classified according to one of 10 domains that are: 1- general, 2- philosophy and psychology, 3- religion, 4- social sciences, 5- languages, 6- natural sciences and mathematics, 7- applied sciences, 8- arts, 9- literature, and 10- geography and history. In [64], O. Dawood suggested to have 15 domains that are as follows: 1- religion, 2- sports, 3- educational sciences and scientific research, 4- medicine and health, 5- encyclopaedias, 6- philosophy and psychology, 7- languages, 8- Arabic articles translated to other languages, arts, 9- Foreign articles translated to Arabic, 10- social sciences, 11- applied and natural sciences, 12- history and its sciences, 13- geography and geology, 14- mathematics, and 15- others. It is clear that

although the topics are more or less the same, the categorization process and numbers of general domains are different.

The enormous number of texts that is available online and is increasingly growing explosively on the web requires for such texts to be classified into domains or topics so that these texts can be for example retrieved easily. The determination of such domains is crucial for the accuracy of the classification process whether there is hierarchy in domains/genres or not. This is so since [165] if a text is to be classified among predefined classes that are Physics, Biology among others where this text's class is actually Bio-Physics (which is not among the predefined classes). Then, it is expected that results of classifying this text will be lower since the boundaries among such classes are neither fixed nor clear. The reason for this lower performance is partially due to the fact that the classifier is designed to only choose one class among those predefined, and then it would provide the wrong classification. However, for the same classes, if the classifier chooses more than one class, then it can provide a more representative answer to the classification required. One of the criteria [161] that a corpus is characterized by is the generality of its classes. A *generality* of a class c_i is defined to be the percentage of documents d_j in this class to the total number of documents in the corpus Ω as shown in eq. 1.

$$g_{\Omega}(c_i) = \frac{\left|\{d_j \in \Omega | \Phi(d_j, c_i) = T\}\right|}{|\Omega|} \qquad (1)$$

Among the issues [178] related to multi-labeled data sets is how to identify the amount of data that is multi-labeled in such sets. A method for doing so is by using the concepts of Label Cardinality (LC) and Label Density (LD). For these concepts eq. (2) are provided below, where *D*: is data set with size |*D*|, |*L*| is number of labels; $|Y_i|$ is number of assigned labels for document d_i. Multi-labeled data sets vary in terms of their LC and LD values. An example [178] showing two data sets is for the

genbase and *yeast* sets with LC = 1.35, LD = 0.05 and LC = 4.25, LD = 0.3 respectively. From the equation for LC it is noticed that LC value does not depend on the number of labels in corpus but rather is the average number of labels per text in it. However, LD value depends on the number of labels assigned.

$$LC(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}|Y_i|, \qquad LD(D) = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i|}{|L|} \tag{2}$$

In Chapter 3, the development of the Arabic text corpus is presented in detail where the two collections used to develop this corpus are described. Next is a description of stemming techniques for Arabic.

2.3 Stemming Techniques for Arabic

Large-scale morphological analyzers [49] usually outputs to the user besides the root for the inputted word further information. Such information may be the meaning of prefixes, suffixes, and/or root disambiguation. Reported Morphological analysis [28] systems for Arabic can be categorized into either systems that were implemented by individuals so as to be used partially in their academic research, or systems that were implemented by commercial institutes or companies as part of the market's needs for Arabic applications such as search engines.

Much research has been conducted for the development, improvement of Arabic light stemmers (i.e. outputs stems only) or morphological analyzers (i.e. outputs roots) according to the level of analysis required. Since the stemming techniques implemented in this work are within the first category, emphasis will be on displaying its respective techniques. These techniques can be further subcategorized into: 1- Rule-based techniques; 2- Lexical-based techniques; and 3- Others as will be described briefly in the following three subsections.

2.3.1 Rule-Based Techniques

Many Arabic morphological analysis approaches use rules, prefix and suffix lists to identify the possible roots for any Arabic word in MSA. Such approaches are thus called rule-based. However, building and implementing the rules for this analyzer is time consuming. These methods, based on the required level of analysis, can be [29] subcategorized into: a) one-level rules that analyze words at the stem level using regular concatenation, b) two-level rules that analyze words as roots + patterns + concatenation, and c) three-level rules that analyze words as roots + templates + vocalization + concatenation for obtaining their roots. Examples of stemming techniques that use one-level rules to produce stems are as in [120], [121], [9], [44], [29] and [50]; whereas examples of stemming techniques that provide roots (whether two-level or three-level rules) are as in [1], [18], [22], [3], [11], [48], [16] and [72].

Although approaches that obtain roots/stems use rules, these approaches differ in: a) the number of patterns, prefixes, suffixes used, rules' order in the approach, and the amount of included function words to be removed, b) applying a normalization step for some letters in words or not and if so to what letters, and c) applying/removing diacritics for words or not.

2.3.2 Lexical-Based Techniques

This method uses lexical databases, dictionaries, and/or thesaurus to establish, among other things, if the possible combinations of prefix-(root + pattern)-suffix is correct for the processed word. Some of the research works that use lexical databases, dictionaries, and/or thesaurus, use it along with methods as finite state transducers [35], while others use it with rules [41], [56], [87], [111] and [182]. However, such techniques are usually collected based on a corpus and so are limited. This limitation affects the performance of such analyzers negatively.

2.3.3 Other Techniques

The third method of stemming approaches, which is proposed for Arabic, cannot be in our opinion categorized into any of the previous two categories. Examples of such approaches are as in [54], [131], [151], [160], [49], [52], [28], [30], [15], [37], [124] and [17]. All the above research works except in [17] used either statistical or ML methods for morphological analysis.

Al-Shalabi, et al work [17] provided a rank, order and weight for each letter in the word according to its position and calculated the product of rank and weight. This method then only extracted a triliteral root for that word by choosing the three least product values for letters of word without any change in their positions. Implementing this method on a small set of Arabic abstracts reported an accuracy of about 90%. Most of the aforementioned ML methods are further explained and compared in appendix I.

ML methods, especially classification methods, usually require large corpora (as the Reuters-21578 [126] corpus) for training in order to establish good results. Next is a comparison between the types of stemming techniques presented above.

2.3.4 Comparison between Stemming Techniques

Although much work [24] have been performed on Arabic morphological analysis and stemming especially for IR applications, yet few of such works handled specific cases of irregular words (i.e. *weak, eliminated-long-vowel, two-letter geminated,* and *hamzated* words) but not all of these cases except the works in [72] and [35]. It is noteworthy that in [72] no results were provided of the system implemented. Also, in [35] the Xerox demo[14] is available, and although efficient it

[14] Xerox demo can be found at http://www.xrce.xerox.com/Research-Development/Historical-projects/Linguistic-Demos/Arabic-Morphological-Analysis-and-Generation

requires usually a relatively long time to provide the required roots (from about 1 hour to 7 day according to number and type of words provided).

The less-studied cases of irregular words are *weak* words [5], [42]. The lack of study of such irregular cases is partially expected since research works concentrated on building various light stemmers for IR especially in more recent times. Such concentration was due to the extensive research on whether to use stems or roots for improving IR performance.

When developing for Arabic a stemmer or morphological analyzer, important issues present themselves [24] such as under-stemming and over-stemming. Other issues that require handling for stemmers are compound words, proper nouns, foreign Arabized words, diacritization, word disambiguation and irregular forms of words. Also, available stemmers [22] automatically stem blindly words whether proper nouns, foreign Arabized words or others and thus perform poorly. Thus, there is a need to build an algorithm that handles some/all of these cases.

There is no comparative study available in the literature that compares between available Arabic stemmers and evaluates their performances except for the works in [160], [122] and [23]. The work in [160] compares between the performances of three different stemmers in terms of accuracy. Larkey et al work [122] compares between the performance of their light stemmer with other stemmers for IR as the well-known Khoja stemmer[15], Buckwalter stemmer, and Diab's Lemmatizer [55]. The work in [23] compares the performance of six existing algorithms for root extraction, four of them are rule-based [16], [84], [172], and [169], one lexical-based [111] and the last is a weight-based one [17]. This work implemented such algorithms and compared their performance using a corpus that was built from 3,823 triliteral roots and applying 73 patterns with 18 suffixes and producing 27.6 million

[15] The words stemmer and root extractor is used here interchangeably.

words. The highest obtained accuracy among these six algorithms was the one of Ghwanmah, et al [84] work of 39% only. The results obtained in this work are rather interesting considering that the reported accuracy in original works were above 90%. However, the corpus used in this work is much larger than the ones used in the original papers stated above.

There was a discrepancy among the published studies [24] regarding which are better for IR, using words, stems or roots as in [3], [11], [97], [9], and [48]. Larkey, et al [121] verified that using either stems or roots improved significantly IR performance (much better than applying stemming on English for IR). Later on, Larkey, et al [122] concluded that the effect of using light stemming is the highest on IR for Arabic after comparing the effect of their light stemmer with other stemmers or root extractors. This agrees [51] with what is known that IR is more tolerant to over-conflation (i.e. removing letters at beginning or end that are not extra letters) than under-conflation. Also, although [158] word sense disambiguation has been reported to decrease retrieval effectiveness, yet by improving the correctness [51] of morphological analysis (here context sensitive which is an akin to sense disambiguation) retrieval results improved slightly. A drawback of context-sensitive morphological analysis is that it requires considerably more computing time than light stemming.

It is worth noting that names of places, countries, cities, months and foreign Arabized words compose about 11% of texts[22]. A more comprehensive percentages of words is described in [2] where the percentage of occurrence for proper nouns, 'verbs, nouns and adjectives', broken plurals, function words, and deverbals (i.e. infinitive forms, active and passive participles, analogous adjectives and nouns of place and time) are 1.14%, 16.01%, 24.3%, 7.87%, 0.37%, and 58.18% respectively.

Such percentages are based on the analysis of two million words using AraMorph[16] and DIINAR.1[17] [57]. Thus, Arabic words are highly derivational. More specifically, *weak*[18], *two-letter geminated*, *hamzated*, and *eliminated-long-vowel words* are available in about 13%, 7%, 11% and 2% (12% of weak words) in texts respectively[19]. Arab linguists and consequently early research works on morphological analysis, lexicons, and machine translation base the analysis of words on their root + pattern structure. However, Dichy and Farghaly [57] argued that this is not sufficient since root + pattern representation does not handle Ancient and medieval words as *IsmAEyl* as well as the complex grammar-lexis relations in Arabic words. This is so since such representation handles only verbs and deverbals. This was also emphasized by Abbès, et al [2] which showed that using only prefix-suffix combinations (without proclitics (i.e. letters at beginning of words as *l*, *w*, *b*) or enclitics (i.e. complement pronouns)) are more ambiguous than when such clitics are taken into consideration. On another point, Darwish, et al [51] investigated the effect of context sensitive morphological analyzers on IR. Results of this work show that better coverage and improved correctness have a dramatic effect on IR effectiveness. Next is a brief description of reported applied TC methods for Arabic and the effect of some preprocessing steps on their performances.

2.4 Text Classification Approaches Applied to Arabic Sources

TM is an interdisciplinary area [87] that involves fields as ML and Data Mining (DM), Statistics and statistical methods, IR, and Natural Language Processing (NLP). Researchers have explored and developed many TM and NLP techniques and

[16] AraMorph can be found at: hhtp://sourceforge.net/projects/aramorph, [last accessed 1/11/2011]
[17] DIINAR.1 can be found at: http://catalog.elra.info/product_info.php?products_id=902, [last accessed 1/11/2011]
[18] We use here Haywood and Nahmad 1998 [96] terminology for describing Arabic irregular forms.
[19] Percentage values presented here are gathered from 40 texts chosen arbitrarily in the collection as will be described in Chapter 4.

algorithms especially to English language but few have been proposed for Arabic text automatic interpretation. This is partially due to the rich morphology [96] of Arabic language. TM methods, which are also known as DM methods for texts, are applied [92] for basically: 1- TC (supervised learning), 2- clustering (unsupervised learning), 3- IE, 4- association analysis, and 5- trend analysis. TC is the process of assigning a text document to one or more predefined classes based on its content [129].

Text processing techniques, such as TM methods, according to application, are applied at different levels [87] at: 1- word level, 2- sentence level, 3- document level or 4- document-collection level. If applied at word level, different processes might be applied by: a- taking word properties (such as homonymy, polysemy, synonymy, and hyponymy or word frequencies) into consideration, b- removal of function (stop) words, c- using stemming or lemmatization, d- using frequent n-grams, and finally e- using lexical relations databases such as Word Net (WN) for English or recently for Arabic (AWN). However, if text processing is applied at document level, it would be used, among others, for text summarization as in [138] when applied on English; and [68], [69] when applied on Arabic. Finally, when it is applied at document-collection level, several issues require handling such as choosing a representation of document, deciding a similarity function to compare documents, reducing the high dimensionality of documents by choosing an effective method to do so for categorization or clustering. One of possible applications of using ML techniques for Arabic is for language identification as in [162].

There are few available commercial tools or software that applies TM techniques on Arabic such as Sakhr™ automatic commercial categorizer, keyword extractor, and summarizer [153] and [66] KP-Miner system[20].

In the following subsection preprocessing steps for TC will be illustrated. In subsection 2.4.2 single-labeled TC methods applied to Arabic will be discussed.

2.4.1 Pre-Processing Steps for Text Classification

Applying TC techniques requires usually a preprocessing stage that would remove punctuation marks, function words and might return the remaining words to their stems (for Arabic words to their stems or roots). For English language, researchers perform the stemming step in order to reduce the high dimensionality of documents [161]. Since the research work on Arabic TC is rather new, many issues need to be investigated in order to establish its effect on TC. Among such issues are: document representation, types of features used, methods of weighting, and feature selection or extraction. Research work investigating such issues is presented next.

2.4.1.1 Document Representation

A text document d_j is usually represented [161] as a vector of term weights $(w_{kj}), \vec{d}_j = \langle w_{1j}, \ldots\ldots, w_{|T|j} \rangle$, where T is the set of different terms (also called features) that occur at least once in at least one document in the training set from the collection. Different approaches are used for document representation, where two differences occur for text representation. The first is related to how a term is considered (for example a word or a phrase). The second is related to which method the weights are calculated by. For Arabic the only representation of features in texts investigated in literature was using words, stems, or roots separately but as far as is known, the choice of features using phrases or combining them with other forms of

[20] Found at: http://www.claes.sci.eg/coe_wm/kpminer/ [last accessed 1/11/2011]

features was not reported for Arabic TC. As for term weighting, the *TFIDF* function, besides the BOOLEAN representation of terms, is used mostly for weighting [156]. *TFIDF* is defined in eq. 1 as:

$$tfidf(t_k, d_j) = \#(t_k, d_j).\log\left(\frac{|Tr|}{\#_{Tr}(t_k)}\right) = tf(t_k, d_j).idf(t_k) \quad (1)$$

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sum_{k=1}^{|T|}(tfidf(t_k, d_j))^2} \quad (2)$$

Where $tf(t_k,d_j)$ (called Term Frequency) = $\#(t_k,d_j)$: number of times term t_k occurs in d_j,
$\#_{Tr}(t_k)$ (called document frequency ($df(t_k)$): number of documents in Tr that t_k occurs in,
$|Tr|$: number of documents in training set,
Inverse document frequency $idf(t_k)$ is given by $idf(t_k) = \log\frac{|Tr|}{df(t_k)}$,

However, other term weighting methods are used such as the weighted inverse document frequency $Widf(t_k,d_j)$ or the recently proposed term weighting Modified inverse document frequency $Midf(i,j)$ [53] or the pivoted document normalization [166], [167] weighting equation. These methods' equations are provided in appendix I. The *idf* part of *TFIDF* function defined in eq. 1 above handles the effect of the presence of a term in documents compared to the total number of documents in the corpus (i.e. global weighing). So if such a term appears in different categories in different percentages and/or the generality of such categories is not balanced (i.e. not the same or near one another) then this part of *TFIDF* will not include such an effect into weighing the term. Thus, weighing such terms by considering their presence among categories is neither proposed nor implemented (which is called here local weighing). Although new variants of *TFIDF* or other functions were proposed and implemented on texts for English as in [45], [46], [58], [116], [128] and [176], nevertheless such methods were not investigated for Arabic. Also, as far as we know, the effect of local weighing was not investigated for Arabic TC.

The method described above for text representation is called Vector Space Model (VSM) [155]. The above weighting measures have been thoroughly investigated and compared for TC on English [161]. Many of the TC methods that were implemented on Arabic, used *tfidf* for weighting but only three works [108], [171] and [173] studied the effect of most of weighting methods for Arabic on TC performance. Thabtah, et al [173] studied the effect of using $tf(t_k,d_j)$, $idf(t_k)$, *tfidf*, $Widf(t_k)$, inverse $tf(t_k,d_j)$ $(itf(t_k,d_j))$, and $\log(tf(t_k,d_j))$ on classification performance and concluded that using *tfidf* provided best results. Also, Syiam, et al [171] studied the effect of using $tf(t_k,d_j)$, Boolean, *tfidf*, and normalized *tfidf* for weighting on classification performance and concluded that using normalized *tfidf* provided best results. However, Kanaan, et al [108] used $tf(t_k,d_j)$, *tfidf*, $Widf(t_k)$ for both k-NN and Rocchio classifiers. This work showed that best results were provided: a) when using *tfidf* for Rocchio classifier, b) when using $Widf(t_k)$ for k-NN classifier. It is noteworthy that no significance testing in works mentioned above was reported. In general, such results for TC on Arabic are in agreement with those concluded for English.

2.4.1.2 Dimensionality Reduction

In TC, the large number of terms [161] could be problematic, since such TC methods cannot scale for large number of terms (i.e. high dimensionality). That is why before implementing any TC algorithm, a technique to reduce the dimensionality of the vector space from $|T|$ to $|T'|$ such that $|T'| << |T|$ is often applied.

There are various Dimensionality Reduction (DR) techniques that are used in the literature whether coming from information theory or from linear algebra. DR methods are viewed through two different ways: 1- by performing it either locally (per category) or globally, or 2- by performing it in terms of the nature of the resulting terms (i.e. term selection versus term extraction). Local DR uses different

sets of document vectors according (e.g. the work of Apté, et al [32]) to their respective categories. Global DR uses the same set $|T'|$ for all categories (e.g. the work of Yang and Pedersen [183]) such that $|T'| << |T|$.

Another method that is used by some researchers for DR is stemming. For English language, [161] researchers differ on the effectiveness of using stems in the representation of documents in the preprocessing stage to improve TC results but agree that the stemming step is done in order to reduce the high dimensionality of documents. Next, DR by applying stemming on Arabic is discussed.

2.4.1.2.1 Dimensionality Reduction by Stemming

Applying different stemming techniques on Arabic texts and investigating the effect of such techniques for DR on TC performance have been undertaken by only few research papers [63], [108], [135], [146], [147], [148], [154] and [176]. The research works that investigated the effect of stemming on classification performance for Arabic are presented in appendix I. In these papers, the classifier(s) used stems, roots or words for features and their performances were compared in order to establish if stemming improved TC.

The works that compared the effect of using words, roots or stems for features, although used different stemmers, classifiers, and text collections, concluded that the performance of classifiers when using stems or roots for features outperformed that when using only words except for the works of [108], [154] and [135]. Kanaan, et al [108] and Mesleh, [135] papers that showed degrading effects used only light stemming whereas the others used light stemmers and root extractors. However, Said, et al [154] work used for stemming and root extraction two different systems: a) Al-Stem (for stems) and Sebawai (for roots) [49], [50] b) both RDIMORPHO3 stemmer and root extractor [28]. Results show: 1- using Al-Stem with either Mutual

Information (MI) or Information Gain (IG) enhances TC performance for small sized dataset, 2- using words leads to worst TC performance in small datasets while in large datasets its performance was among the best, 3- Al-Stem gave better TC performance results than RDI stemmer while RDI root extractor gave better TC performance results than Sebawai one. It should be noted that none of the works mentioned above reported significance tests to provide a conclusion whether for improvement/degradation or no effect.

2.4.1.2.2 Dimensionality Reduction by Term Selection

In the literature, various methods [87] are used for feature subset selection (abbreviated as FSS). Simple Filters are usually used for large number of features and are basically either function based on Information theory or based on term or document frequency or based on using embedded approaches.

For Arabic Information theory-based methods used for FSS are as IG, Cross entropy for text, MI, Chi-square (χ^2), NGL [140] and GSS coefficients [82] (named after the initials of their founders) (NGL and GSS coefficients are two variants of χ^2), and Odds Ratio (OR), whereas the ones based on simply term frequency are as in [95] or document frequency (threshold DF) as in [20]. However, [161] using the first two approaches for simple filters are computationally easier alternatives.

Many of the reported Arabic TC methods used one or more of the well known FSS methods as DF, χ^2, NGL, IG, OR, MI and GSS but mostly used DF or χ^2. However, for Arabic TC only one work that is known in [95] used Singular Value Decomposition (SVD) method for FSS. In two other research works [137], [184], two different optimization methods were used for that purpose.

DF is an effective global and simple method that is used to select the features with highest values among others. Examples of works that used DF for Arabic are as

[19], [20], [154], [171] and [184]. However, using DF in some of these studies didn't provide highest TC performance compared with using other FSS methods. As for applying such FSS techniques on Arabic texts: a) IG was applied in [107], [171], [137], [146], [151], [154] and [148], b) MI was used in [137], [154] and [136], c) χ^2 was used in [135], [137], [136], [7], [98], [174], [146], [147], [148] and [184] works, d) OR was used in [171], [137] and [136], e) NGL coefficient was used in [171], [137] and [136], and f) GSS coefficient was used in [171], [137] and [136] works. Yet, to the best of our knowledge, only the works of [171], [136], [184], [146], [147], [148], [154] and [137] investigated and compared the effect of FSS methods on classification performance (for further details regarding these works, kindly refer to appendix I). However, it is not possible here from the literature mentioned above to conclude which FSS method(s) provides best performance for Arabic TC. This is so for two reasons: a) such studies were conducted on different text collections, b) in above studies, the results of applying FSS methods were rather near in values and no significance tests were reported.

The results of the above comparative works indicate that using χ^2, NGL or GSS separately improved TC performance better than others. However, comparative works on Arabic gave contradictory results regarding the effect of OR on TC compared with those on English. Also, it is noticed that using optimization techniques for FSS outperformed the other Information-theoretic ones on Arabic.

2.4.1.2.3 Dimensionality Reduction by Term Extraction

Term extraction is basically [161] a method that attempts to generate a set $|T'|$ formed of synthetic terms such that $|T'| << |T|$ in order to maximize the effectiveness of a classifier. There are two major methods for term extraction which are term clustering and Latent Semantic Indexing (LSI).

Term clustering is a process by which features [117] with high degree of pair wise semantic relatedness are grouped so that their representative would be used instead of them as features in VSM. There are two types of clustering methods that have been studied: 1- one-way clustering, and 2- co-clustering. As far as we know, there is no research work that implements term clustering methods for Arabic TC.

LSI is a statistical [87] technique that attempts to estimate the hidden content structure within documents where it uses SVD, and discovers statistically most significant co-occurrences of terms. LSI was used for the unsupervised induction of MSA verb classes in [168]. Another use of LSI for Arabic was by Brants, et al [37] for topic analysis and segmentation. However, for TC it was implemented by Zukas and Price[21] where it reported an accuracy of 97% when LSI was used for TC.

2.4.2 Applied Text Classification Techniques on Arabic Sources

TC is [87] a three stage process. These stages are: 1- pre-processing stage, 2- the classification stage where usually ML techniques (mainly supervised) are used, and 3- the evaluation stage. In the past few years more ML techniques have been applied on Arabic for TC.

For Arabic texts VSM is mainly used for document representation. In [86], TC methods were used to enhance an Arabic IR system. The work of Al-Kabi and Al-Sinjilawi [10] investigated different measures to classify Arabic texts as Cosine, Jaccard, Dice, and inner product measures, then compared their results with those of using NB and Euclidean distance. Its results showed that NB surpasses the five measures and among those five measures, the cosine measure provides best results.

The effect of pre-processing step on TC performance was discussed above. In [161] the classification stage, besides the classifier used, the proposed corpus's size

[21] A. Zukas and R. Price, "Document Categorization using Latent Semantic Indexing", Found at: http://www.contentanalyst.com/images/images/Categorization_LSI.pdf [last accessed 1/11/2011]

used for classification task, its training/validation and testing sets ratios are important factors that affect the performance of TC. The training set (*Tr*) is used to train the classifier and the validation set (*Tv*) is used for fine tuning its internal parameters, while the test set *Te* is used for evaluating the effectiveness of the classifier. This is called the *train-and-test* approach. Other approaches are the *k-fold cross-validation* and the *hold-out* approaches.

The evaluation [161] of classifier performance is done through its effectiveness which is the ability to take the right classification decisions. Effectiveness is thus usually measured by Precision (*P*), Recall (*R*), Accuracy (*A*), and/or Error (*E*). Precision and Recall [161] measures cannot be looked into separately, so a combination of their effect is used by: 1- the *eleven-point average precision*, 2- the *breakeven* point, or 3- the F_β function. The above measures' equations are as shown from the contingency matrix in appendix I according to: 1- micro-averaging, 2- macro-averaging. These two methods might provide different results depending on if the number of documents per category is the same. Next is a brief description of ML methods used for Arabic TC.

2.4.2.1 Specific Machine Learning Techniques

For Arabic, relatively few of ML methods have been used. Examples of supervised ML techniques applied for Arabic TC are as: 1- decision trees [8], [94]; 2- statistical as n-grams [112] or maximum entropy [70], [159] 3- Artificial Neural Network (ANN) [95], [93], 4- distance-based [60], 5- association rule mining [27], 6- profile-based as Rocchio classifier [171], [108], and 7- more recently Rule-based as RIPPER [6]; [175]. Most of the above mentioned works are further presented in appendix I. Also, few other ML methods were implemented more often for Arabic as

parametric-based methods such as NB, example-based as k-NN, and SVM. These last three methods will be described in the coming subsections.

2.4.2.2 Naïve Bayes Classifier

NB is a simple probabilistic [92] classifier based on applying Bayes' theorem. It is a powerful, easy and language independent method. When NB classifier is applied in order to choose a class for a test document among predefined classes, equations presented in appendix I are used.

NB classifier was investigated on Arabic in works as [90], [89], [7], [108], [10], [135], [174], [146], [147], [148], [71], [141], [34], [104] and [61] as shown in details in appendix I, where in many of these studies, *tfidf* was used for weighting, whereas χ^2, DF, and IG and/or using stemming were used for FSS. It is noteworthy that such papers used different text collections and training-testing ratios, thus this classifier's performance varied among such works in a wide range from 0.73 to 0.94 for F1-measure. However, the highest performance reported for this classifier (F1 = 0.9369) is in the work of Hadi, et al [90] which used a small corpus of 600 texts under 6 classes with 70%-30% training – testing ratios.

2.4.2.3 Example-Based Classifiers (k-NN Classifier)

K-NN is a [92] statistical learning algorithm. It is a simple yet very efficient example-based approach for TC. Many parameters affect its performance such as the similarity measure (as Cosine, Euclidean, Jaccard, and Dice measures) and the choice of the number of nearest neighbors (k).

When k-NN classifier was applied for Arabic as in [135], [63], [62], [98], [20], [90], [173], [19], [109], [171], [34], [107], [104] and [61] (detailed info are shown in appendix I), about 23% of such research works did not state the distance measure used. Also, about 38% of these works used the cosine measure and the remaining

works used either Euclidean, Dice, and/or Jaccard measures. However, about 30% of these works did not specify the value of k. Other issues regarding these studies are that such papers used different text collections, FSS methods and training-testing ratios and thus the k-NN classifier performance using F1 varied among such works from about 0.70 for small corpus size to about 0.90 for much larger corpora. The highest reported F1 value [19] using this classifier was 0.96. This work used Cosine similarity, DF and light stemming, k = 21, and a small corpus of 621 texts under 6 classes, and a 90%-10% training-testing ratio.

2.4.2.4 Support Vector Machines Classifier

SVM's [105] principle is based on the structural risk minimization principle from computational learning theory. The idea is to find a hypothesis *H* for which the lowest true error is guaranteed (i.e. by searching for the maximum marginal hyper plane). A separating hyper plane can be found using $W.X + b = 0$, where $\boldsymbol{W}$ is a weight vector. SVMs learn either linear threshold or nonlinear (kernel) threshold function(s). Examples of nonlinear functions are as polynomial classifiers, radial basic function (RBF) networks, and three-layer sigmoid neural nets. However, using nonlinear [92] threshold functions is expensive. One remarkable [105] property of SVMs is that their learning ability is independent of the dimensionality of the feature space but depend on the number of training documents. Available SVM software online are TinySVM[22], GIST[23] SVM, SVM^{light}[24]and WEKA's Sequential Minimal Optimization (SMO)[25].

SVM classifier was investigated by few research works for Arabic (detailed info of works applying SVM classifier on Arabic are shown in appendix I) where 33.3%

[22] Found at: http://chasen.org/~taku/software/TinySVM [last accessed 1/11/2011]
[23] Found at: http://svm.sdsc.edu/cgi-bin/nph-SVMsubmit.cgi [last accessed 1/11/2011]
[24] SVM^{light} for single-class TC is found at: http://svmlight.joachims.org/ and for multiclass TC is found at: http://svmlight.joachims.org/svm-multiclass.html [last accessed 1/11/2011]
[25] Further info on SMO can be found at: http://weka.sourceforge.net/doc/weka/classifiers/functions/SMO.html [last accessed 7/6/2012].

of such works [135], [136], [137]; used the same in-house collection composed of 1,445 documents with 9 classes, training – testing ratios (2/3-1/3), weighting using *tfidf*, TinySVM tool and varied in using different FSS methods and its results for F1 varied between 0.74 for no FSS to **0.896** for ACO. Also, another 33.3% of works applying SVM on Arabic used 7,000 documents with 7 classes, applied both χ^2 and IG for FSS and various stemming techniques for DR [146], [147] and [148]. These works found that for 2,000 features maximum F1 value was obtained (about **0.92**) when using 3-gram for stemming and χ^2. However, the highest reported F1 value of 0.982 was in the work of Hmeidi, et al [98], which used GIST SVM, *tfidf* for weighting, local χ^2 for FSS, 2,237 texts and 98.6%-1.3% training-testing ratio.

2.4.2.5 Comparison between Classifiers' Performances

The most popular [161] classification methods were implemented for English on Reuters 20 newsgroups text collection in an attempt to compare their performance. Such implementation was performed for the following: a) total number of documents is 12,902, b) number of training documents is 9,603, c) number of test documents is 3,299, and d) the number of categories is 90. Results showed that value of F1 for the following classifiers: 0.795 using NB classifier, 0.794 using C4.5 classifier, 0.856 using k-NN classifier, 0.870 using SVM classifier, and 0.878 using boosted tree classifier.

Since there is no bench mark corpus for Arabic, it is not possible to conclusively decide if performance of classifiers' on Arabic is comparable with that for English. Only few classifiers were implemented for Arabic as mentioned above. Many classifiers that are not used for Arabic TC such as Ensemble of Nested Dichotomies that is used in this thesis in Chapter 5 and is described fully in Chapter 6.

2.5 Summary

Although, lexical-based approaches are expected to provide better results, yet these approaches require access to available lexical database(s), dictionaries, and/or thesaurus or building them from scratch. Since such approaches were not available for this thesis at the time of building our root extraction methods, and since building such methods is time consuming, it was decided to work with other techniques. It should be noted that since Buckwalter's stemmer provided the lowest performance results among others [120], [121], [160], it was decided not to use it. Although applying rule-based techniques is expected to provide good results, yet building and implementing it is also time consuming. However, their building will expectedly take less time compared to that of lexical-based approaches.

At the time of building the root extraction techniques in this work some of well known stemmers as the Khoja's stemmer were not accessible. Also, although statistical or ML-based techniques provided higher accuracy results than rule-based or lexical-based methods, nevertheless these techniques will not be used/implemented here since such methods: 1- are not available online, 2- require a relatively large annotated corpus which is not available here. Finally, since the effect of vowelized words had a rather small effect on root extraction performance as presented above, it was decided not to apply vowels to words in MSA texts. Thus, only two methods for root extraction will be implemented here. These are a rule-based method [1] and a weight-based one [17].

The rule-based technique that is used here will be explained in chapter 4. It was developed by Al-Ameed [1] where it was reported to have accuracy greater than 90%. However, since it did not handle many irregular cases as *weak*, *geminated*, and *hamzated* words, it was enhanced by addressing such cases as will be shown in

Chapter 4 (i.e. by developing and adding the ***Correction*** algorithm that handles these cases to the rule-based one). Also, in comparison with the enhanced rule-based technique, a weight-based (also named here positional-letter-ranking) approach is also investigated [17] along with proposing and implementing an adjustment and two expanded weight-based methods in Chapter 4. The choice of this technique was due to the fact that it is simple to implement and its reported accuracy value in [17] is about 90%. Also, the ***Correction*** algorithm is added to all four weight-based techniques and its effect on the performance of such techniques is shown in Chapter 4. The technique with best accuracy results is used further on to represent texts in terms of their normalized words, stems or roots and investigates which of such representations improves best TC techniques as is described in Chapter 5. However, if a stemmer doesn't remove efficiently/correctly prefixes and/or suffixes then the remaining analysis to extract the root of such word would produce the wrong root. This is called here the prefix-suffix paradigm.

The following was concluded for Arabic TC:

1- regarding document representation and term weighting: a) for document representation VSM was used and for feature choices only words, stems, roots were separately used, b) for term weighting methods, *TFIDF* was used frequently and compared with others in terms of their effect on TC performance, but it was not reported if such variation is statistically significant. The *idf* part of *TFIDF* function defined above handles the effect of the term globally. So the local effect of this part is not tested for Arabic TC.

2- for DR methods implemented and their effect on Arabic TC performance: a) stemming was investigated but there were no reports of significance tests to validate their results; b) term selection methods were investigated and it can be concluded that when using DF in some research studies TC performance wasn't the highest and it is not possible to conclude which FSS method(s) provides best performance for Arabic TC since different text collections were used, and results of applying FSS methods were rather near in values and no significance tests were

reported; c) term extraction techniques there is no research work that implemented any of term clustering methods for Arabic, whereas LSI was used for TC on Arabic with a reported accuracy of 97%.

3- When applying NB, k-NN, and SVM classifiers on Arabic, no conclusive result of their performance on Arabic can be provided since different text collections, different FSS methods, and training-testing ratios were used. The performance of NB classifier varied in a wide range from 0.73 to 0.94 for F1. K-NN classifier's performance using F1 varied from about 0.70 to about 0.90. SVM classifier's performance using F1 varied from about 0.74 to about 0.986. However, recent results of research works presented in section 2.4.2 that compared between those three classifiers indicate that performance of SVM classifier is highest followed by k-NN then by NB ones.

4- Few clustering methods have been implemented for Arabic such as in [12], [159], [85], [21], and [109]. However, since the scope of this thesis is to investigate classification methods for Arabic, clustering techniques will not be discussed.

5- Only a few classifiers were implemented for Arabic as mentioned above. There is no investigation in the literature of the effect of representing texts by phrases for Arabic (whether alone or combined with words) on TC performance. The intent of this thesis is to study the effect of including phrases as features on TC and compare the performance of many well performing classifiers as will be presented in Chapter 5 and discussed in Chapter 6.

Chapter 3: The Development of an Arabic Text Corpus and Pre-processing Steps

3.1 Introduction

The first aim of the work reported in this chapter is to present and describe two new single-labeled Arabic text collections designed hereby to support the forthcoming research on pre-processing and classifier performance study, to introduce a comprehensive label set that unifies labeling of the two text collections, and to integrate these two collections into one final corpus with an aim to use its texts in the implementation of: a) root extraction techniques presented in Chapter 4, and b) single-label TC techniques presented in Chapter 5. The second aim is to present preprocessing steps necessary for both root extraction and TC methods. This requires the handling of several issues such as removing function words, diacritics, non Arabic alphabet and digits among others. Such requirements are presented in this chapter whereas other preprocessing steps are presented in Chapter 5 for TC.

The investigation for text labelling is performed in order to classify a series of texts into one domain from eight specified general domains namely *politics, economics, social issues, sports, music, religious issues, 'arts, literature, and culture'*, and finally *'educational, science, and health'*. The first Arabic Text collection under 8 classes (AT8) is gathered by the author of this thesis and contains 380 texts only, while the second one is downloaded from Leed's University website. Leed's Arabic Contemporary Corpus (LACC) contains 424 texts only. Since the number of texts in each collection is small, the need to incorporate both into one final corpus and unifying their classes is evident in order to acquire better performance results for

both root extraction and TC methods where the final corpus is a better representative than each collection of available Arabic texts on the web written in MSA.

This chapter is organized as follows: Section 3.2 describes the first Arabic text collection. In Section 3.3, the second Arabic text collection is described, where its re-categorization process is presented in order to unify labels for both collections among the eight domains mentioned above. The final corpus' single-labeling results are also presented in Section 3.3. Pre-processing steps needed, for both root extraction techniques and TC ones, are described in Section 3.4. Finally, conclusions are presented in Section 3.5.

3.2 Description of Newly Gathered Text Collection

In order to support implementing root extraction techniques to be discussed in Chapter 4, we have built up a new collection of Arabic texts. This was performed by acquiring randomly from various online Arabic newspapers, academics, magazines and other sources published online in the period 23/7/2008 - 1/2/2009. This collection was presented in notepad text files (UTF-8).

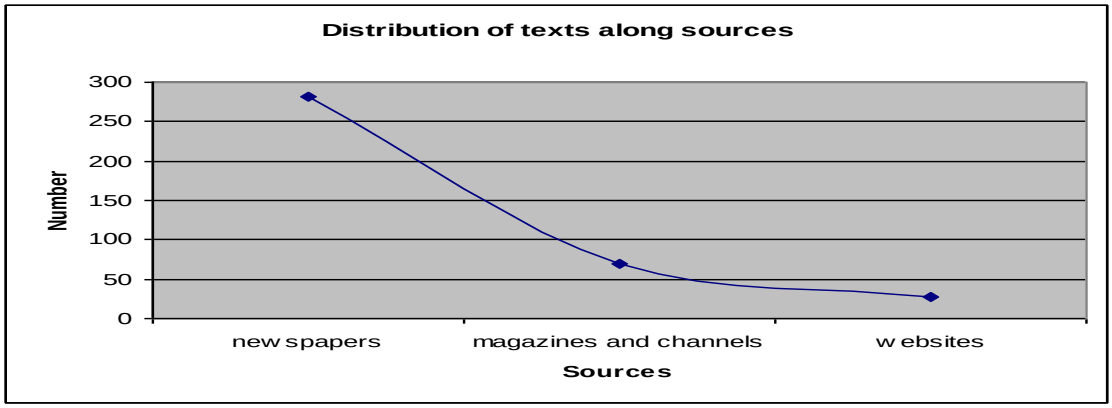
3.2.1 Newly Gathered Text Collection

The AT8 collection is gathered randomly according to eight general subject domains as stated in section 3.1. In each domain close to 50 texts were chosen randomly with a total of 380 texts in all 8 domains (about 200,000 words). On average, the number of words per text is about 526 words. These domains were chosen here in such a way that: 1- these domains were in general chosen by the text's respective websites, 2- these domains would contain articles, short stories … etc. Also, two of these domains, which are the *educational, science and health* domain or the *arts, literature and culture* domain, were chosen each containing three topics. This was performed

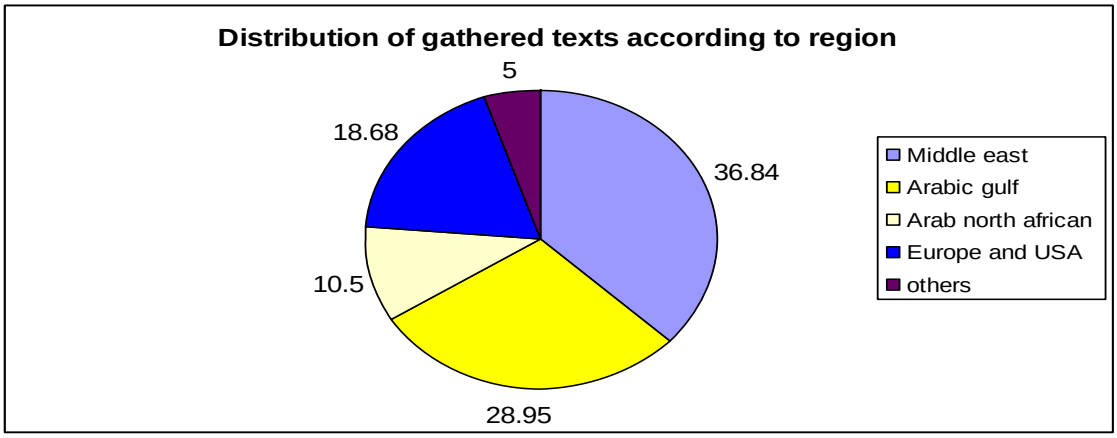
since although different, yet these three topics are related, and the number of texts that is gathered from websites for each individual topic is comparatively low. The actual number of text documents in each domain is shown in Table 1. Furthermore, the list of source websites and number of texts chosen from each are shown in Table 2. The distribution of these texts along three different major source categories, namely 'newspapers', 'magazines and channels', and 'other' websites as well as different geographical regions are shown in Figure 3. 'Other' websites source category presents in general personal websites that are constructed by individuals.

No.	Domain	# words	# files
1-	Politics	27,164	50
2-	Economics	22,516	45
3-	Religious issues	28,538	51
4-	Social issues	21,562	38
5-	Sports	22,266	61
6-	Educational, science and health	25,538	43
7-	Arts, literature and culture	33,518	50
8-	Music	12,180	42
	Total	***193,282***	***380***

Table 1: Number of Texts and words in AT8 collection



(a)



(b)

Figure 3: Distribution of texts in AT8 along (a) three major sources, (b) region

Website of	Country	Region	Texts no	Texts
Addustour newspaper	Jordan	Middle East region (36.84%)	54 (14.2%)	12
Al-Rai newspaper	Jordan			31
Alarabalyawm newspaper	Jordan			7
UOP	Jordan			4
Al-Anwar newspaper	Lebanon		53 (13.95%)	3
Al-Intiqad newspaper	Lebanon			1
Assafir newspaper	Lebanon			7
Annahar newspaper	Lebanon			13
Alhayat newspaper	Lebanon			27
Almustaqbal newspaper	Lebanon			2
Furat-alwehda newspaper	Syria		6 (1.6%)	3
Jamahir-alwehda	Syria			3
Al-Sabar magazine	Palestine		20 (5.3%)	3
Alyaum newspaper	Palestine			2
Al-ayyam newspaper	Palestine			2
Alhayat-jadida newspaper	Palestine			1
Alquds newspaper	Palestine			12
Fasl-almaqal newspaper	Israel		7 (1.8%)	7
Akhbar-alkhaleej	Bahrain	Arabic Gulf region (28.95%)	99 (26.05%)	1
Ommandaily newspaper	Omman			5
Al-sharq newspaper	Qatar			3
Aljazeera Channel	Qatar			28
Alqabas newspaper	Kuwait			13
Alwatan newspaper	Kuwait			7
Alkhaleej newspaper	Emirates			1
Akhbaralarab newspaper	Emirates			1
Kul-alwatan newspaper	Saudi Arabia			5
Okaz newspaper	Saudi Arabia			7
Al-madina newspaper	Saudi Arabia			11
Al-jazirah newspaper	Saudi Arabia			1
Al-Riyadh newspaper	Saudi Arabia			8
Asharqalawsat newspaper	Saudi Arabia			8
Azzaman newspaper	Iraq		6 (1.6%)	6
Arabiya	MBC news channel		5 (1.32%)	5
Alaswaq	Al-Arabia channelsatellite	Europe and USA	40 (10.5%)	1
Alquds-alarabi newspaper	UK			10
BBC Arabic channel	UK			16
CNN channel	USA			13
Al-fadjr newspaper	Algeria	Arab North African region (18.68%)	46 (12.11%)	23
Al-Alam newspaper	Morocco			3
Alkhabar newspaper	Algeria			13
El-massa newspaper	Algeria			2
Assaheefa newspaper	Libya			1
MAP news agency	Morocco			4
Al-Ahram newspaper	Egypt		25 (6.6%)	12
Al-Gomhuria newspaper	Egypt			5
Al-Wafd newspaper	Egypt			1
Arabnet	Egypt			5
Watani	Egypt			2
Alarab online	UK	Others (5%)	(4.21%)	16
Maktoob	Jordan		(0.26%)	1
Jeeran	Saudi Arabia (4.21)		(0.26%)	1
Hazemsakeek	-		(0.26%)	1

Table 2: Source Websites and their Number of Texts in AT8 collection

Despite this text collection effort, it is observed that the AT8 text collection remained small in size whether in terms of the number of text files or words and did

not represent a sufficient corpus for both the root extraction and TC methods. It is determined that combining the newly collected text set with the existing collection at the University of Leeds, would provide a comparatively large enough corpus for the root extraction and TC methods.

3.3 Description of University of Leeds Arabic Contemporary Corpus

LACC corpus was downloaded on January 2010 and will be described in detail. Section 3.3.1 discusses the properties, components of LACC. The re-categorization of LACC is performed in Section 3.3.2 as well as a comparison between the two text collections in terms of their domains.

3.3.1 Original Categorization of LACC

LACC corpus that is available online [25] is presented here where its written texts were originally put into 15 categories as shown in Table 3. Such texts were put in XML mark-up as raw UTF-8 text files (except ScienceB category where its texts were put in notepad text files (also UTF-8)) that contained many details such as title, original publishing organization, author name(s), date of publication, number of words …etc.

No.	Original Category	# files	# words	% words
1-	**Politics**	*10*	*44,590*	**5.03**
2-	**Autobiography**	*72*	*151,687*	**17.13**
3-	**Economics**	*28*	*66,354*	**7.49**
4-	**Religion**	*19*	*111,199*	**12.56**
5-	**Short stories**	*31*	*46,884*	**5.294**
6-	**Sociology**	*30*	*88,577*	**10.002**
7-	**Tourism and Travel**	*60*	*46,093*	**5.21**
8-	**Recipes**	*9*	*4,972*	**0.56**
9-	**Sports**	*4*	*8,809*	**0.995**
10-	**Education**	*10*	*24,674*	**2.79**
11-	**Health and Medicine**	*32*	*40,480*	**4.57**
12-	**Science**	*45*	*105,206*	**11.88**
13-	**Interviews**	*23*	*56,428*	**6.37**
14-	**ScienceB**	*25*	*67,720*	**7.65**
15-	**Children's stories**	*26*	*21,958*	**2.48**
	Total	***424***	***885,632***	

Table 3: Number of Texts and words in LACC corpus

The target users [25] of this corpus are language teachers, language engineers, foreign learners of Arabic and material writers. Table 4 illustrates for each original website name, country, region, number of files and words in LACC. Figure 4 briefly illustrates the contents of this table.

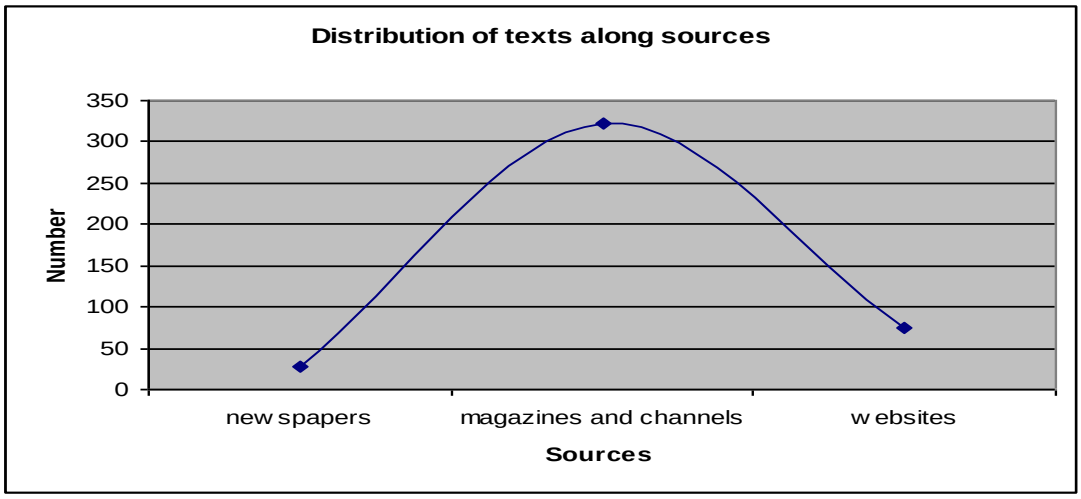
No.	Magazine, newspaper, website name	Country	Region	# files	% files	# words	% words
1-	Alarabi magazine	Kuwait		138	**32.55**	353,171	**39.88**
2-	Radio Qatar	Qatar		2	**0.47**	1,771	**0.2**
3-	Alrai Alaam	Kuwait		13	**3.07**	61,592	**6.96**
4-	Islamonline website	Qatar	Gulf Area	45	**10.61**	135,037	**15.25**
5-	Lahaonline website	Saudi Arabia	**(84.86%)**	13	**3.07**	2,926	**0.33**
6-	Economic world Magazine	Saudi Arabia		83	**19.58**	86,501	**9.77**
7-	Islam-online website	Qatar		8	**1.89**	35,078	**3.96**
8-	Al Marefah	Saudi Arabia		10	**2.36**	24,674	**2.79**
9-	Akalaat website	UAE		8	**1.89**	4,620	**0.52**
10-	Arabic Story	Bahrain		30	**7.08**	45,831	**5.18**
11-	Arab Medical Magazine	Lebanon		27	**6.37**	34,395	**3.88**
12-	Ofouq	Syria	Middle	12	**2.83**	21,667	**2.45**
13-	Al Hourriah	Syria	East **(6.85%)**	2	**0.47**	4,639	**0.52**
14-	Sayidaty Magazine	UK		7	**1.65**	5,599	**0.63**
15-	BBC	UK	Europe **(0.68%)**	1	**0.24**	411	**0.05**
16-	Science And Technology Magazine		**(7.65%)**	25	**5.90**	67,720	**7.65**
	Total			***424***	**-**	***885,632***	**-**

Table 4: Source Websites and their Number of Texts in the LACC corpus

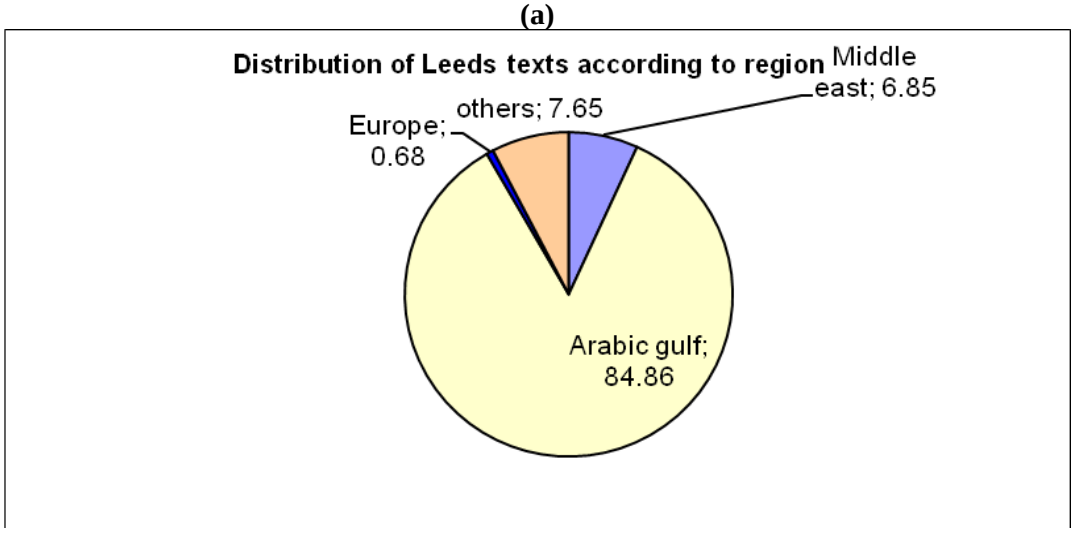
The small size of both collections led us to include both in one corpus. Yet, the difference in the type of domains between the two collections led us to investigate which type of domains to use. Figure 4a shows that large percentage of texts came from magazines and Figure 4b shows that about 85% of texts were provided from the Arabian Gulf region. This corpus has 424 texts and about 900,000 words. On average, the number of words per text in LACC is about 2,089 words.

From Tables 1 and 3, the domains in LACC are different than those in AT8 collection. This is due to the fact that, among other reasons, the target users in both collections are different. Also, a comparison between Tables 2 and 4 as well as Figures 3 and 4 shows that AT8 collection is more spread around the regions of Arab speaking countries compared to LACC. This is due to the fact that LACC was limited by the number of websites (publishers) that accepted that their texts in LACC

be available online. Yet, the number of words in LACC is far more than AT8 collection.



(a)



(b)

Figure 4: Distribution of texts in LACC along (a) three major sources, (b) regions

No.	Original Domain	# files	# words	Major Domain	% words	$g_{\Omega}(c_i)$
1-	Politics	*10*	*44,590*	Politics	5.03	2.36
2-	Autobiography	*72*	*151,687*			
3-	Short stories	*31*	*46,884*	Arts	24.9	30.42
4-	Children's stories	*26*	*21,958*			
5-	Economics	*28*	*66,354*	Commerce	7.49	6.6
6-	Religion	*19*	*111,199*			
7-	Sociology	*30*	*88,577*	Social sciences	25.35	13.92
8-	Education	*10*	*24,674*			
9-	Tourism and Travel	*60*	*46,093*	Leisure	6.21	15.09
10-	Sports	*4*	*8,809*			
11-	Recipes	*9*	*4,972*	Life	6.93	7.55
12-	Interviews	*23*	*56,428*			
13-	Science	*45*	*105,206*	Natural sciences	11.88	10.61
14-	Health and Medicine	*32*	*40,480*	Applied sciences	12.22	13.44
15-	ScienceB	*25*	*67,720*			
	Total	***424***	***885,632***			

Table 5: Percentage of Texts and words in LACC corpus under major domains

3.3.2 Re-Categorization of LACC and Final Corpus Categorization

From Table 3, genre of domains in LACC is not as that in AT8 collection. In order to use both collections for TC, their texts should be under the same genre of domains. It was decided first to re-categorize the texts in LACC such that each text fits into only one domain among eight domains of AT8 collection as shown in Table 6.

New Domain Index	No.	Original Domain	# files	# words	% words	% Words - combined
1-	1-	**Politics**	*10*	*44,590*	**5.04**	**8.00**
	2-	**Autobiography for politics**	*13*	*26,226*	**2.96**	
2-	3-	**Economics**	*28*	*66,354*	**7.49**	**7.49**
3-	4-	**Religion**	*19*	*111,199*	**12.55**	**15.08**
	5-	**Autobiography for religion**	*13*	*22,440*	**2.53**	
4-	6-	**Sociology**	*30*	*88,577*	**10.00**	**15.4**
	7-	**Tourism and Travel for Social**	*60*	*46,093*	**5.21**	
	8-	**Autobiography for social**	*1*	*1652*	**0.19**	
5-	9-	**Sports**	*4*	*8,809*	**1.00**	**1.00**
6-	10-	**Education**	*10*	*24,674*	**2.79**	**27.86**
	11-	**Health and Medicine**	*32*	*40,480*	**4.57**	
	12-	**Science for educational**	*45*	*105,206*	**11.88**	
	13-	**Autobiography for Educational**	*6*	*8,617*	**0.97**	
	14-	**ScienceB for Educational**	*25*	*67,720*	**7.65**	
7-	15-	**Recipes for Arts**	*9*	*4972*	**0.56**	**23.69**
	16-	**Short stories for Arts**	*31*	*46,884*	**5.29**	
	17-	**Children's stories for Arts**	*26*	*21,958*	**2.48**	
	18-	**Interviews for Arts**	*22*	*56,011*	**6.32**	
	19-	**Autobiography for Arts**	*33*	*80,023*	**9.04**	
8-	20-	**Interviews for music**	*1*	*417*	**0.05**	**1.49**
	21-	**Autobiography for music**	*6*	*12,729*	**1.44**	
		Total	***424***	***885,632***	-	-

Table 6: Number of Texts and words in LACC corpus after Re-categorization

Also, a comparison between the two collections is shown in terms of number of files as in Table 7 and in terms of words as in Table 8. It is clear from these two tables that number of files or words are not evenly distributed among domains in both collections, i.e. their generality, although in LACC, this is clearer. However, as is shown below, after combining LACC with AT8 collection, the generality of the final corpus is more similar among domains using AT8 scheme.

No.	Category	# files, LACC	$g_\Omega(c_i)$, LACC	# files, AT8	$g_\Omega(c_i)$, AT8	# files, final corpus	$g_\Omega(c_i)$
1-	**Politics**	23	*5.42*	50	*13.16*	73	**9.08**
2-	**Economics**	28	*6.60*	45	*11.84*	73	**9.08**
3-	**Religion**	32	*7.55*	51	*13.43*	83	**10.32**
4-	**Social**	91	*21.46*	38	*10*	129	**16.04**
5-	**Sports**	4	*0.94*	61	*16.05*	65	**8.08**
6-	**Educational,**	118	*27.83*	43	*11.32*	161	**20.02**
7-	**Arts, Culture**	121	*28.54*	50	*13.16*	171	**21.27**
8-	**Music**	7	*1.65*	42	*11.05*	49	**6.10**
	Total	***424***	-	***380***	-	***804***	-

Table 7: Comparison between LACC and AT8 collections in terms of number of files according to AT8 categorization scheme

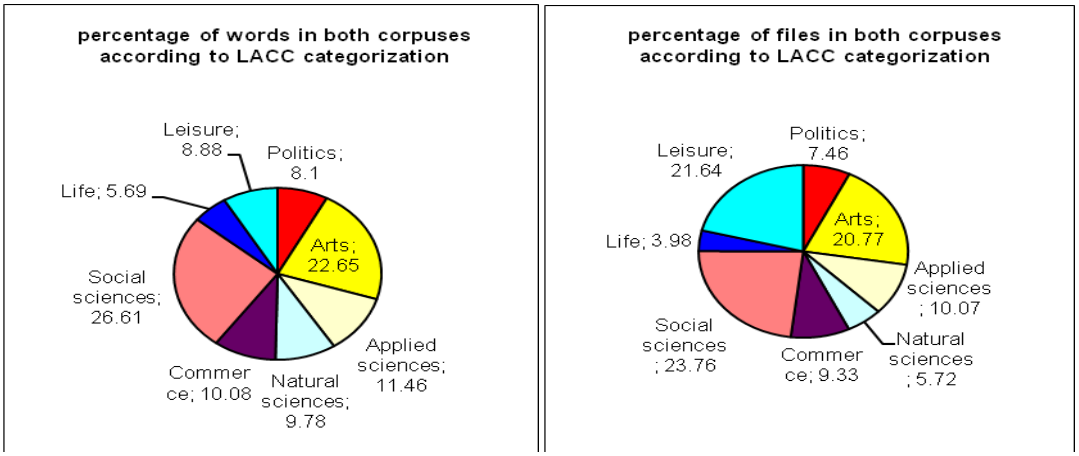
No.	Category	# words, LACC	% words, LACC	# words, AT8	% words, AT8	# words, final corpus	% words, final to total
1-	**Politics**	70,816	*72.28*	27,164	*27.72*	97,980	**9.08**
2-	**Economics**	66,354	*74.66*	22,516	*25.34*	88,870	**8.24**
3-	**Religion**	133,639	*82.40*	28,538	*17.6*	162,177	**15.03**
4-	**Social**	136,322	*86.34*	21,562	*13.66*	157,884	**14.63**
5-	**Sports**	8,809	*28.56*	22,266	*71.65*	31,075	**2.88**
6-	**Educational, health and medicine**	246,697	*90.62*	25,538	*9.38*	272,235	**25.23**
7-	**Arts, Culture and Literature**	209,849	*87.17*	33,518	*13.77*	243,367	**22.56**
8-	**Music**	13,146	*51.91*	12,180	*48.09*	25,326	**2.35**
	Total	***885,632***	**-**	***193,282***	**-**	***1,078,914***	**-**

Table 8: Comparison between LACC and AT8 collections in terms of number of words according to AT8 categorization scheme

Also, AT8 collection was re-categorized according to the general domains assigned in [163] and a comparison between LACC and AT8 collection in that regard is shown in Table 9.

Domain	AT8 collection				LACC				Final corpus			
	# file	$g_\Omega(c_i$	# word	% word	# file	$g_\Omega(c_i$	# word	% word	# file	$g_\Omega(c_i)$	# word	% word
Politics	*50*	*13.16*	*27,164*	*14.05*	*10*	*2.36*	*44,590*	*5.03*	*60*	**7.46**	**71,754**	**8.1**
Arts	*38*	*10*	*23,891*	*12.36*	*129*	*30.42*	*220,529*	*24.9*	*167*	**20.77**	**244,420**	**22.65**
Appl science	*24*	*6.32*	*15,432*	*7.98*	*57*	*13.44*	*108,200*	*12.22*	*81*	**10.07**	**123,632**	**11.46**
Nat science	*1*	*0.26*	*308*	*0.16*	*45*	*10.61*	*105,206*	*11.88*	*46*	**5.72**	**105,514**	**9.78**
Commerce	*47*	*12.37*	*22,876*	*11.84*	*28*	*6.6*	*66,354*	*7.49*	*75*	**9.33**	**89,230**	**10.08**
Social sciences	*110*	*34.74*	*62,696*	*32.44*	*59*	*13.92*	*224,450*	*25.34*	*191*	**23.76**	**287,146**	**26.61**
Life	-	-	-	-	*32*	*7.55*	*61,400*	*6.93*	*32*	**3.98**	**61,400**	**5.69**
Leisure	*110*	*28.95*	*40,915*	*21.17*	*64*	*15.09*	*54,902*	*6.2*	*174*	**21.64**	**95,817**	**8.88**
Total	***380***	***47.26***	***193,282***	**-**	***424***	***52.74***	***885,632***	**-**	***804***	**-**	***1,078,914***	**-**

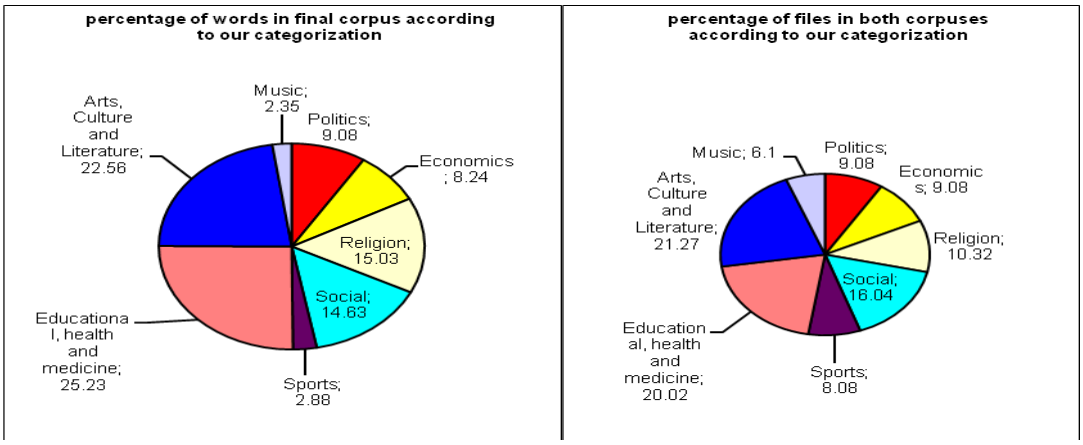
Table 9: Comparison between LACC and AT8 collections in terms of number of files and words according to LACC categorization scheme



(a) **(b)**

Figure 5: Percentage in final corpus according to LACC categorization scheme for (a) words, (b) files

Figures 5 and 6 illustrate the information provided in Tables 7 & 8 regarding the effect of both categorization schemes on final corpus.



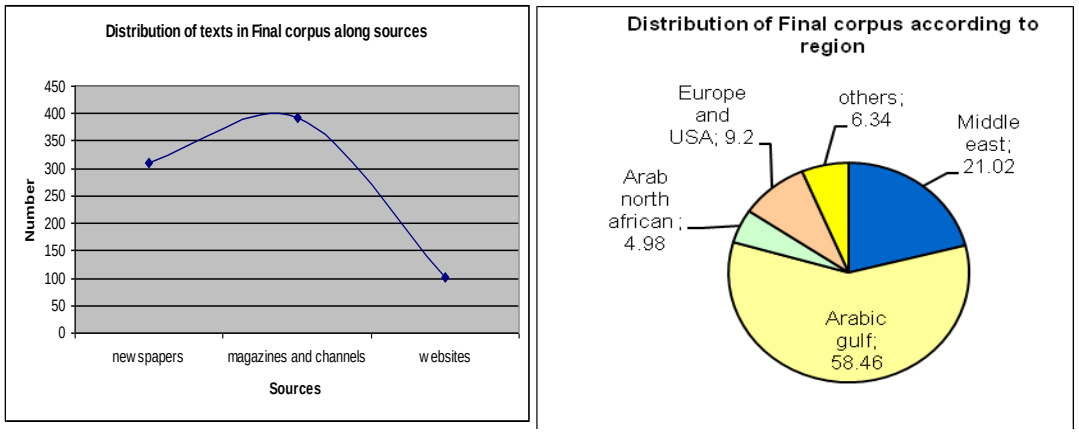
(a) **(b)**

Figure 6: Percentage in final corpus according to AT8 collection categorization scheme for (a) words, (b) files

Percentages in Tables 7 and 9 as well as in Figures 5 and 6, for both categorization schemes illustrate that the generality among suggested domains is not consistent. Also, when using the LACC categorization scheme no or nearly no texts were available in the two domains *life* and *natural sciences* as in the AT8 collection.

The distribution of texts in the final corpus along three major source categories and according to geographical regions is shown in Figure 7. Figure 7a illustrates that large percentage of texts came from both magazines and newspapers sources and

Figure 7b illustrates that about 60% of texts were provided from the Arabian Gulf region. Also, on average, the number of words per text in final corpus is about 1,342 words. Thus, we have decided to use the AT8 collection categorization scheme. The final corpus characteristics are presented in Table 10. The final corpus categorization is used for single-label TC in Chapter 5.



(a) **(b)**

Figure 7: Distribution of texts in final corpus along (a) three major sources, (b) region

Table 10 presents the number of files in each category of the final corpus and as can be seen from generality values that the corpus is less skewed than before. It should be noted that through the process of manual re-categorization. Many texts could be classified into more than one domain. This was further investigated by distributing a questionnaire to native Arabic speakers requesting them to classify attached texts with one or two classes among predefined eight classes to develop the first multi-labelled Arabic corpus that is briefly presented in Chapter 7 for future work.

Domain	# files	$g_{\Omega}(c_i)$
Politics	73	*9.08*
Economics	73	*9.08*
Religion	83	*10.32*
Social	129	*16.05*
Sports	65	*8.08*
Educational, science and health	161	*20.02*
Arts, Culture and Literature	171	*21.27*
Music	49	*6.1*
Total	***804***	-

Table 10: Final Corpus' number of files and generality among classes

Usually Arabic texts are available in MSA and do not contain words with short vowels, nunnation, kasheeda or assimilation markers. Thus, in the pre-processing stage before applying root extraction or TC techniques such markers are removed (if available in the text) along with punctuation marks, function words, digits and English letters. Next, the construction of function words list is presented as part of the pre-processing stage.

3.4 Pre-processing Steps

3.4.1 Arabic Function Word List Construction

From a non-linguistic point of view, a function word is a word [87] that does not carry information. It has mainly a functional role and is usually removed in TM methods to help the methods to perform better.

Here we present the Arabic function words list that is formed from 2,549 words [96]. Examples of function words are the separate prepositions, personal pronouns, demonstrative pronouns, interrogative pronouns, relative pronouns, conjunctions, and interjections as shown in Table 11 as well as in appendix I. Imperfect verbs such as *kAn wAxwAthA* standing for the verbs 'was and its sisters' were included in the function word list along with similar verbs such as *OSbH*, *mAzA*l, or *OmsY*. Also, words as *Ontm*, *mvlhm*, *Elyhn*, which are derivations from *Ont*, *mvl*, *ElY* respectively, whether for dual or plural forms, are added to function word list. The function word list constructed here is used in both Chapters 4 and 5 when preparing texts for root extraction or TC methods by removing function words from texts in the final corpus.

Arabic function word	Transliterated	Stands for
'إلى'	*'IlY'*	to, unto, until
'على'	*'ElY'*	Over, on, or against
'أنا'	*'OnA'*	I
'نحن'	*'nHn'*	We
'أنتَ'	*'Onta'*	masculine single you
'هن'	*'hn~'*	plural feminine they
'هذا'	*'hA*A'*	masculine single this
'هٰتان'	*'hAtAn'*	feminine dual this
'منْ'	*'mn'*	Who
'ما' 'ماذا'	*'mA*A', 'mA'*	What
'الذي'	*'Al*y'*	Who
'و'	*'w'*	AND
'أو'	*'Ow'*	OR

Table 11: Some examples of Function Word list

3.4.2 Arabic Text Pre-processing

From Figure 8, the first process before performing root extraction or TC methods is to remove from texts English letters, punctuation marks, nunations, assimilation markers, short vowels, kasheeda, function words or numerals (either Hindi or Arabic). This is performed for all texts in final corpus in preparation for applying root extraction and/or TC methods.

3.5 Conclusions

Two Arabic text collections are fully described and manually classified into one final corpus and their labeling is unified under eight general classes. This corpus is composed of 804 files and about a million words. It is prepared for the implementation of root extraction techniques as shown in Chapter 4, and single-label TC techniques as shown in Chapter 5. The generality differs among the final corpus's eight classes. This difference is expected to affect TC results presented in Chapter 5.

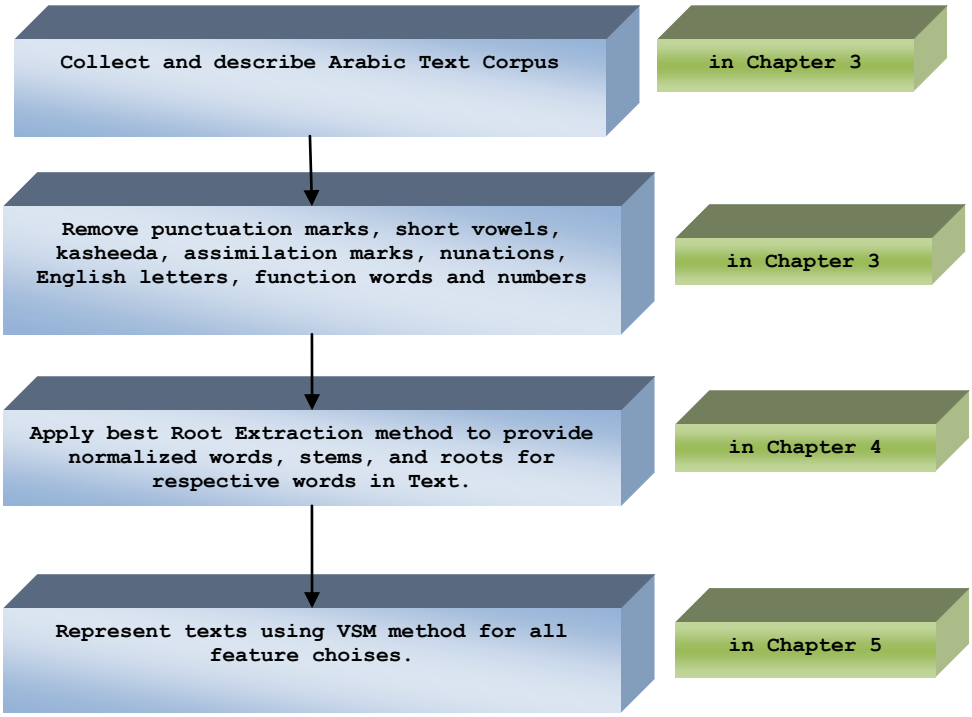


Figure 8: Preprocessing Steps before Arabic TC

Chapter 4: The Development of an Arabic Root Extraction System

4.1 Introduction

In this chapter, the focus will be on investigating two different Arabic stemming techniques, improving them and finally comparing their performances. The two approaches for stemming used here are based on the works of Al-Ameed's [1] and Al-Shalabi, et al [17]. The first approach is a rule-based one [1]. The choice of the rule-based technique was because it was reported in the original work to have an accuracy value higher than 90% [1]. The second approach is a weight-based technique that is introduced by Al-Shalabi, et al [17] technique. The choice of this technique was because it is simple, easy to implement and had a reported 90% accuracy. However, in [17] no information was provided for the reasons of choosing the weight and rank values for the alphabetical letters.

In this Chapter, a proposed adjustment method to the weight-based technique described in [17] and two enhancement methods (named ***Expanded Weight Based Method1*** (***EWBM1***) and ***Expanded Weight Based Method2*** (***EWBM2***)) are implemented here. Such contributions have also been reported by the author in [14]. The two original approaches [1, 17] presented here do not handle *weak* words, names of places, countries, cities, months, foreign Arabized words, *geminated* words (except for the rule-based one in section 4.2 where geminating is partially handled), or broken plurals (except for the rule-based one in section 4.2). In both approaches, the concentration of affix removal is on the letters in *sOltmwnyhA*.

Since the two approaches used here do not handle irregular words, then the first contribution of this thesis is through proposing and implementing the ***Correction*** algorithm (as investigated in [13]). This algorithm is included here into all root extraction algorithms and its effectiveness in improving their performance is investigated. The results of implementing these techniques will be compared with those of a rule-based approach thoroughly investigated by the author of this thesis and reported in [14].

Figure 9 summarizes the contributions provided by all implemented root extraction techniques (described in coming subsections), and demonstrates how original algorithms were incorporated and enriched in this thesis. The comparison between these techniques is performed according to two criteria: 1- accuracy, and 2- execution time.

The final contribution here addresses the case of handling foreign Arabized words and names of places, countries, cities, and months by developing a list of such cases as described in section 4.5. This list is incorporated in a final proposed root extraction system that is presented in Figure 28 at the end of this chapter. Figure 28 briefly summarizes the effort of the author to combine the best features of proposed root extraction techniques that handle *weak, eliminated-long-vowel, hamzated,* and *geminated* words, the best choices of investigated normalization lists, and extracting quadriliteral roots (proposed in ***EWBM2*** method) and as such presents the first contribution. It also includes the second contribution by handling foreign Arabized words. Thus, Figure 28 presents an effort to combine root extraction algorithms in an overall approach.

The remainder of this chapter is organized as follows: in Section 4.2 the rule-based approach is presented along with the contribution for correcting irregular words through our proposed ***Correction*** algorithm. In Section 4.3, the four weight-based techniques are presented. Section 4.4 presents and analyzes the evaluation criteria and experimental results for implementing all techniques. In Section 4.5, the list of foreign Arabized words and names of places, countries, cities, and months is constructed and presented. Section 4.6 presents the final proposed root extraction system. Finally, Section 4.7 discusses conclusions and future work.

4.2 Rule-Based Approach

In this part, the concentration will be on investigating/improving a rule-based light stemmer/root extractor technique on Arabic based on the work of Al-Ameed [1]. Al-Ameed method was chosen here since it reported an accuracy of root extraction of more than 90% when tested on many derivations of many roots. However, Al-Ameed's method was not designed to handle irregular words in the Arabic language. Irregular words represent a significant portion of words used in standard text (about 34%)[26]. This limitation in Al-Ameed's method is addressed in this thesis by introducing an enhanced method, based on Al-Ameed original approach, which properly handles irregular words during the root extraction step without degrading the performance of the original rule-based method.

The performance of the original method by Al-Ameed and the performance of the enhanced method in handling irregular words in Arabic such as *weak, two-letter geminated, hamzated, and eliminated-long-vowel* words, is evaluated using first the AT8 text collection and then LACC collection. Furthermore, their efficiencies (based

[26] Percentage values presented here are gathered from 40 texts chosen arbitrarily in the collection.

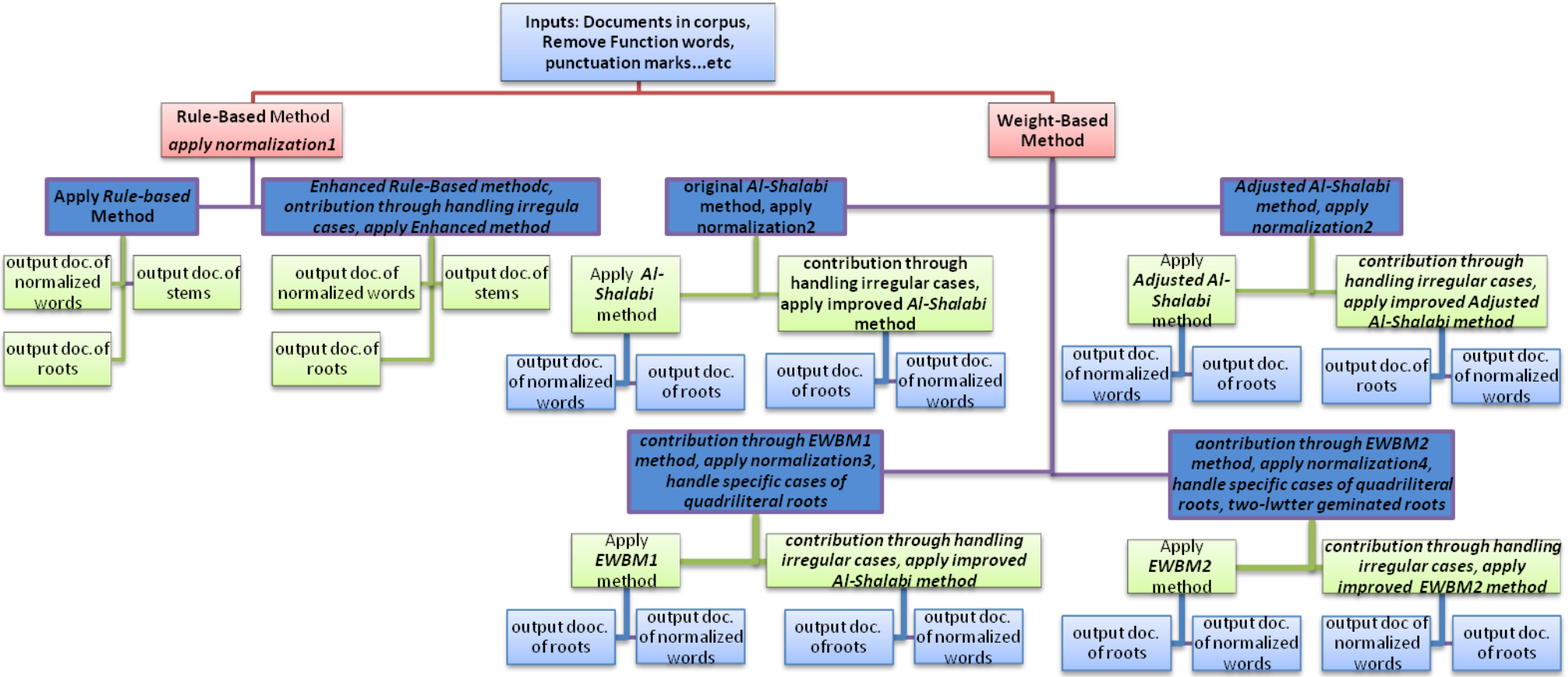


Figure 9: A brief illustration of implemented root extraction techniques

on execution time) are analyzed. A similar set of evaluation steps are also performed for the weight-based approach, which will be discussed in detail at a later stage in this chapter.

Next a brief description of the rule-based approach is presented.

4.2.1 Description

The rule-based root extractor is implemented starting from the work of Al-Ameed [1] and is composed of two parts. The first part is a rule-based light stemmer where prefixes and suffixes are removed from the word also according to specific rules. The second part is a pattern-based infix remover where infixes are removed from the word according to also specific patterns. These two parts represent the entire original algorithm (named here the ***Rule-Based*** algorithm). As cited in Al-Ameed's work, this algorithm was tested against the work of Chen and Gey [44] only due to the fact that the later work gave better results than both Darwish's and Larkey's works [49], [121]. In Al-Ameed's work [1], the analysis of the performance of the two algorithms (Chen and Gey algorithm versus Al-Ameed one) showed that Al-Ameed's algorithm gave much better performance results. However, Al-Ameed's ***Rule-Based*** algorithm does not handle irregular words, resulting in substantial percentage errors during root extraction. Thus, a new algorithm that handles such cases is presented in Section 4.2.2 when it is added at the end of the ***Rule-Based*** algorithm in order to enhance its performance. In Section 4.2.3, the results of this ***Rule-Based*** root extraction approach will be compared with that of the enhanced ***Rule-based*** technique (i.e. with the ***Correction*** algorithm included).

As can be seen from the algorithm below, the rule-based technique outputs, besides to normalized words file, two files: the first file contains the stems, and the second file contains the roots.

Rule-Based Algorithm

```
Inputs: Set of preprocessed documents D = {d1, d2, …..,dn},
Predefined root lists
Outputs: triliteral and quadriliteral roots for each new document
new_di_2 in output set DD2, stems for each new document new_di_1 in
output set DD1

START
1- For each document di do {
2-    LastWord = Count_No_Words(di)
3-     For j = 1 to LastWord in di do {
4-      LastLetter = Count_No_Letters(wj,c), m = 0
5-      If (LastLetter <= 3) then {Final_Wordj = wj, m = 1, go to *}
% output normalized words to an output document to be used later for
TM
6-      New_Wordj = Normalize(wj)
7-      Write New_Wordj to output document new_di_1
% Perform light stemming algorithm for word wj
8-      New_Wordj = Light_Stemmer_Algorithm(wj)
9-      LastLetter = Count_No_Letters(New_Wordj,c)
10-     Write New_Wordj to output document new_di_2 %output stems to
a different document to be used later for TM
11-     If (LastLetter <= 3) then {Final_Wordj = New_Wordj, go to *}
% Perform Infix Remover algorithm
12-     New_Wordj = Inf_Remover_Algorithm(New_Wordj)
13-     LastLetter = Count_No_Letters(New_Wordj,c)
14-     Write Final_Wordj to output document new_di_3 % output roots
to an output document to be used later for TM
15- *      if m == 1 then {Write Final_Wordj to output document
new_di_1}
% calculate the accuracy of algorithm
16- count = Count_Correct_Roots(Final_Wordj, count)}
17- Accuracy_of_document_ new_di_2 = (count/LastWord) 100% }
END
```

To illustrate the performance of the algorithm above, an example of the outcome of each part in it is provided here. For the word استحسانهم transliterated *"AstHsAnhm"*[27], this word becomes after the light stemmer part in step 8 is performed حسان *"HsAn"* and after the infix remover part is performed in step 12 حسن *"Hsn"* which is the correct root for that word.

4.2.2 *Enhanced Rule-Based* Technique

We contribute to enhance the ***Rule-Based*** approach by proposing an algorithm to correct irregular words as presented in [13]. This proposed algorithm handles:

1- *weak* words by replacing the long vowel in it by another long vowel according to specific rules in Arabic,

27 Arabic letters and words are presented using Buckwalter's transliteration which is available in appendix III.

2- *eliminated-long-vowel* words where, for specific cases of these triliteral words, when for example their tense is changed from past to present tense, the vowel is cancelled and an extra letter is added to that word (whether at beginning or end),
3- *two-letter geminated* roots as *"rd"* when the word starts/ends with either '*y*, *t*, *n*, or *A'*/ '*t*, *p*, or *h'* respectively the extra letter must be deleted and the letter '*d'* is doubled,
4- specific cases of *Hamza* (if present) in a root is corrected.

The proposed ***Correction*** algorithm includes 5,737 possible corrections of words in 71 predefined lists (collected from references [31], [33] according to specific rules for only triliteral roots (see detailed flowchart for algorithm and samples of lists in appendix II)).

The accuracy of the ***Rule-Based*** algorithm is calculated by first comparing its extracted roots with a predefined list of triliteral and quadriliteral list of 5,405 roots (4,655 triliteral roots and 750 quadriliteral roots) gathered from [31], [33], [67] (see sample root lists in appendix II), then counting the roots that match the ones in the predefined list, and finally calculating the percentage of correctly extracted roots. The same applies when the ***Correction*** algorithm is added at the end of the ***Rule-Based*** approach. In other words, when our ***Correction*** algorithm is added at the end of the ***Rule-Based*** approach, the extracted root is checked whether in the predefined root list. If not, the extracted root is checked if triliteral and if any of the rules in the ***Correction*** algorithm apply for it. Finally, if the extracted root belongs to a predefined list for a specific rule, then the root is corrected to the proper one and then accuracy is calculated as above.

As can be seen from the ***Correction*** algorithm and Figure 10 that not only rules were used to specify each case but also in following the rule the word was compared with a predefined root list of words that do indeed follow that rule in Arabic. This additional step was performed in order to minimize the effect of extracted roots where the rule apply but are not the correct ones. Examples of words that this

algorithm handles are **يصل** *"ySl"*, **تلد** *"tld"*, **قال** *"qAl"*, or **تمت** *"tmt"*, these words become after performing this algorithm **وصل** *"wSl"*, **ولد** *"wld"*, **قول** *"qwl"*, or **تمم** *"tmm"* respectively.

Correction Algorithm

```
Inputs: Arabic triliteral word, 71 predefined lists
Output: corrected triliteral Arabic word

START
1- Let ch1 <- first character of Word; ch2 <- second character of
Word; ch3 <- third character of Word
% handling 27 weak cases, some eliminated-long-vowel cases (6 cases
for pattern yEl, one case for yfE) and 18 hamzated word cases (e.g.
y$O becomes $A`, some are composite with either weak or eliminated-
long-vowel cases) (total of 47 different rules)
2- If ch1 is either y, t, &, A, n, or } { % (hamzated, eliminated-
long-vowel cases, or both)
3-      if word is in specific lists {
4-      change ch1 according to specific cases, go to *.}}
5- If ch3 is either y, Y, &, }, w, or A { % (weak, eliminated-long-
vowel, hamzated cases)
6-      if word is in specific lists {
7-        change ch3 according to specific cases, go to *.}}
8- If ch2 is either }, w, O or A { % (weak, eliminated-long-vowel,
hamzated cases)
9-      if word is in specific lists {
10-      change ch2 according to specific cases, go to *. }}
% handling geminated words (2 different rules)
11-If ch1 is either " t, y, n, or A {
12-    if word is in specific lists {
13-     delete ch1 and double ch3 according to specific cases, go to
*}}
14- If ch3 is either h, or p {
15-     if word is in specific lists {
16-       delete ch3 and double ch2 according to specific cases, go
to *}}
% handling one geminated, 6 hamzated (some are composite with
eliminated-long-vowel cases) or eliminated-long-vowel cases (18
cases for pattern fEt, 4 for flt) (23 different rules)
17- If ch3 is t {
18-     if word is in specific lists {
19-     either
20-         delete ch3 and double ch2 according to specific cases
%(geminated cases)
21-         OR replace ch3 by only one of letters A, y, w, or Y
according to specific rules % (eliminated-long-vowel & hamzated
cases)}}
22- *  Return corrected word
END
```

It should be mentioned here that during the construction of the ***Rule-Based*** method, function words were removed before the ***Rule-Based*** algorithm was implemented and also at its middle (i.e. after light stemming is performed). However, from

preliminary experiments, it was found that this second step removes words that are not function words. Thus, it was decided to remove function words only once before implementing this algorithm. The results of implementing this proposed algorithm are presented next.

4.2.3 Results of Implementation

Al-Ameed method [1] reported accuracy for root extraction of more than 90% when tested on many derivations of many roots. Al-Ameed's algorithm was tested there using a specially customized test set which was composed of 199,584 distinct words derived from 24 distinct triliteral roots and 119,700 words derived from 25 distinct quadriliteral roots. Since this test set is not available to us, such accuracy values could not be verified. However, we used the AT8 collection (described in section 3.2) to test both Al-Ameed's algorithm and the ***Enhanced Rule-Based*** technique.

The experimental results of the accuracy for the ***Rule-based*** approach and the ***Enhanced Rule-Based*** technique are presented. In Table 12 and Figure 11 the following stand for:

RB: ***Rule-Based*** algorithm, **Enh_RB**: **Enhanced *Rule-Based with Correction*** algorithm (in some figures and tables it is abbreviated RB_corr)

Results in Table 12 show that adding our proposed ***Correction*** algorithm to the ***Rule-based*** approach increased the latter's accuracy by about 14% and relatively improved it by about 23%. Also, bolded values in Table 12 present maximum accuracy values whereas italic ones present minimum accuracy values.

Category	RB (%)	Enh_RB (%)
Politics	58.89	73.3
Economics	58.16	71.39
Religious issues	**62.99**	**75.01**
Social issues	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 12: Performance of *Rule-Based* and *Enhanced Rule-Based* algorithms in all categories using AT8 collection

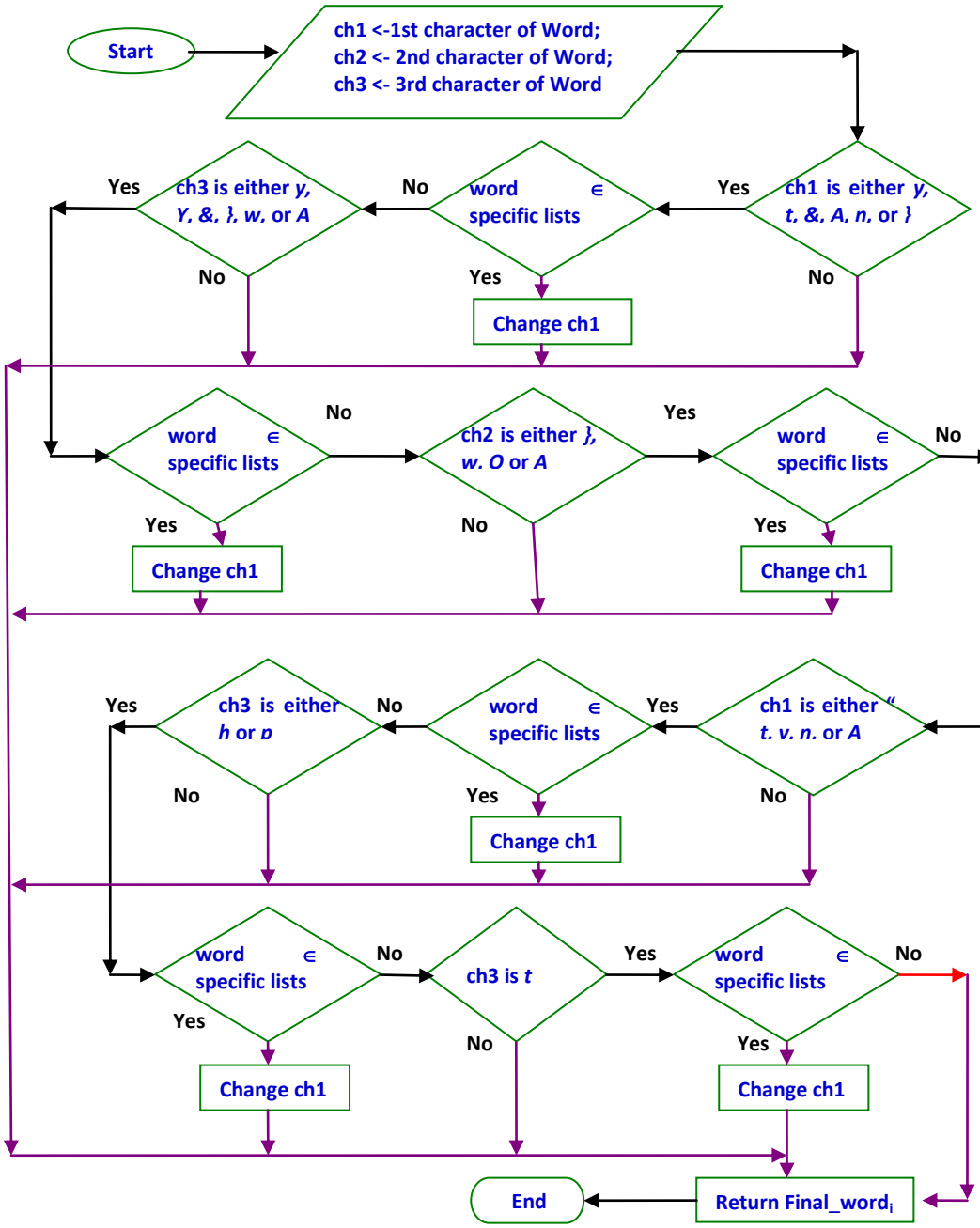


Figure 10: Flowchart of *Correction* Algorithm

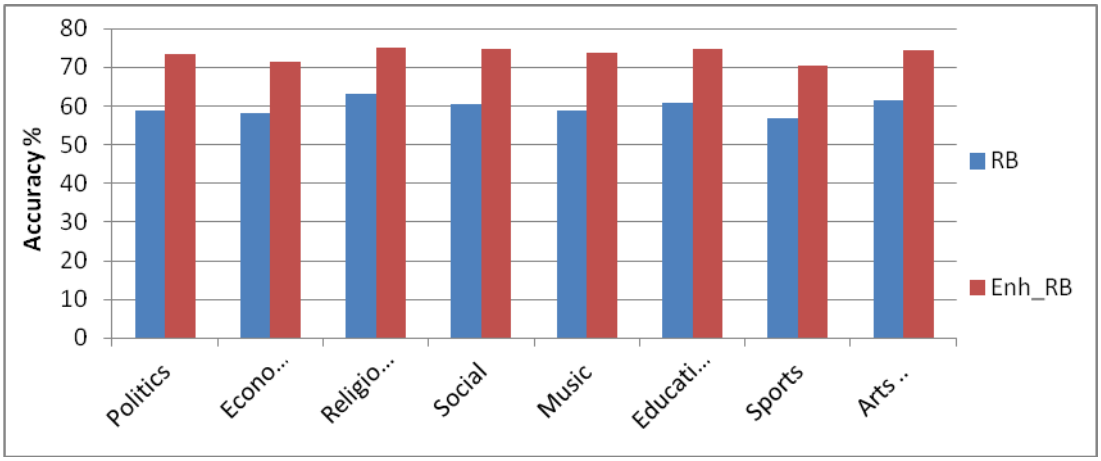


Figure 11: Performance of *Rule-Based* and *Enhanced Rule-Based* algorithms.

At the start of this part, the intention was to use Buckwalter's stemmer[28] for comparison. However, Sawalha and Atwell [160] reported that it used Buckwalter's stemmer along with two other stemmers (Khoja's algorithm [111] and Al-Shalabi weight-based [17] one). Buckwalter's stemmer provided lowest accuracy values among stemmers used. The main limitations of ***Rule-Based*** approach were: 1- a rather limited number of patterns used, 2- *two-letter geminated* roots were not extracted as a first step, 3- couldn't handle prefix-suffix dilemma completely and efficiently although it used the most available prefixes and suffixes. In general, the performance of proposed ***Correction*** algorithm can be increased by adding further rules and restrictions.

4.3 Weight-Based Approach

In this part, the main purpose is to use and propose variants of a weight-based approach to extract roots of words in texts as a preprocessing step for TC and to compare the results of such techniques with those of the ***Rule-based*** one explained in section 4.2. The weight-based work proposed by Al-Shalabi, et al [17], named here ***Al-Shalabi***, will be described in section 4.3.1, and a slight adjustment to it (***Adjusted Al-Shalabi***) will be described in section 4.3.2. The contribution here is through proposing two variants (***Expanded Weight Based Method1*** (***EWBM1***) and ***Expanded Weight Based Method2*** (***EWBM2***)) [14] that will be explained thoroughly in sections 4.3.3 & 4.3.4 respectively.

The above techniques test at the beginning if the number of letters in the word is less than or equal to 3 and if so take the word, except for ***EWBM2*** technique, without any further processing. ***EWBM2*** technique tests if a two-letter word is *geminated* by

28 Buckwalter stemmer version 2.1 is found at URL: http://www.qamus.org/

comparing it to a two-letter *geminated* words list. If it is in the list, the ***EWBM2*** technique presents the two-letter word as a triliteral root by doubling its second letter. Also, ***EWBM1*** & ***EWBM2*** techniques extract specific cases of quadriliteral roots along with triliteral ones whereas ***Al-Shalabi*** and ***Adjusted Al-Shalabi*** techniques extract only triliteral roots. Section 4.3.5 presents the outcome of implementing these techniques. Next is a description of ***Al-Shalabi*** algorithm.

4.3.1 Description of *Al-Shalabi* Algorithm

Al-Shalabi algorithm [17] employs a letter weight, an order index and assigns a rank to a letter according to its order in the word. It extracts the root for the word through the following simple steps: 1- for each letter in the word (from right to left) it applies weight and rank values according to Tables 13 and 14 while assigning order values, 2- calculate the product of the rank and weight for each letter, then 3- keep only the letters with the first three smallest product values without changing their order in the word. In order to illustrate the steps of this algorithm, two examples of words are shown in Table 15 where the least three product values are bolded. As shown in Table 13, the rank of a word is calculated differently when its number of letters is odd from that when it is even (an example showing ranking is presented in Table 15). The weights of letters are numerical values provided for letters categorized into groups (e.g. allocating the group of letters '*p, A*' a weight of 5) as shown in Table 14. Al-Shalabi, et al [17] work did not explain or clarify why or on what basis it used such ranking or weighting only that such groups and their values were chosen after extensive experimentation. For a native Arabic speaker it is understood why the letters *p, A* are given a high weight (compared to others) since the letter *p* appears at the end of a word and is a suffix and the letter *A* appears at any position in a word where it is also a prefix, infix or suffix in most cases. Also, it is noted that letters *b, f,*

k are considered among the 'Rest' group, which means that these letters will always be considered by the algorithm as original. Yet, such letters, if present at the beginning of a word, might be extra letters. The two examples shown in Table 15 are presented separately where the algorithm first provides an order value for each letter, a weight value, a rank value then calculates the product and finally takes the three letters in this word with least product values. ***Al-Shalabi*** algorithm was implemented with/out the ***Correction*** algorithm, and in section 4.3.5 the results of its implementation is shown and analyzed.

Letter position	Rank (if word length: even)	Rank (if word length: odd)
1	N	N
2	N – 1	N – 1
3	N – 2	N – 2
:	:	:
⌈N/2⌉	N/2 + 1	⌈N/2⌉
⌈N/2⌉ + 1	N/2 + 1 – 0.5	⌈N/2⌉ + 1 – 1.5
⌈N/2⌉ + 2	N/2 + 2 -0.5	⌈N/2⌉ + 2 – 1.5
⌈N/2⌉ + 3	N/2 + 3 -0.5	⌈N/2⌉ + 3 – 1.5
:	:	:
N	N – 0.5	N – 1.5

Where N: number of letters in a word

Table 13: Letter ranking in *Al-Shalabi* algorithm *(derived from [17])*

Letters	*A, p*	*y, }*	*t, w, Y*	*O, I, m, n*	*l, s, h*	Rest
Weight	5	3.5	3	2	1	Zero

Table 14: Weights of letter groups in Al-Shalabi algorithm *(derived from [17])*

letters	***h***	***m***	***A***	***d***	***x***	***t***	***s***	***I***
Order	8	7	6	5	4	3	2	1
Weight	1	2	5	0	0	3	1	2
Rank	7.5	6.5	5.5	4.5	5	6	7	8
Product	7.5	13	27.5	**0**	**0**	18	**7**	16
Root	*sxd* (X)							

a) word *IstxdAmh*, correct root *xdm*

Letters	*t*	*A*	*m*	*Y*	*l*	*E*	*t*	*l*	*A*
Order	9	8	7	6	5	4	3	2	١
Weight	3	5	2	3.5	1	0	3	1	5
Rank	7.5	6.5	5.5	4.5	5	6	7	8	9
Product	22.5	32.5	11	15.75	**5**	**0**	21	**8**	45
Root	*lEl* (X)								

b) word *AltElymAt*, correct root *Elm*

Table 15: Examples of extracted roots using *Al-Shalabi* algorithm *(from right to left)*

4.3.2 Adjustment of *Al-Shalabi* Algorithm

It was noticed in [17] that there was a discrepancy in some of its examples. The two examples that caused such discrepancy were the ones when the letter *l* was at first or second position in a word where the authors have assigned it a weight of 5. However, it was given a weight of 1 when it was in other positions (as was specified in their paper for the weight of this letter). This information was not explained or mentioned throughout that paper except only in the two examples. So, here this is considered as an adjustment (named ***Adjusted Al-Shalabi***) by implementing it and investigating its accuracy while maintaining the rest of the procedure mentioned in [17]. Thus, the same ranking, weighting and ordering of letters in a word was maintained, except that for letter *l* a different weight of 5 was given if it was in the first or second position in the word. Following this adjustment for the weight of the letter *l* when applied on the same two examples in Table 15, the expected extracted roots would be *sxd*, *Elm* respectively. ***Adjusted Al-Shalabi*** algorithm was implemented with and without the ***Correction*** algorithm. In section 4.3.5, the results of its implementation is shown and analyzed.

As can be seen from the examples in the previous two techniques, it is expected that these algorithms will not extract roots with high accuracy. However, since this approach is very simple and easy to implement, then proposing a different weighting scheme for the groups of letters might produce higher accuracy results. This is on the basis of taking into consideration the characteristics of Arabic language letters. This led us to look for any specific percentages of occurrences for letters in texts. Throughout the process of searching for information regarding these letters, statistics showing the percentages of such Arabic letters were found[29]. After close

29 From Khaled AlShamaa web site, URL: http://www.al-shamaa.com/php/arabic/index.html, [last accessed: 4/6/2010]

examination of these percentages and including the effect of the number of letters before and after them as shown in Table 16, it was not possible to quantitatively reach a weight for these letters or classify them into separate distinct groups.

However, it was possible to do so qualitatively: 1- At a first analysis, it was proposed that these letters be grouped into five groups (as ***Al-Shalabi*** algorithm or its adjustment) where such groups are assigned classes: *high, high or moderate, moderate, moderate or low*, and finally *low*. A *high* class contained letters *p* and *A*. A *high or moderate* class contained letters *y* and *}*. A *moderate* class contained letters *t, w,* and *Y*. A *moderate or low* class contained letters *m* and *n*. Finally, a *low* class contained letters *h, l,* and *s*. 2- In a second analysis; it was proposed that these letters be grouped into four groups where such groups are assigned classes: *high, high or moderate, moderate*, and finally *moderate or low*. A *high* class contained letters *p, h,* and *A*. A *high or moderate* class contained letters *y* and *}*. A *moderate* class contained letters *l, t, w,* and *Y*. Finally, a *moderate or low* class contained letters *m, s,* and *n*. 3- Finally, at a third analysis, it was proposed that these letters be grouped into three groups where such groups are assigned classes: *high, moderate,* and finally *moderate or low*. A *high* class contained letters *p, h* and *A*. A *moderate* class contained letters *y, }, t, w,* and *Y*. A *moderate or low* class contained letters *l, m, s,* and *n*. The reason why no conclusive number of groups was reached is the nature of some of these letters and their similar percentages in appearing as extra and original letters in words.

It was not possible to reach all the weights proposed by ***Al-Shalabi*** algorithm from these statistics. However, since the initial number of groups found here are 5, weighting letters was thus given by assigning the groups weights from 5 to 1 according to classes assigned: 5 for high, 3.5 for high or moderate, 3 for moderate, 2

for moderate or low, and 1 for low. In order to further explore such different choices of the number of groups, ***EWBM1*** method (explained next) will adapt grouping these letters into four groups as shown in Table 17, while ***EWBM2*** method will group such letters into three groups as shown in Table 18.

Letter	Rate (%)	Letters no after,	letters no before,	letter after	letter before	Qualitative weight for rate values only
space	**not given**	27 84	32 100	43.02 *A*	21.9 *p*	**not a character**
A	19.65	31 96	30 94 OL	40.8 *l*	42.16	**High**
p	4.22	1 3 OL	30 84 OL	100 space	٣٨,٤١ *y*	**moderate or low**
h	1.79	14 44	22 69 OL	٤١,٠٣ *A*	١٥,٨ *l – t*	**Low**
}	0.50	10 31	6 19 OL	٤١,٥٤ *y*	52.31 *A*	**Low**
y	6.66	28 88	31 97 OL	25.49	16.07 *f*	**high or moderate**
l	12.99	30 94	29 91 OL	٢١,٢٤ *A*	٦١,٧١ *A*	**high**
t	5.64	31 97	24 75 OL	22.76	22.49 *A*	**moderate**
w	5.70	30 94	27 84 OL	١٦,٠٩ *A*	41.02	**moderate**
Y	0.91	1 3 OL	9 28 OL	100 space	٧١,٤٣ *l*	**Low**
m	8.52	30 94	25 78 OL	19.73	22.33 *l*	**high or moderate**
n	3.86%	25 78	21 66 OL	42.57	٢٨,١٢ *A*	**moderate or low**
s	2.48%	20 63	17 53 OL	٢٠ *l*	٢٨,٩٢ *l*	**moderate or low**

Where OL stands for Of Letters

Table 16: Percentages of Letter Appearances in Texts

4.3.3 First Expanded Weight Based Method

Here, it is proposed to use the same ranks of letters as that of ***Al-Shalabi*** algorithm but to assign a different set of weights to letters as shown in Table 17 in order to provide a triliteral root according to their order. The five groups of letters that were proposed in [17] have been reduced to four with shown weights. The letter *l* was moved to third group with weight 3. The letter *s* was moved to the fourth group to give it a higher value especially when at the beginning of a word (most likely it will be an extra letter but an original letter elsewhere) and finally the letter *h* was moved to the first group with weight 5 since it is expected that when *h* is at the end of the word, it is likely to be a suffix since it might be wrongly written as *h* where as it is meant to be *p*. This algorithm is called ***EWBM1*** (its flowchart is shown in appendix II). Moreover, this algorithm proposes to extract specific cases of quadriliteral-root-based words. This is performed by counting the number of zero product values in the word. If the number of zeros is greater than 3 and the number of letters in the word is

greater than or equal to 4 then this proposed variant algorithm provides the quadriliteral root of the letters (i.e. choose the least four product values keeping the order of letters maintained) else it provides the triliteral root. ***EWBM1*** algorithm shown below is a combination of the original weight-based method and rules to handle quadriliteral roots, i.e. a hybrid method.

Letters	*A, p, h*	*y, }*	*l, t, w, Y*	*O, I, m, n, s*	**rest**
Weight	5	3.5	3	2	Zero

Table 17: Weights of Letter groups for *EWBM1* algorithm

EWBM1 Algorithm

Inputs: Set of preprocessed documents D = {d_1, d_2, …..,d_n}, Predefined root lists, Predefined letter groups weight lists
Outputs: List of triliteral and some quadriliteral roots for each new document new_d_i_1 in output set DD

```
START
1- For each document d_i do {
2-   LastWord = Count_No_Words(d_i)
3-   For j = 1 to LastWord in d_i do {
4-     LastLetter = Count_No_Letters(w_j,c)
5-     If (LastLetter <= 3) then {Final_Word_j = w_j, go to *}
6-     Provide the order, weight values for each letter in word w_j
7-      Perform calculating the product of order and weight values
for each letter in word w_j
8-     Count = Count_No_Zero_Product_Letters(w_j)
%       Take least 4 product value letters keeping their order
9-     If ((Count > 3) and (LastLetter >= 4)) then
10-         {Final_Word_j = Extract_4letter_with_least_product(w_j), go
to *}
11-    Else {Final_Word_j = Extract_3letter_with_least_product(w_j)}
12- *  Write Final_Word_j to output document new_d_i_1
13- count = Count_Correct_Roots(Final_Word_j, count)}
14- Accuracy_of_document_ new_d_i_1 = (count/LastWord) 100%}
END
```

So, the original two examples in Table 15 would generate when using this proposed algorithm roots *sxd*, *Elm*. In brief, this algorithm varies from previous ones by providing different groups of letters with different weight values and extracting four-letter roots. ***EWBM1*** algorithm was implemented with/out the ***Correction*** algorithm and in section 4.3.5 results of its implementation is presented and analyzed.

4.3.4 Second Expanded Weight Based Method

EWBM2 technique uses the same ranks as described in ***Al-Shalabi*** technique. This second technique performs the following steps:

1- It excludes the letter combination *Al* from the word if the word starts with it,

2- It replaces the letters *O, I,* | with *A* only and replaces letters *}, y* with *Y* only and replaces letter *p* with *h* (i.e. a normalization step),

3- It presents specific *two-letter geminated* words as triliteral by comparing them with a predefined list of *two-letter geminated* words (provided in appendix II) and if the two-letter word is in the list, the algorithm duplicates the second letter and adds it to the word,

4- It uses a different weighting scheme from the previous three techniques as shown in Table 18,

5- It provides a quadriliteral root by counting the number of zero product values for letters in a word (other than the letter *b*) and by counting the number of repetitions a letter occurs in a word (other than the letters *b* or *w* or *A*). If the number of zeros is greater than 3 or the number of repetitions of any letter in the word is greater than 2 and the number of letters in the word is greater than or equal to 4, then it chooses the four-letter root with the least product values keeping the order of letters maintained, else it chooses the three-letter root with the least product values.

As can be noticed here, more rules were put for choosing a quadriliteral root. This is due to the fact that in some words such as *$dyd*, the letter *d* appears twice but separated by *y* and when using ***EWBM1*** algorithm, it will be considered as a correct root where it is not. The above steps are illustrated in ***EWBM2*** algorithm (its flowchart is shown in appendix II). Since this is also a combination of the original weight-based method and rules, it is then a hybrid method.

As can be seen from Table 18, the five groups of letters that were proposed in ***Al-Shalabi*** algorithm have been reduced to only three with the shown weights. Here, the second group in Table 14 is cancelled since its letters are replaced by *Y*. Also, the letters *l, m, s* and *n* are moved to the third group with weight 2, and the letters *t, w* and *Y* were moved to the second group with weight 3. ***EWBM2*** algorithm was

implemented with and without the ***Correction*** algorithm where in section 4.3.5, the results of its implementation is shown and analyzed.

Letters	*A, h*	*t, w, Y*	*l, m, n, s*	Rest
Weight	5	3	2	Zero

Table 18: Weights of Letter groups for *EWBM2* algorithm

EWBM2 algorithm

Inputs: Set of preprocessed documents D = {d_1, d_2, …..,d_n}, Predefined root lists, Predefined two-letter geminated words list, 3 Predefined Replace lists, Predefined letter groups weight lists
Outputs: List of triliteral and some quadriliteral roots for each new document new_d_i_1 in output set DD
START
1- For each document d_i do {
2- LastWord = Count_No_Words(d_i)
3- For j = 1 to LastWord in d_i do {
4- LastLetter = Count_No_Letters(w_j,c)
5- If (LastLetter < 3) then {Final_Word$_j$ = w_j, go to *}
% ***Remove Al from word (if it starts with it)***
6- w_j = Remove_AL(w_j)
% ***Replace some letters with others from word (a normalization step)***
7- w_j = Replace_letters(w_j)
8- Provide the order, weight values for each letter in word w_j.
9- Perform calculating the product of weight and order values for each letter in word w_j.
10- Count = Count_No_Zero_Product_Letters_Not_*b*(w_j) % ***count number of zeros for product values for letters other than b***
11- Repeat = Count_No_Repetitions_Not_*b_w_A*(w_j) % ***counts the number of repetitions a letter occurs in word other than the letters b or w or A***
% ***Take least 4 product value letters keeping their order***
12- If (((Count > 3) or (Repeat > 2)) and (LastLetter >= 4)) then
13- {Final_Word$_j$ = Extract_4letter_with_least_product(w_j), go to *}
14- Else {Final_Word$_j$ = Extract_3letter_with_least_product(w_j)}
15- * LastLetter = Count_No_Letters(Final_Word$_j$,c)
16- If (LastLetter == 2) then
17- {cc = Compare (Final_Word$_j$, 2_letter_list)
18- If (cc == 0) then Final_Word$_j$ = Correct_Word(Final_Word$_j$)}
19- Write Final_Word$_j$ to output document new_d_i_1
20 count = Count_Correct_Roots(Final_Word$_j$, count)}
21- Accuracy_of_document_ new_d_i_1 = (count/LastWord) 100%}
END

Also, the original two examples in Table 15 would generate using this proposed variant method roots, *xdm*, *Elm* respectively. In brief, this algorithm varies from the previous ones in that it: 1- provides different weight values for different groups of letters, 2- removes *Al* from words if these words start with it, 3- replaces specific

letters by others (a normalization step), 4- extracts *two-letter geminated* roots, and 5- extracts four-letter roots. Table 19 illustrates briefly the various weights for letters used in all weight-based algorithms as was explained in the sections above.

Letter	Rate (%)	*Al-Shalabi*	*Adjusted Al-Shalabi*	*EWBM1*	*EWBM2*
A	19.65	5	5	5	5
p	4.22				
h	1.79	1	1		
}	0.50	3.5	3.5	3.5	3
y	6.66				
l	12.99	1	5 if at beginning 2 positions of word, 1 else	3	2
t	5.64	3	3		3
w	5.70				
Y	0.91				
m	8.52	2	2	2	2
n	3.86				
s	2.48	1	1		

Table 19: Proposed weighting for Assigned Groups in algorithms

It should be noted that ***Al-Shalabi***, its adjustment, ***EWBM1*** and ***EWBM2*** techniques do not handle *weak, eliminated-long-vowel, hamzated* words, names of places, countries, cities, months, broken plurals, or foreign Arabized words (examples are presented in page 90). So, the ***Correction*** algorithm described in section 4.2.2 is added to all techniques in order to improve their performance and investigate its effectiveness.

4.3.5 Results of Implementation

Here experimental results demonstrating the accuracy of implementing the weight-based techniques with/out our proposed ***Correction*** algorithm using AT8 collection are presented. In the following tables and figures the following stand for:

S1: *Al-Shalabi* algorithm,
S2: *Adjusted Al-Shalabi* algorithm,
S3: *EWBM1* algorithm,
S4: *EWBM2* algorithm,

S1_corr: *Al-Shalabi* with *Correction* algorithm,
S2_corr: *Adjusted Al-Shalabi* with *Correction* algorithm,
S3_corr: *EWBM1* with *Correction* algorithm,
S4_corr: *EWBM2* with *Correction* algorithm.

	S1(%)	S2(%)	S3(%)	S4(%)
Politics	55.36	62.16	59.39	58.9
Economics	***52.59***	***60.73***	58.52	57.44
Religious	53.82	61.63	57.84	59.09
Social	**56.44**	63.21	59.98	59
Music	55.02	60.86	59.49	60.26
Educational	53.46	62.37	59.73	59.05
Sports	55.53	61.67	***57.58***	***57.23***
Arts ..	55.67	**63.22**	**61.38**	**60.42**

Table 20: Performance of weight-based algorithms using AT8 collection

	S1_corr(%)	S2_corr (%)	S3_corr (%)	S4_corr (%)
Politics	62.84	72.12	67.08	69.45
Economics	***58.9***	***70.41***	65.69	68.61
Religious	61.06	70.78	***64.07***	68.25
Social	**64.13**	**72.81**	67.22	69.37
Music	63.97	71.48	67.1	69.18
Educational	59.85	71.59	66.33	**70.11**
Sports	62.19	71.53	64.61	***67.46***
Arts ..	63.42	72.64	**67.89**	69.78

Table 21: Performance of weight-based with *Correction* algorithm using AT8 collection

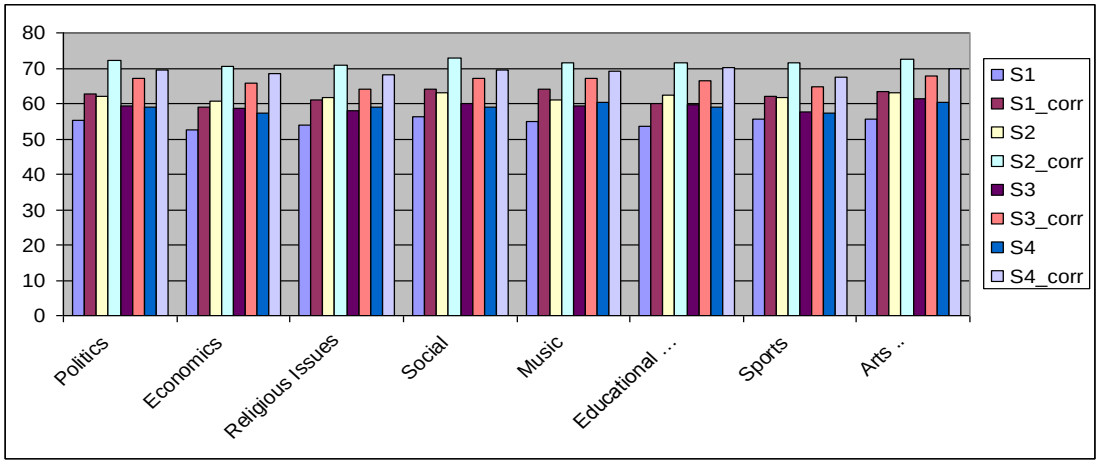


Figure 12: Comparison between accuracy results of all weight-based algorithms in all categories with the ones incorporating the *Correction* algorithm using AT8 collection

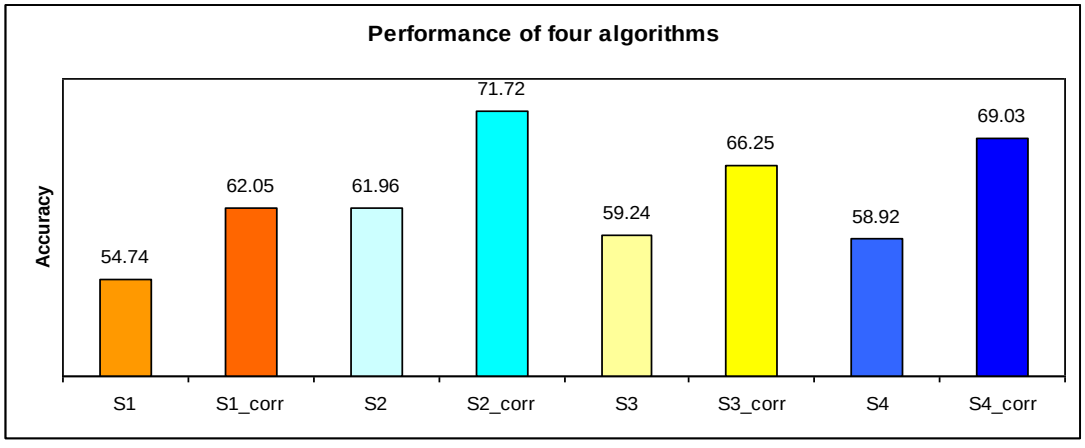


Figure 13: Comparison between average accuracy results of all weight-based algorithms with the ones incorporating the *Correction* algorithm using AT8 collection

Bolded values in tables above or below present maximum values whereas italic ones present minimum accuracy values.

Figure 13 shows that ***Adjusted Al-Shalabi*** algorithm with/out ***Correction*** provides the highest accuracy values among the other weight-based algorithms.

Accuracy of all techniques is found by each algorithm through:

1- comparing each extracted root with a predefined list of 5,405 roots that contains lists of only triliteral and quadriliteral roots (4,655 triliteral roots and 750 quadriliteral roots) (this root list provides the roots without relating them to their possible derived or inflected words),

2- The algorithm counts the roots that match the ones in the predefined list, and

3- Finally calculates the percentage of correct roots in each text of the collection.

A second method for calculating accuracy is performed by a native Arabic speaker (NAS) (the author) who manually provided the root for each word. NAS compared the root extracted by each algorithm with this root and counted the extracted roots that matched hers and finally gave the percentage of correctly extracted roots for each algorithm.

4.4 Analysis of Results

A comparison between the experimental results of the ***Rule-Based*** and the weight-based approaches are presented here using the two methods of accuracy calculations described above. Also, ***Rule-Based*** and ***Adjusted Al-Shalabi*** algorithms were implemented using LACC corpus and a comparison between their accuracy and execution time is presented.

4.4.1 First Accuracy Analysis Method

As has been illustrated, in the tables above and in Table 22 (samples of results of algorithms are shown in appendix III), Figures 14 and 15, that among the four

weight-based algorithms (without ***Correction*** algorithm), ***Adjusted Al-Shalabi*** algorithm provided the highest accuracy results. It is followed (in descending order) by ***EWBM1*** algorithm then by ***EWBM2*** algorithm and finally ***Al-Shalabi*** algorithm. Thus, ***Al-Shalabi*** algorithm had the lowest accuracy values, for all categories, among all four algorithms. This algorithm's accuracy values are in agreement with those reported in [160], although implemented on a different text collection. Also, among the four weight-based algorithms with the ***Correction*** algorithm, ***Adjusted Al-Shalabi with Correction*** algorithm provided the highest accuracy results followed by (in descending order) ***EWBM2 with Correction*** algorithm then by ***EWBM1 with Correction*** algorithm and finally ***Al-Shalabi with Correction*** algorithm.

The effect of adding the ***Correction*** algorithm to the weight-based algorithms, discussed in section 4.3.5 above and shown in Table 21, was to increase the accuracy of these algorithms by about 7%-10%. As is clear from the results shown above, that although the ***EWBM1*** algorithm is higher in accuracy than the ***EWBM2*** algorithm, yet their algorithms with ***Correction*** give the opposite result (i.e. ***EWBM2 with Correction*** algorithm is more accurate than ***EWBM1 with Correction*** algorithm). This preliminary observation indicates that ***EWBM2*** is more sensitive to irregular words.

	S1 %	**S1_corr**	**S2 %**	**S2_co**	**S3 %**	**S3_co**	**S4 %**	**S4_co**	**RB %**	**Enh_**
Politics	55.36	62.84	62.16	72.12	59.39	67.08	58.9	69.45	58.89	73.3
Economi	***52.59***	***58.9***	***60.73***	***70.41***	58.52	65.69	57.44	68.61	58.16	71.39
Religious	53.82	61.06	61.63	70.78	57.84	***64.07***	59.09	68.25	**62.99**	**75.01**
Social	**56.44**	**64.13**	63.21	**72.81**	59.98	67.22	59	69.37	60.56	74.79
Music	55.02	63.97	60.86	71.48	59.49	67.1	60.26	69.18	58.7	73.78
Educatio	53.46	59.85	62.37	71.59	59.73	66.33	59.05	**70.11**	60.67	74.81
Sports	55.53	62.19	61.67	71.53	***57.58***	64.61	***57.23***	***67.46***	***56.9***	***70.37***
Arts	55.67	63.42	**63.22**	72.64	**61.38**	**67.89**	**60.42**	69.78	61.4	74.27

Table 22: Accuracy results for all ten algorithms (all categories) using AT8 collection

It is noticed in Table 22 that the performances of algorithms vary among categories. An example is the *economics* category which had the lowest accuracy for the first four algorithms whereas the *sports* category had the lowest accuracy for five other

algorithms. Also, the *arts, culture and literature* category had the highest accuracy for four algorithms whereas the *social* category had the highest accuracy for other three algorithms. An interesting observation from Table 22 is that in general, categories that had the lowest results for some algorithms did not have the highest accuracy results for others. An exception for this observation is the *religious issues* category. From Figure 15, the effect of adding ***Correction*** algorithm to the weight-based algorithms was to increase the accuracy of these algorithms by about 7%-10% (with relative improvement of about 12%-17%), whereas its effect when added to the ***Rule-Based*** algorithm was to increase its accuracy by about 14% (with relative improvement of about 23%).

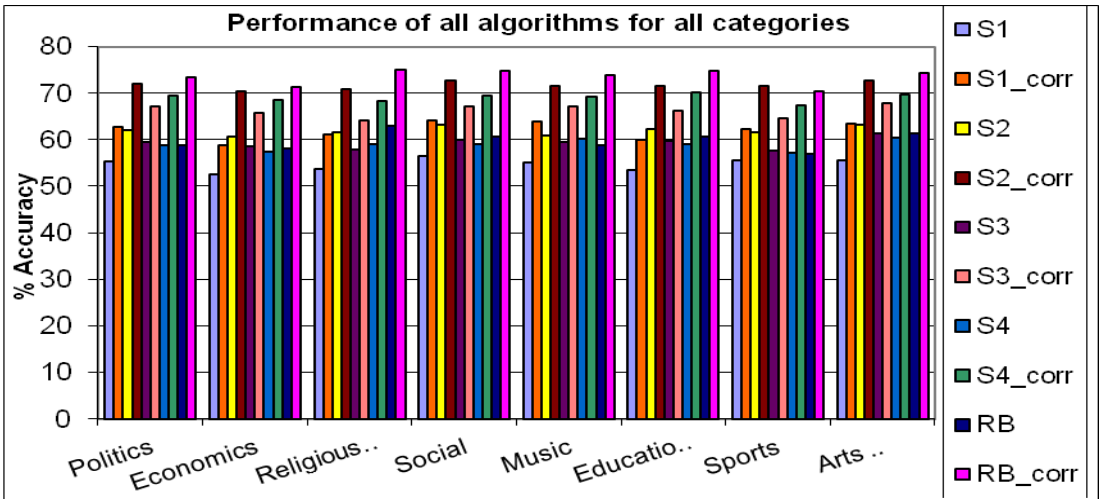


Figure 14: Comparison between accuracy results of all algorithms in all categories with the ones incorporating the *Correction* one using AT8 collection

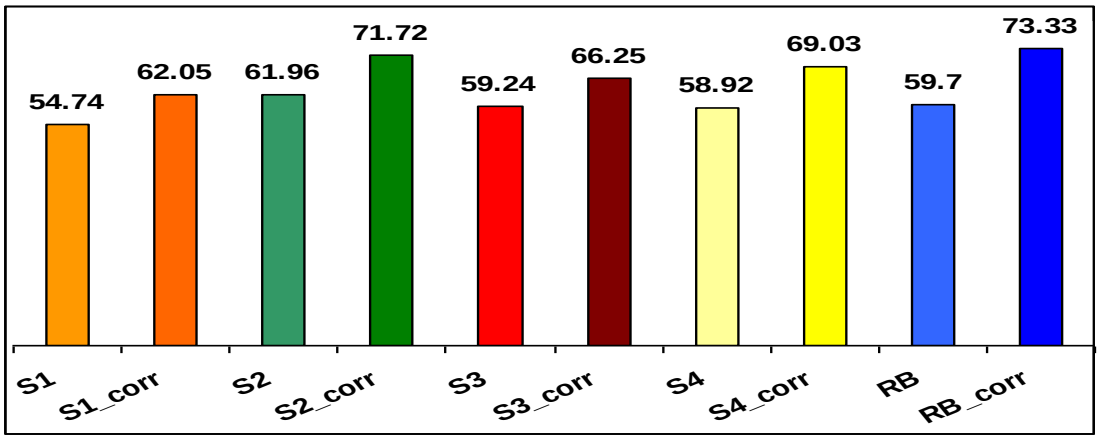


Figure 15: Comparison between average accuracy results of all algorithms with the ones incorporating the *Correction* one using AT8 collection

Table 23 and Figure 16 illustrate a comparison between ***Adjusted Al-Shalabi*** algorithm (the highest among weight-based algorithms) and ***Rule-Based*** algorithm along with their ***Correction*** algorithms in terms of their accuracy values. ***Rule-Based*** algorithm is less in accuracy than ***Adjusted Al-Shalabi*** algorithm in the range 1.81%-3.82%. However, ***Enhanced Rule-Based*** algorithm's accuracy is rather higher than ***Adjusted Al-Shalabi with Correction*** algorithm's accuracy by about 2%.

	S2(%)	S2_corr(%)	RB(%)	Enh_RB (%)
Politics	62.16	72.12	58.89	73.3
Economics	***60.73***	***70.41***	58.16	71.39
Religious issues	61.63	70.78	**62.99**	**75.01**
Social	63.21	**72.81**	60.56	74.79
Music	60.86	71.48	58.7	73.78
Educational	62.37	71.59	60.67	74.81
Sports	61.67	71.53	***56.9***	***70.37***
Arts ..	**63.22**	72.64	61.4	74.27

Table 23: Accuracy results for *Rule-Based* algorithm and *Adjusted Al-Shalabi* algorithm along with their *Enhanced* algorithms (all categories) using AT8 collection

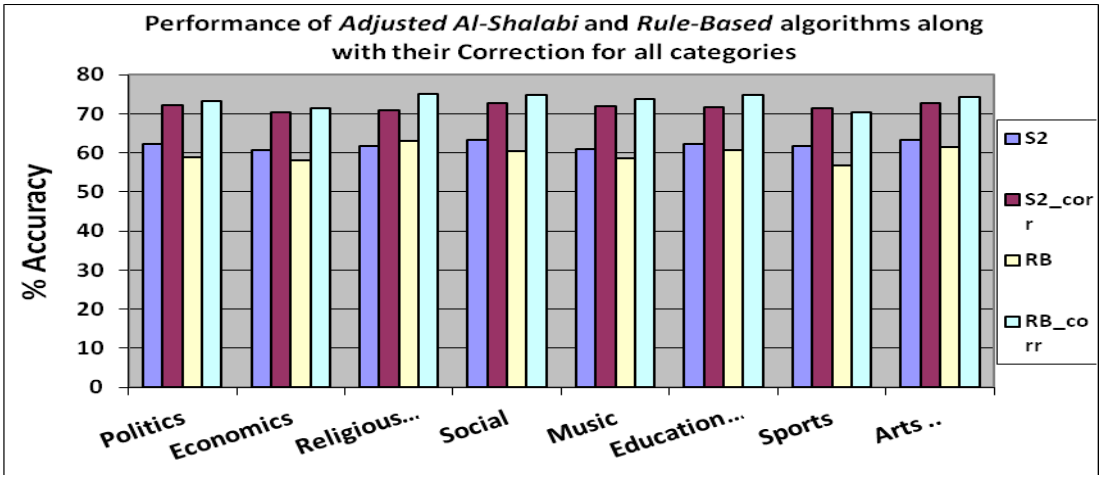


Figure 16: Comparison between accuracy results for *Rule-Based* and *Adjusted Al-Shalabi* algorithms along with their *Enhanced* algorithms (all categories) using AT8 collection

The difference in accuracy [129, pp. 208 – 210] between algorithms implemented is rather small. This makes it more difficult to conclude which is really better in performance. Thus, variance was calculated using eq. (1) for all algorithms and categories:

$$Var = \sum_{i=1}^{n} (x_i - \bar{x})^2 \quad (1)$$

Where *n: number of texts, x_i: accuracy of i^{th} text, $\bar{x}$: average accuracy of n texts.*

The results of obtaining all algorithms variance in all classes are shown in Figure 17.

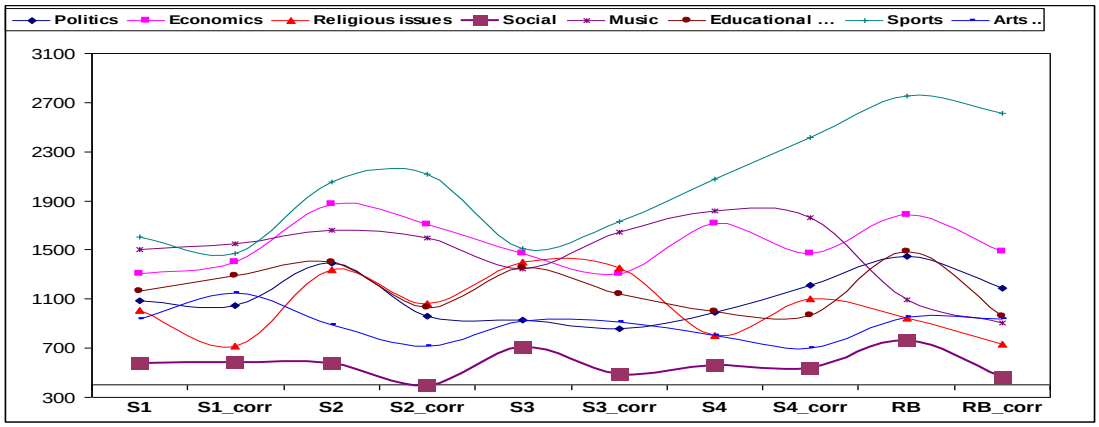


Figure 17: Variance values for all algorithms among all categories *(points were connected here by smooth curves for illustration purposes only)*

From Figure 17 it is clear that the worst category in performance is the *sports* category whereas the relatively best category in performance is the *social* one. Also, the *arts, culture and literature*, *educational*, *health and medicine*, *religious issues* and *politics* categories have lower variance values than the others. This is in agreement with earlier indication that the *social* category gave the highest accuracy values and that the *sports* category gave the lowest values. It is expected that the high variance in such categories is partially due to the higher presence of names and foreign Arabized words in them.

Since the two algorithms with highest accuracies are ***Adjusted Al-Shalabi*** and ***Rule-Based*** algorithms as was explained above, the concentration here will be on the variance values for ***Adjusted Al-Shalabi*** and ***Rule-Based*** algorithms along with their ***Enhanced*** algorithms as illustrated in Figure 18. The variance values are very near and cannot clarify which of the two algorithms (or with their ***Enhanced*** algorithms) is better. Thus, [129, pp. 208 – 210] we use here the t-test (see appendix II for SPSS analysis results of normal distributions for these algorithms) by hypothesizing that

Adjusted Al-Shalabi algorithm is better than the ***Rule-Based*** algorithm (as the null hypothesis). Then, we calculate the t-value using eq. (2) shown below:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{2s^2}{n}}} \quad (2)$$

Where $\bar{x_1}$*: accuracy of* ***Adjusted Al-Shalabi*** *algorithm,* $\bar{x_2}$*: accuracy of* ***Rule-Based*** *algorithm,* s^2*: pooled variance of both algorithms* and

$$s^2 = \frac{Var_1 + Var_2}{n_1 + n_2 - 2} \quad (3)$$

Where Var_1*: variance of* ***Adjusted Al-Shalabi*** *algorithm,* Var_2*: variance of* ***Rule-Based*** *algorithm,* $n_1 = n_2$*: number of texts for both algorithms*

After substituting the accuracy values of algorithms and their pooled variance, t-value is found to be 5.56. At a probability level of $\alpha = 0.01$, the critical value of t is 2.576 (using a one-tailed test with ∞ degrees of freedom [129, pp. 609]. Since here $t = 5.56 > 2.576$ then the hypothesis is accepted. The t-test is also performed when including ***Correction*** algorithm to the two mentioned ones above. The hypothesis here is that ***Enhanced Rule-Based*** algorithm is better than ***Adjusted Al-Shalabi with Correction*** algorithm. Their t value is 4.52 and at a probability level of $\alpha = 0.01$, the critical value is $t = 2.576$ (using a one-tailed test with ∞ degrees of freedom). Since here $t = 4.52 > 2.576$ then the hypothesis is accepted.

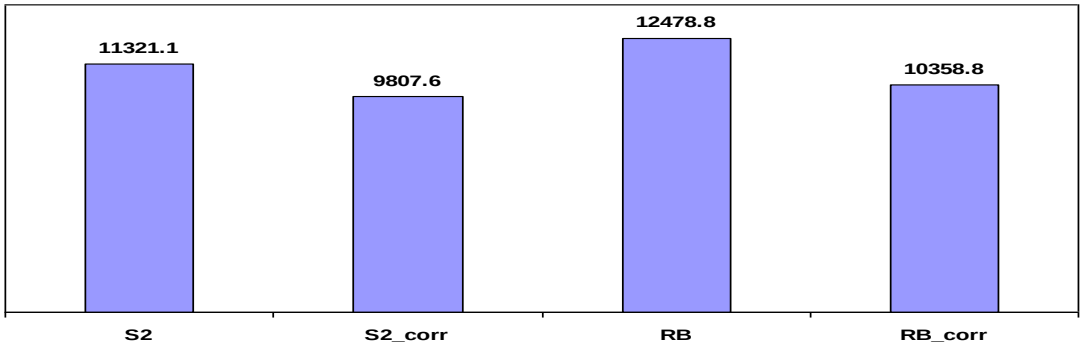


Figure 18: Comparison between total variance results for *Rule-Based* and *Adjusted Al-Shalabi* algorithms

Thus, one concludes that the approach with the highest accuracy among all algorithms would be ***Enhanced Rule-based*** algorithm.

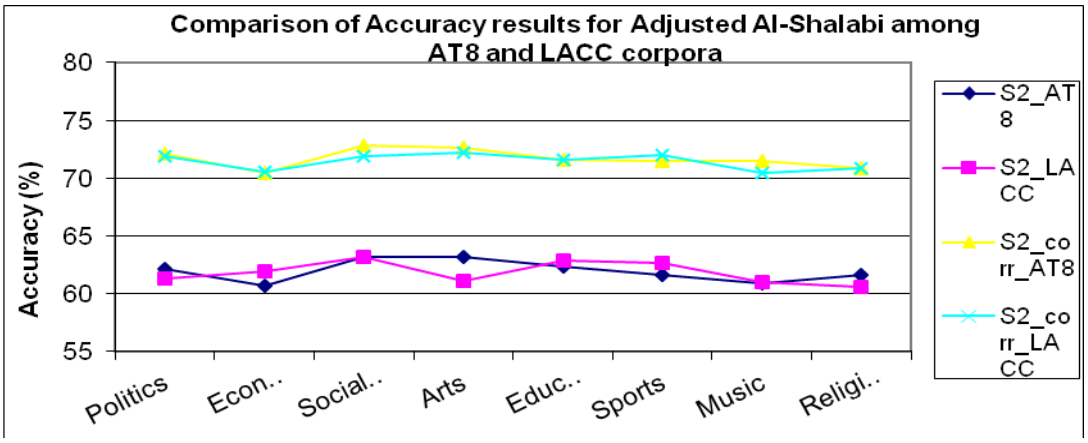
	S2	S2_corr	Variance change for S2		RB	Enh_RB	Variance change for RB	
			ΔS2	(ΔS2/S2)%			ΔRB	(ΔRB/RB)%
Politics	1394.4	977.3	417.1	**29.9**	1446.2	1211.1	235.1	**16.3**
Economics	1876.6	1747.9	128.7	**6.9**	1789.5	1521.7	267.8	**15**
Religious	1337.8	1084.9	252.9	**18.9**	942.55	746.2	196.4	**20.8**
Social	572.1	400.3	171.8	**30**	765.2	478	287.2	**37.5**
Music	1660.8	1638.6	22.2	**0.1**	1094.9	925.8	169.1	**15.4**
Educational	1403.3	1053.7	349.6	**24.9**	1487.1	980.3	506.8	**34.1**
Sports	2056.2	2114.8	-58.6	**-2.9**	2753.2	2609.4	143.8	**5.2**
Arts ..	891.8	716.9	174.9	**19.6**	951.2	937.4	13.4	**1.5**
All	***11321.1***	***9807.6***	***1513.5***	***13.4***	***12478.***	***10358.8***	***2120***	***17***

Table 24: Variance values among categories for *Rule-Based* and *Adjusted Al-Shalabi* algorithms along with their *Enhanced* algorithms using AT8 collection

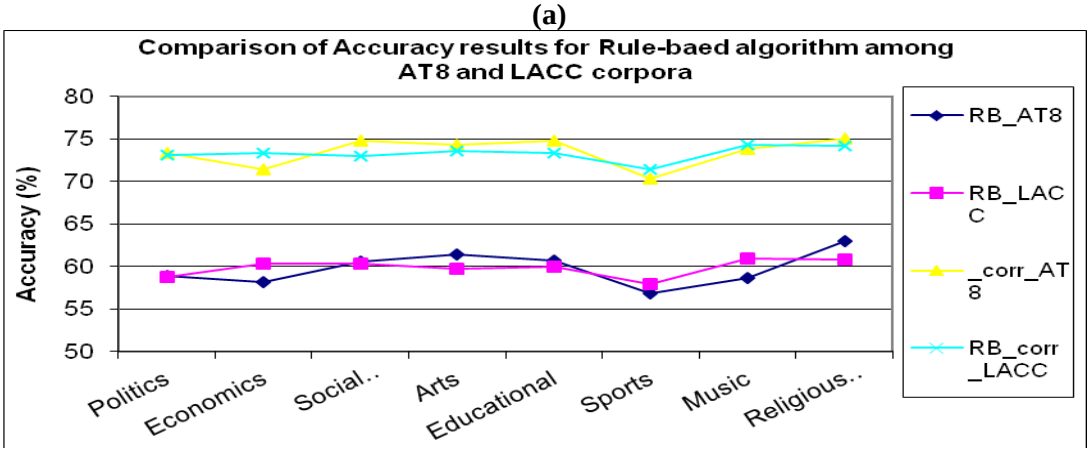
From Table 24, the effect of using the ***Correction*** algorithm varied among algorithms and categories in minimizing variance values. It was more effective in doing so for *social* and *educational* categories. However, it varied in doing so for other categories such as *politics*, *music* and *sports*. In general, such ***Correction*** algorithm lowered variance and improved performance of all algorithms and categories. The above results are for the implementation of algorithms on only the AT8 collection. However, since the LACC corpus, which is described in Chapter 3, is used in Chapter 5 for the implementation of TC methods, it was decided to implement both ***Rule-Based*** and ***Adjusted Al-Shalabi*** algorithms and their ***Enhanced*** algorithms on LACC corpus. These two algorithms were chosen only since they provided the best results as was described above. The results of such implementation are shown in Table 25 and a comparison between results of these algorithms on AT8 and LACC corpora is shown in Figure 19.

	S2 (%)	S2-Corr (%)	RB (%)	Enh_RB (%)
Politics	**61.35**	**71.86**	**58.79**	**73.07**
Economics	**61.97**	**70.6**	**60.35**	**73.37**
Social issues	**63.22**	**71.88**	**60.28**	**72.92**
Arts	**61.16**	**72.2**	**59.68**	**73.52**
Educational	**62.86**	**71.6**	**59.98**	**73.28**
Sports	**62.69**	**72.04**	**57.91**	**71.41**
Music	**60.99**	**70.45**	**60.96**	**74.34**
Religious issues	**60.58**	**70.87**	**60.84**	**74.12**
Average	***61.85***	***71.44***	***59.85***	***73.25***

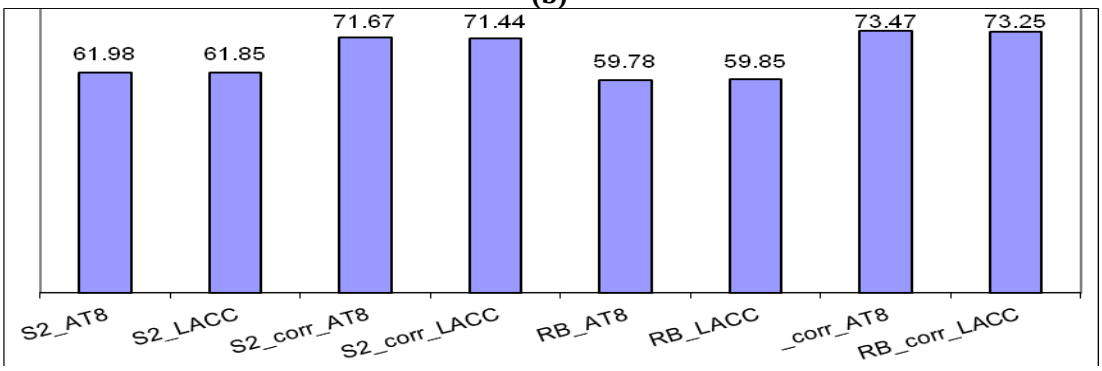
Table 25: Accuracy results for *Rule-Based* algorithm and *Adjusted Al-Shalabi* algorithm along with their *Enhanced* algorithms (all categories) using LACC corpus



(a)



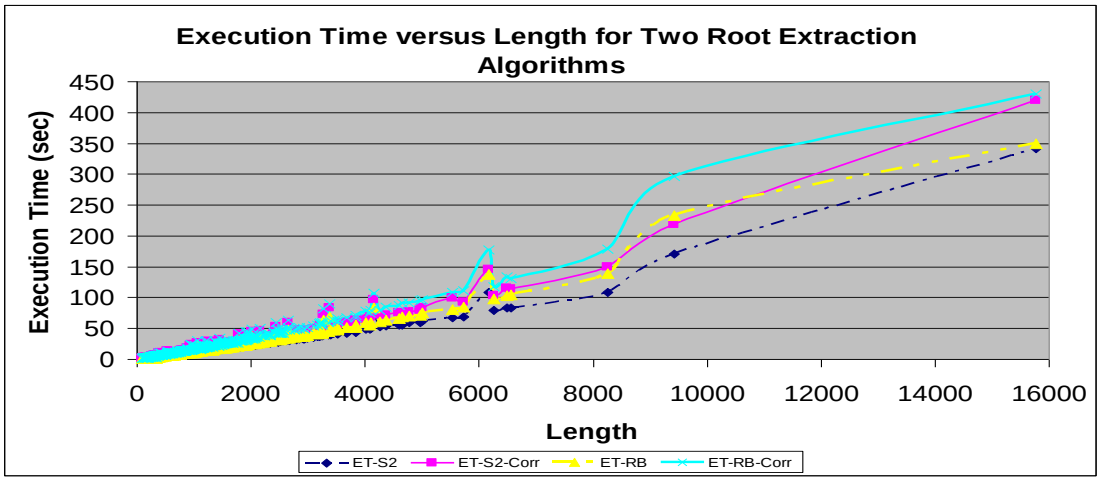
(b)



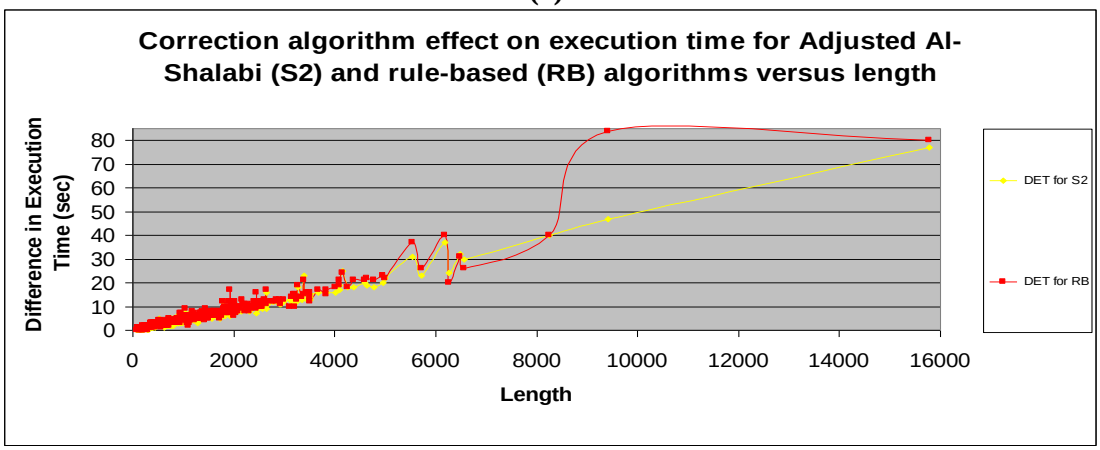
(c)

Figure 19: Comparison among AT8 and LACC corpora: (a) for *Adjusted Al-Shalabi* in all categories, (b) for *Rule-based* in all categories, (c) between two algorithms along with their *Enhanced* algorithms on average

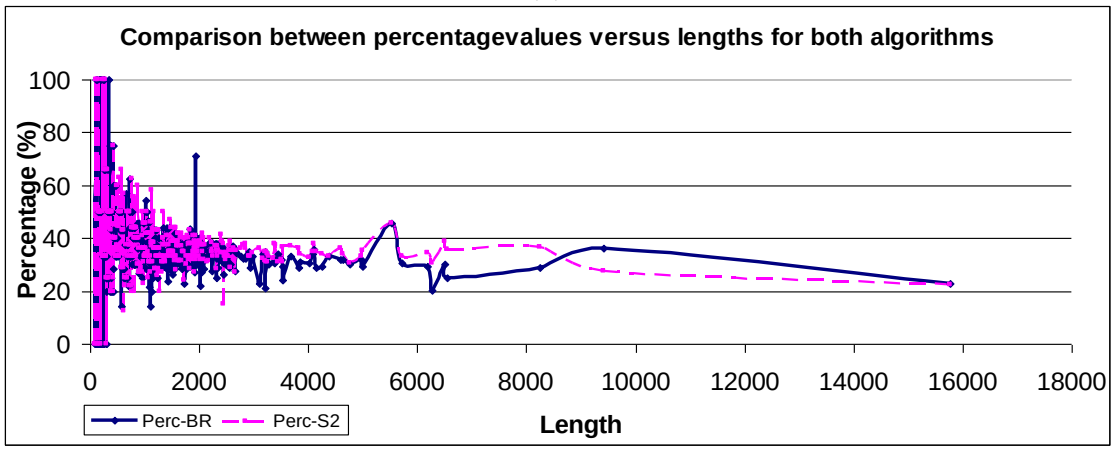
Figure 19 emphasizes our previous conclusion that ***Enhanced Rule-Based*** algorithm provides the best results among investigated algorithms in terms of accuracy. Another criterion that the two algorithms performances are compared by is their execution time. This was performed using LACC corpus since this corpus has a large range of texts' length (89 – 15,773). Figure 20 presents results of such comparison.



(a)



(b)



(c)

Figure 20: Investigation of performance of both *Adjusted Al-Shalabi* and *Rule-Based* algorithms and their *Enhanced* algorithms as Length of texts increases with (a) change in their Execution time, (b) the difference in Execution time for each algorithm, (c) the Percentage of (difference in execution time by execution time for each algorithm)

Figure 20a shows that for lengths less than 5,000, the results of execution time for both algorithms and their ***Enhanced*** algorithms are similar whereas as length increases above 5,000 the difference in execution time becomes more apparent especially

above 8,000. This result indicates that although the ***Rule-based*** algorithm is supposed to take longer time since it has many rules, yet the time it takes to execute is similar to the one taken by the ***Adjusted Al-Shalabi*** algorithm. Such a result is an indication of its efficiency. Figure 20b illustrates the effect of ***Correction*** algorithm on execution time. For texts with length less than 7,000 the execution time is highly similar for both algorithms. However, for length values higher than that, the effect of ***Correction*** algorithm on execution time for ***Rule-Based*** algorithm is apparently higher than that for ***Adjusted Al-Shalabi*** one. Finally, Figure 20c presents the efficiency of both algorithms in terms of the effect of ***Correction*** algorithm by finding the percentage of the difference of execution time for each algorithm (i.e. with/out ***Correction*** one) to the execution time for that algorithm without ***Correction*** as length increases). It indicates that the effect of ***Correction*** algorithm is similar in both algorithms for text length less than about 5,500. However, for length range 5,500 to 8,300 the effect of ***Correction*** algorithm is more in ***Rule-Based*** algorithm than in ***Adjusted Al-Shalabi*** algorithm since percentage is less. Nevertheless, surprisingly so, this effect is the opposite for lengths more than 8,300. In general, Figure 20c shows that as length increases the percentage decreases until it reaches a rather constant value.

4.4.2 Native Arabic Speaker Accuracy Analysis

NAS manually, as a preliminary analysis, provided the roots for words in only about 40 texts (5 in each category) chosen randomly from the AT8 collection only. First, all compound words or single letters (named here unidentified words), names of places, countries, cities or months and foreign Arabized words, un-detected function words were excluded and their percentages in the texts in each category was counted then the accuracy for each algorithm was calculated.

The preliminary results of the NAS's analysis percentages of excluded words are shown in Figure 21. Here, it is noticed that the percentage of names and foreign Arabized words is highest in *music*, *economics*, *politics* and *sports* categories whereas it is lowest in *social* and *religious issues* categories. This might explain partially the high and low accuracy and variance values in these categories as was shown in the above section. On average the presence of these excluded words are as shown in Figure 22. From Figure 22, it is expected that the NAS's accuracy results would be less than the ones of the first method by about 14% (by excluding names and foreign Arabized words 10.62%, unidentified words 0.57% and function words 3.42% percentages). NAS noted that the rather unexpected presence of undetected function words was due to a main factor which is: in some texts (coming for example from Al-Ahram news paper) most of the undetected function words were misspelled such as the function word *ElY* (meaning on) was misspelled as *Ely* (an Arabic name of a male person).

NAS counted the number of words in each text that has the letter *b* at its first or second position in the word and found that it appeared in analyzed texts with a percentage of about 6.22%. Such appearance may affect the accuracy of algorithms if the special effect of *b,* as an extra letter when at beginning of a word, is not handled. It is worth noting that the effect of *b* was not handled in any of the weight-based algorithms as was illustrated in section 4.3 where it was given the weight zero (i.e. it will always be considered an original letter in any word that starts with it in these algorithms).

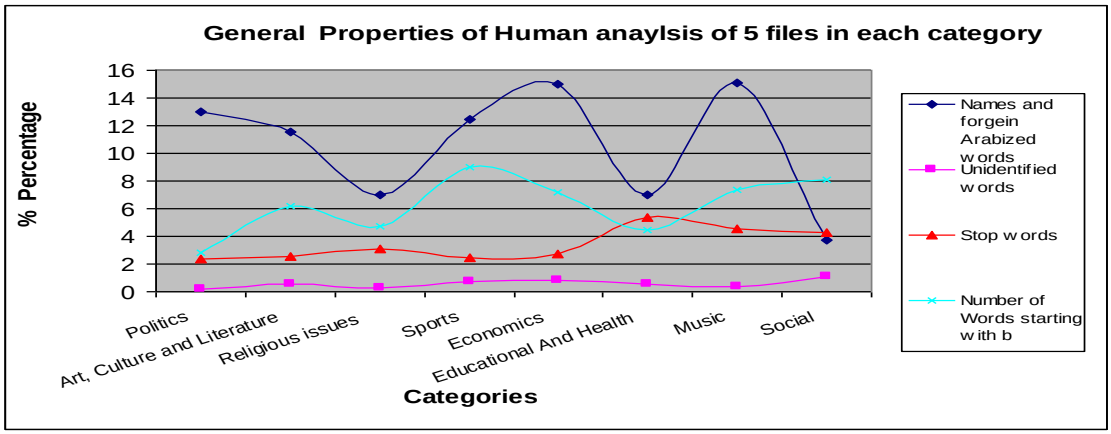


Figure 21: Percentages of unidentified words, function words, foreign Arabized words in texts in all categories *(points were connected here by smooth curves for illustration purposes only)*

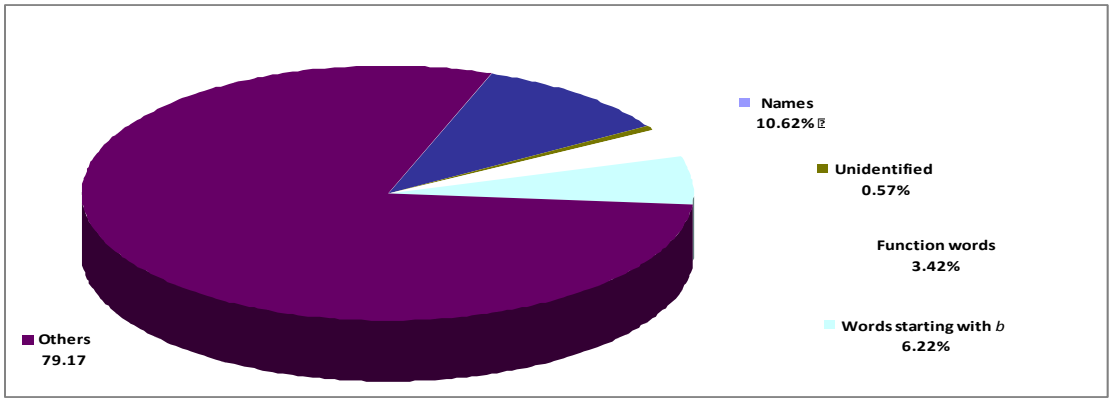


Figure 22: Average percentage for function words, unidentified words and foreign Arabized words

NAS then compared the roots she provided with those extracted by each algorithm and calculated the accuracy of correctly identified roots for each algorithm as shown in Table 26 and Figure 23. The accuracy values for the algorithms clearly emphasize that the best algorithm for root extraction in terms of accuracy is ***Enhanced Rule-Based*** algorithm followed by ***Adjusted Al-Shalabi with Correction*** algorithm.

	Human expert analysis of Algorithms Accuracy %Acc									
	S1	**S1_corr**	**S2**	**S2_corr**	**S3**	**S3_corr**	**S4**	**S4_corr**	**RB**	**Enh_RB**
Politics	41.23	43.21	52.07	54.84	47.39	50.44	48.68	51.8	51.45	58.47
Economics	40.27	**42.51**	49.64	52.07	47.23	**48.36**	**45.59**	**48.36**	50.17	54.01
Religious	43.85	**47.53**	55.52	**59.1**	52.53	**55.86**	**51.56**	**55**	**56.81**	**61.67**
Sports	42.05	43.62	53.1	56.04	48.51	50.33	49.03	51.66	49.51	53.85
Social	44.21	45.88	53.36	55.34	48.82	51.01	48.98	50.93	54.83	59.11
Educational	41.49	42.75	53.74	56.11	49.44	52.21	50.51	53.52	55.65	59.73
Music	**39.25**	42.82	**46.28**	**50.15**	**44.04**	48.6	45.93	49.08	**45.52**	**52.45**
Art, ..	**45.05**	46.65	**55.6**	57.4	**52.81**	54.24	51.43	53.19	54.14	57.72
Aver.	***42.18***	***44.37***	***52.41***	***55.13***	***48.85***	***51.38***	***48.96***	***51.69***	***52.26***	***57.13***
Imp.	***+2.19***		***+2.72***		***+2.53***		***+2.73***		***+4.87***	
Rel. Imp.	***+5.19***		***+5.19***		***+5.18***		***+5.58***		***+9.32***	

Table 26: Native Arabic speaker analysis of algorithms' accuracy using AT8 collection

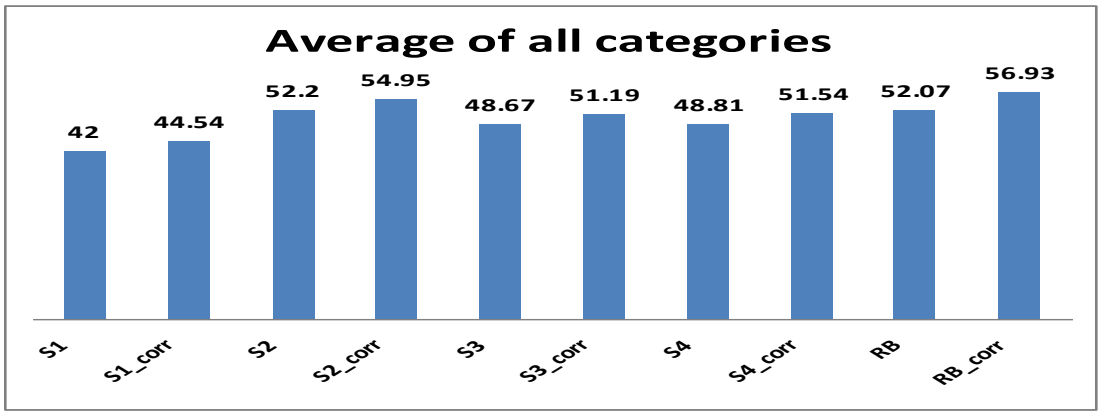


Figure 23: Native Arabic speaker analysis of algorithm's accuracy

It is noticed in Table 26 that the performances of algorithms vary among categories. However, the *religious issues* category had the highest accuracy for seven algorithms whereas the *music* category had the lowest accuracy for six algorithms. Thus, *religious issues* category is affected much more by the performance of such algorithms. Also, the *music* category is affected much less by the performance of such algorithms.

By simple comparison one can observe that the difference between the first and second methods' accuracies is on average for: ***Al-Shalabi*** algorithm in the range 12.74%–17.51%, ***Adjusted Al-Shalabi*** algorithm in the range 9.78%–16.77%, ***EWBM1*** algorithm in the range 10.57%–15.06%, ***EWBM2*** algorithm in the range 9.42%–18.49%, and for ***Rule-Based*** algorithm in the range 7.72%–16.54%. Such differences are expected and are due partially to the fact that in human analysis names, function words, and unidentified words were excluded (around 14%) before counting the accuracy. However, for the ***Rule-based*** algorithm, the difference between the first and second analyses is the least one (about 7%) compared to the others. This smaller difference is partially due to the fact (as the human analyzer observed) that the ***Rule-based*** approach did not extract a trililetral root for many foreign Arabized words as the other approaches did.

From NAS's analysis, the ***Correction*** algorithm improvement on algorithms' accuracies varied on average where for: ***Al-Shalabi*** algorithm it was 2.54%, ***Adjusted Al-Shalabi*** algorithm about 3%, ***EWBM1*** algorithm 2.52%, ***EWBM2*** algorithm 2.73%, and for ***Rule-Based*** algorithm about 5%.

In order to investigate the performance of these algorithms apart from the effect of different words (i.e. names, foreign Arabized words, function words, and compound words), NAS recalculated the accuracy of each algorithm per category by dividing the number of correct roots by (the total number of words in text minus the number of such different words). The results are shown in Figure 24.

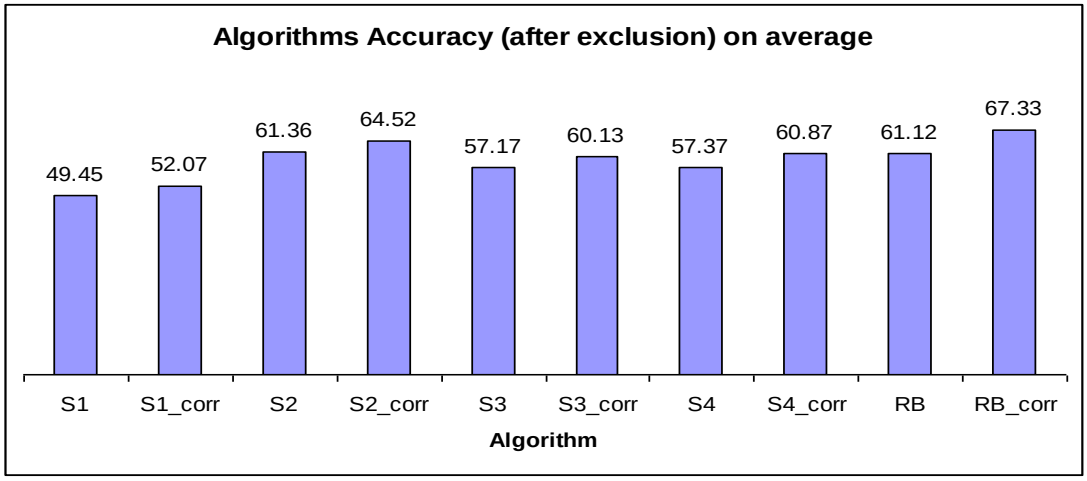


Figure 24: Native Arabic speaker analysis of algorithm's accuracy after excluding no. of names, transliterations, function words and compounds from total no. of words in texts

From Figure 24, one can observe that there is a slight difference in the relative improvements among algorithms based on weight-based technique (about 5.1% – 6.9%), whereas for ***Rule-Based*** algorithm the relative improvement is about 10.1%. The above mentioned relative improvement percentage values are not near those found from Figure 15 in section 4.4.1. This difference can be due to some limitations such as:

1- In specific cases ***Correction*** algorithm does not check the extracted root since it is not reached (this is due to the fact that the extracted root is found in the predefined root list (so is considered correct even though it is actually the wrong

root), e.g. if an algorithm extracts a root *sgl* for the word *IstglAl*, then it checks if this root is in root list and finds it is so even though the correct root for this word is *gll* which is *two-letter geminated* root)),

2- In other few cases the extracted root is not found in ***Correction*** algorithm to be corrected, although required so, since its case is not handled,

3- In other cases the extracted root is not found in the root list, although correct, since the root list provided here does not include all roots (since Arabic roots are estimated to be 10,000 [49]),

4- In other cases, although relatively few, a surface word might have more than one option for correction and ***Correction*** algorithm chooses (according to its structure) only one of them (that might be wrong), (e.g. is the word 'نمت' *nmt* if pronounced *nemto* "I slept" is thus corrected to the root *nwm* but if pronounced *nam~t* "she gossiped" then is corrected to *nmm*). This limitation can be handled by redesigning the algorithm to take such options as alternatives and to include all possible roots in the root list,

5- In most wrongly handled cases, the original algorithm is not successful in removing prefixes and suffixes correctly always so the extracted root is wrong although the ***Correction*** algorithm can handle its case.

Other factors that the native Arabic speaker has studied in the algorithms' performance were their efficiency in extracting *weak, two-letter geminated* roots and four-letter roots. The results of such analysis are presented in Figures 25, 26, and 27 respectively where Figure 25 presents the percentage of wrongly detected *weak* roots to the number of words in text, Figure 26 presents the percentage of wrongly detected *two-letter geminated* roots to the number of words in text, and Figure 27 presents percentage of wrongly detected four-letter roots to number of words in text.

As can be seen from Figure 25, the effectiveness of algorithms in correctly extracting *weak* roots can be categorized according to their percentage values. This figure indicates the higher efficiency of ***Rule-based*** approach (with relative improvement of about 33% when ***Correction*** algorithm is included) compared with the weight-based-based ones except for ***EWBM2*** algorithm (with relative improvement of about 11% for ***Al-Shalabi*** algorithm, 19% for ***Adjusted Al-Shalabi*** algorithm, 14% for

EWBM1 algorithm, and 18% for ***EWBM2*** algorithm when ***Correction*** algorithm is included). ***EWBM2*** algorithm presented the highest algorithm in detecting *weak* roots among all five original algorithms. This is due to the fact that this algorithm, as was explained in section 4.3.4, replaced the letter *y* by *Y* in words in the text before extracting the roots. This pre-process helped some *weak* words to produce the desired root but lowered in general the accuracy. However, ***Rule-based*** algorithm is the most sensitive among the rest since the inclusion of ***Correction*** algorithm to it in terms of correcting *weak* roots had the highest relative improvement of 33%.

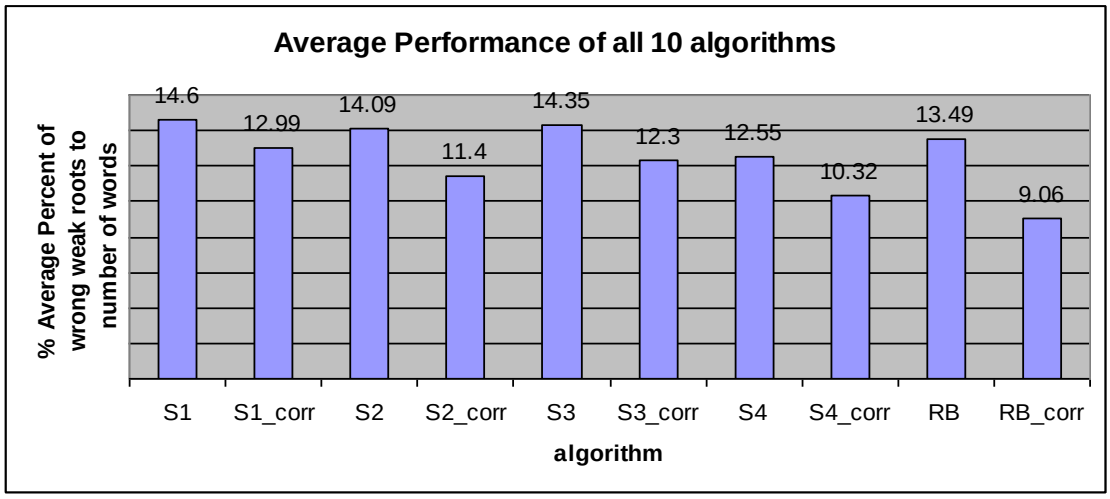


Figure 25: Percentage of wrongly extracted *weak* words by all algorithms

As for *two-letter geminated* roots, Figure 26 illustrated that ***EWBM2*** algorithm performed better than the rest of the four weight-based algorithms, which is expected since it is the only one among them that attempted to handle this issue. However, ***Enhanced Rule-based*** algorithm gave the best results in extracting *two-letter geminated* roots among all algorithms with a relative improvement of about 17%. The performance of the weight-based algorithms when ***Correction*** algorithm was included also increased with relative improvements of 9.4%-13.7%. This is another indication of the effectiveness of the proposed ***Correction*** algorithm.

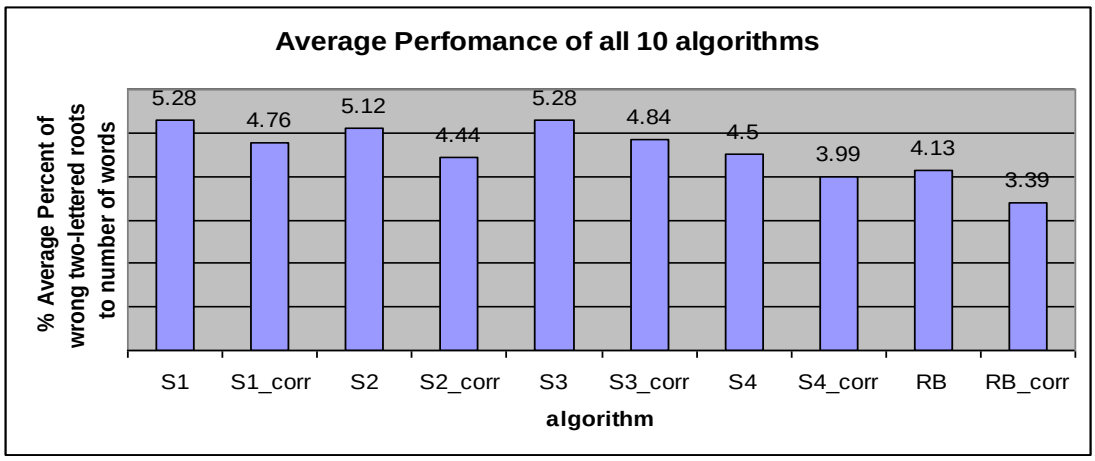


Figure 26: Percentage of wrongly extracted *two-letter geminated* words by all algorithms

From preliminary results, only 69.23% of analyzed texts by the native Arabic speaker had more-than-three-lettered roots with very low numbers (about only 1%). Algorithms ***Al-Shalabi*** and ***Adjusted Al-Shalabi*** do not handle such roots. Only ***EWBM1*** and ***EWBM2*** algorithms attempt to extract four-letter roots and Figure 14 shows that these algorithms succeeded to partially extract four-letter roots with about 12% improvement whereas ***Rule-based*** algorithm was successful in extracting four-letter roots with about 37% improvement. Although both ***EWBM1*** and ***EWBM2*** algorithms provided the same improvement percentage, nevertheless the Arabic speaker noticed that ***EWBM1*** algorithm in few cases wrongly extracted three-letter roots as four-letter ones and this increased the error. Also, although ***EWBM2*** algorithm was not successful in extracting all four-letter roots (if available in texts) but it did not perform as ***EWBM1*** algorithm in wrongly extracting some three-letter roots as four-letter ones. Still, ***Rule-based*** approach is considered the best algorithm among all five algorithms to provide correct four-letter roots.

From the description of ***Correction*** algorithm in section 4.2.2 above, this algorithm corrects special cases of triliteral roots and thus is expected not to change the performance of any of the algorithms as is shown in Figure 27.

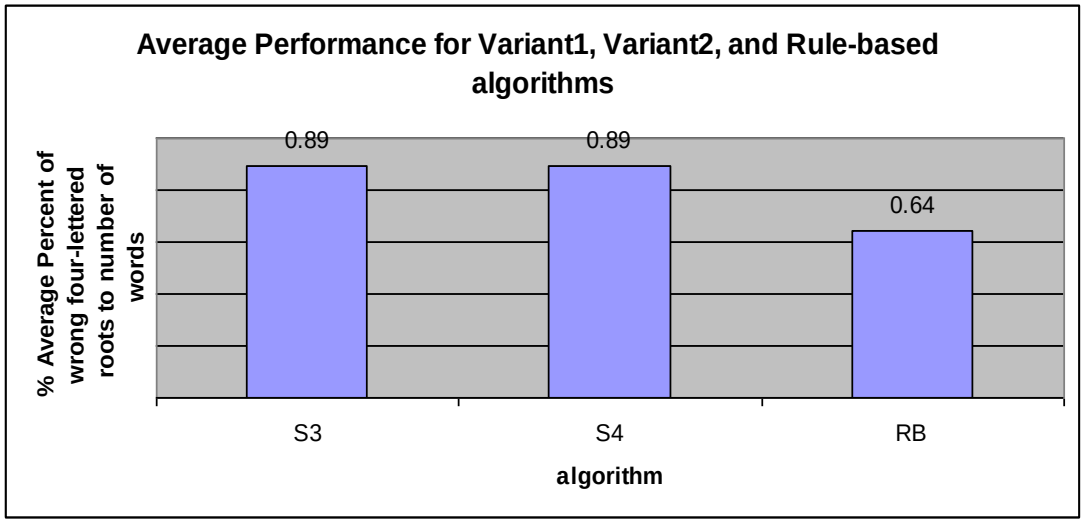


Figure 27: Percentage of wrongly extracted four-letter words

In brief the effectiveness of presented algorithms showed:

1) In correctly extracting *weak* roots, ***Enhanced Rule-Based*** algorithm had the highest percentage among all algorithms, followed by ***EWBM2 with Correction*** algorithm. The ***Adjusted Al-Shalabi with Correction*** algorithm showed the third highest percentage. This clearly indicates the higher efficiency of the ***Rule-Based*** approach compared with the others.
2) In correctly extracting *two-letter geminated* words, only ***EWBM2*** and ***Rule-Based*** algorithms were investigated along with their ***Enhanced*** algorithms since they are the ones that handle such cases. Here ***EWBM2*** algorithm performed less efficiently than ***Rule-Based*** one. By including the ***Correction*** algorithm, ***Enhanced Rule-Based*** algorithm gave the best results in extracting *two-letter geminated* roots.
3) In correctly extracting more-than-three-lettered roots, only ***EWBM1*** and ***EWBM2*** algorithms extract four-letter roots along with the ***Rule-Based*** one. The proposed two algorithms succeeded in partially extracting four-letter roots with about 12% relative improvement whereas the ***Rule-Based*** algorithm was successful in extracting such roots with about 37% relative improvement.

From the results shown above, it is evident that the accuracy obtained here for the non-***Rule-Based*** approach of ***Al-Shalabi*** algorithm or its ***Adjusted*** one is not near the accuracy claimed in that work. This is partially due to that in [17] the text collection was a small corpus whether in terms of its texts or words numbers and concentrated into a specific category. There is no information in Al-Shalabi's, et al [17] work regarding availability of *weak*, *hamzated*, or *geminated* words in abstracts collection.

Also, the accuracy of ***Rule-based*** root extraction approach of Al-Ameed work implemented here is not near to the accuracy claimed at Al-Ameed work. This is also partially due to the fact that at Al-Ameed's work the algorithm was tested using a specially customized test of derived words from triliteral roots and quadriliteral roots. Since neither the corpus used in Al-Shalabi, et al work nor the test set used in Al-Ameed work could be acquired, the AT8 collection was used to test all algorithms.

In all, the addition of ***Correction*** algorithm to the ***Rule-Based*** algorithm gave the highest improvement in accuracy among all original five algorithms, yet the above results are still preliminary. Also, the examples presented at Al-Ameed's work, as far as was observed, did not include any *weak*, *hamzated* or *geminated* words. However, AT8 and LACC collections used here contain proper nouns, foreign Arabized words as well as *weak, hamzated,* or *geminated* words. The analysis presented here urged the author to construct a list that handles foreign Arabized words and names of countries, places as will be presented next.

4.5 Foreign Arabized Words List

The list presented here is composed of Foreign Arabized Words as well as Names of Places, Countries, Continents and Cities was constructed here by gathering manually foreign words available in Arabic texts of this corpus (named here ***FAW*_List**). Foreign words that are used in Arabic texts are of two categories. The first includes words that obey Arabic patterns but are not Arabic while the second does not include such cases. The gathered foreign words list here is mostly for the second category. In future it is intended to increase this list to include the first category as well as other

words of Persian or Turkish origins. Examples of the constructed foreign Arabized words, names of places, countries, cities, and month's list are presented in Table 27.

Foreign Arabized words	"دولار"،"بالديمقراطيـــة"، "وديمقراطيـــة"، "البرلمانيـــة"، "أســيك"، "باســتاد"، "بـــروكس"، "شـــامب"، "الفلافيـــو"، "كوامبيـــه"، "لموســـي"، "أوتـــرا"، "بـــإينو"، "بوجلبـــان"، "جيلبرتـــو"، "مـــارتن"، "بـــاوا"، "وباتريـــك"، "نـــداو"، "موســـي"، "سولي"، "كيا"، "جوزيـه"، "فلافيو"، "اليونسكو"، "الدكتورة"، "والتكنولوجيا"، "التكنولوجيـا"، "رمسـيس"، "العربجليزيـة"، "الإنترنـت"، "والفـيس"، "بـوك"، "والإنجليزيـــة"، "فرعـــون"، "مفـيش"، "فايـدة"، "ديمقراطيـــة"، "عســـكران"، "المريخ"، "يحيي"، "فينيكس"، "ناسا"، "للمريخ"، "ميكروبات"، "أندرو"، "ويل"

Table 27: Examples o Foreign Arabized words list

This list consists of 7,227 words[30]. Next, a description of the proposed root extraction system is presented.

4.6 Final Proposed Root Extraction System

The two root extraction algorithms with highest accuracy values (based on the results of all root extraction algorithms presented in section 4.4, pages 72-77), namely the ***Enhanced Rule-based*** (***Enh_RB***) and the ***Improved Adjusted Al-Shalabi*** (***S2-Corr***) methods, although near in their accuracy values, are selected for use. The other methods performed lower than these two, but after analyzing their results some of their proposed and implemented parts proved to be effective, namely the ***EWBM2*** method. Thus, the proposed root extraction system would be composed of these two root extraction techniques. Figure 28[31] is a flowchart that summarizes the proposed root extraction system. ***EWBM2*** algorithm included handling *two-letter geminated* roots and results of analyzing its effect (shown in page 87) showed that it improved the performance of such method by about 5%. Thus, the ***Enh_RB*** method is improved by including at its beginning handling the extraction of *two-letter geminated* roots and named here as ***Enhanced Rule-Based_2*** method (abbreviated

[30] This list was gathered from: 1- http://www.bbc.co.uk/arabic/learningenglish/2010/08/801016_cojo_arabic_guide5.shtml, 2- http://mogameh.ahlamontada.net/t9899-topic, 3- remaining words gathered by author from corpus.
[31] Flowchart symbols are according to the link: http://www.eng.iastate.edu/efmd/161algor.htm

Enh_RB_2 method). The ***Improved Adjusted Al-Shalabi*** one (***S2-Corr***) does not handle quadriliteral roots or *two-letter geminated* roots, so it is improved by including to it: a) handling specific cases of quadriliteral roots proposed and implemented in ***EWBM2*** algorithm presented in section 4.3.4 (only steps 8-15 of this algorithm, page 68), and b) handling the extraction of *two-letter geminated* roots. This improved algorithm is named ***Improved Adjusted Al-Shalabi_2*** algorithm as shown below (abbreviated here ***IAA*** method). Both original methods lack handling foreign Arabized words and as such the handling of such cases are included in the proposed root extraction system. The pseudo-code of this system is presented (this system enables the user to choose which of these two methods to use or both). Also, the proposed improved algorithms are briefly presented below.

The two improved root extraction algorithms output lists that are included in the proposed system in order to be incorporated with the identified foreign Arabized words in documents and outputted for the user as respective output documents of normalized words, stems or roots. This proposed root extraction system algorithm is an addition to this work where it combines the best performing implemented algorithms in an overall one.

4.7 Conclusions and Future Work

In this part we contribute with ***Correction*** algorithm [13] in order to:

1) replace long vowels appearing in words that require to be changed in order to have the correct root for a word according to specific rules,
2) delete an extra letter (at the beginning or end) of *two-letter geminated* roots,
3) handle specific cases of *eliminated-long-vowel* words,
4) handle specific cases of *hamzated* rots.

The ***Correction*** algorithm was included into two different approaches for root extraction, a ***Rule-based*** and a weight-based one. Furthermore two contributions of

Proposed Root Extraction System Algorithm

Inputs: Set of n documents in Arabic corpus d_i i = 1,n, Two-letter_geminated_words_List, ***FAW***_List, Chosen root extraction method(s)
Outputs: List of triliteral and some quadriliteral roots, stems and normalized words for each new document new_d_i_1 separately in output set DD

START
1- For each document d_i do {
2- LastWord = Count_No_Words(d_i)
3- For j = 1 to LastWord in d_i do {
4- LastLetter = Count_No_Letters(w_j,c)
// Identify foreign Arabized words
5- If (wj ∈ ***FAW*_List**) then {Add wj to TEMP_LIST } // word is a foreign word
// handle 2-letter words
6- If (LastLetter == 2) then {
7- If (wj ∈ Two-letter_geminated_words_List) then {Stem_wj = Double_2_letter(wj), Root_wj = Stem_wj, Add wj to Norm_Words_list_1, Add wj to Norm_Words_list_2, Add Stem_wj to Stems_list_1, Add Root_wj to Roots_list_2, Add Root_wj to Roots_list_1, go to *}}
// apply *Enh_RB_2* method
8- If (Chosen root extraction method == ***Enh_RB_2*** method) {
9- Apply ***Enh_RB_2***(di) method}
// apply *IAA* method
10- Else if (Chosen root extraction method == ***IAA*** method) then {
11- Apply ***IAA*** method(di)}
12- Else {
13- Apply ***Enh_RB_2***(di) method
14- Apply ***IAA***(di) method}
15- * If (TEMP_LIST is not Empty) then {
// lists Norm_Words_list_1, Stems_list_1, Roots_list_1 are outputs // of *Enh_RB_2* method, lists Norm_Words_list_2, Roots_list_2 are //outputs of *IAA* method
16- Add TEMP_LIST to Norm_Words_list_1, Norm_Words_list_2, Stems_list_1, Roots_list_2, Roots_list_1}
17- Write all lists to respective output documents according to chosen method(s)}
END

***Enh_RB_2*(document di)** Algorithm

Inputs: document di, Normalization_list1, Root_lists, Function_words_List, ***Correction***_algorithm
Outputs: Lists of triliteral and quadriliteral roots, stems and normalized words for each new document new_d_i_1 separately

START
1- LastWord = Count_No_Words(d_i)
2- For j = 1 to LastWord in d_i do {
3- If (Numerals or non-arabic letters ∈ wj) then remove these
4- If (wj ∈ Function_words_List) then wj =""
5- LastLetter = Count_No_Letters(w_j,c)
// apply *Rule_Based* algorithm
6- Norm_wj = Normalization_List1(wj)
7- Stem_wj = Light_Stemmer(Norm_wj)
8- Root_wj = Infix_Remover(Stem_wj)
// apply *Correction* algorithm

```
9-        If    (Root_wj    ∉    Root_Lists)    then    Root_wj    =
Correction_algorithm(Root_wj)
10-       Add Norm_wj to Norm_Words_list_1
11-       Add Stem_wj to Stems_list_1
12-       Add Root_wj to Roots_list_1})
// below lists are to be outputted to the proposed system
13-   Return Norm_Words_list_1, Stems_list_1, Roots_list_1
END
```

***IAA*(document di)** Algorithm

```
Inputs: document di, Predefined letter groups weight lists,
Normalization_List2,        Root_Lists,        Function_words_List,
Correction_algorithm
Outputs: lists of triliteral and some quadriliteral roots and
normalized words

START
1- LastWord = Count_No_Words(di)
2- For j = 1 to LastWord in di do {
3-    If (Numerals or non-arabic letters ∈ wj) then remove these
4-    If (wj ∈ Function_words_List) then wj =""
5-    LastLetter = Count_No_Letters(wj,c)
6-    Norm_wj = Normalization_List2(wj)
// apply only steps 8-15 of EWBM2 algorithm
7-    Root_wj = EWBM2(Norm_wj) algorithm
// apply Correction algorithm
8-     If     (Root_wj     ∉     Root_Lists)     then     Root_wj     =
Correction_algorithm(Root_wj)
9-    Add Norm_wj to Norm_Words_list_2
10-   Add Root_wj to Roots_list_2}
11- *  LastLetter = Count_No_Letters(Root_wj,c)}
// below lists are to be outputted to the proposed system
12- Return Norm_Words_list_2, Roots_list_2
END
```

variants of weight-based approach were implemented in [14] using AT8 collection. The ***Adjusted Al-Shalabi*** method proved to be the highest in accuracy among all five original algorithms. However, ***Rule-based*** algorithm became the approach with the highest accuracy among all ten algorithms when ***Correction*** algorithm was included in it. Also, ***Correction*** algorithm improved performance of all algorithms especially ***Rule-based*** one by about 14% while it improved other algorithms' accuracy by 7% to 10%. It was observed that ***EWBM2*** algorithm had the following advantages:

1) higher sensitivity to handling *weak* words among the five original algorithms,
2) the highest capacity to extract *two-letter geminated* roots among the four original weight-based algorithms but lower than that of the ***Rule-Based*** algorithm,
3) partial success in extracting quadriliteral roots.

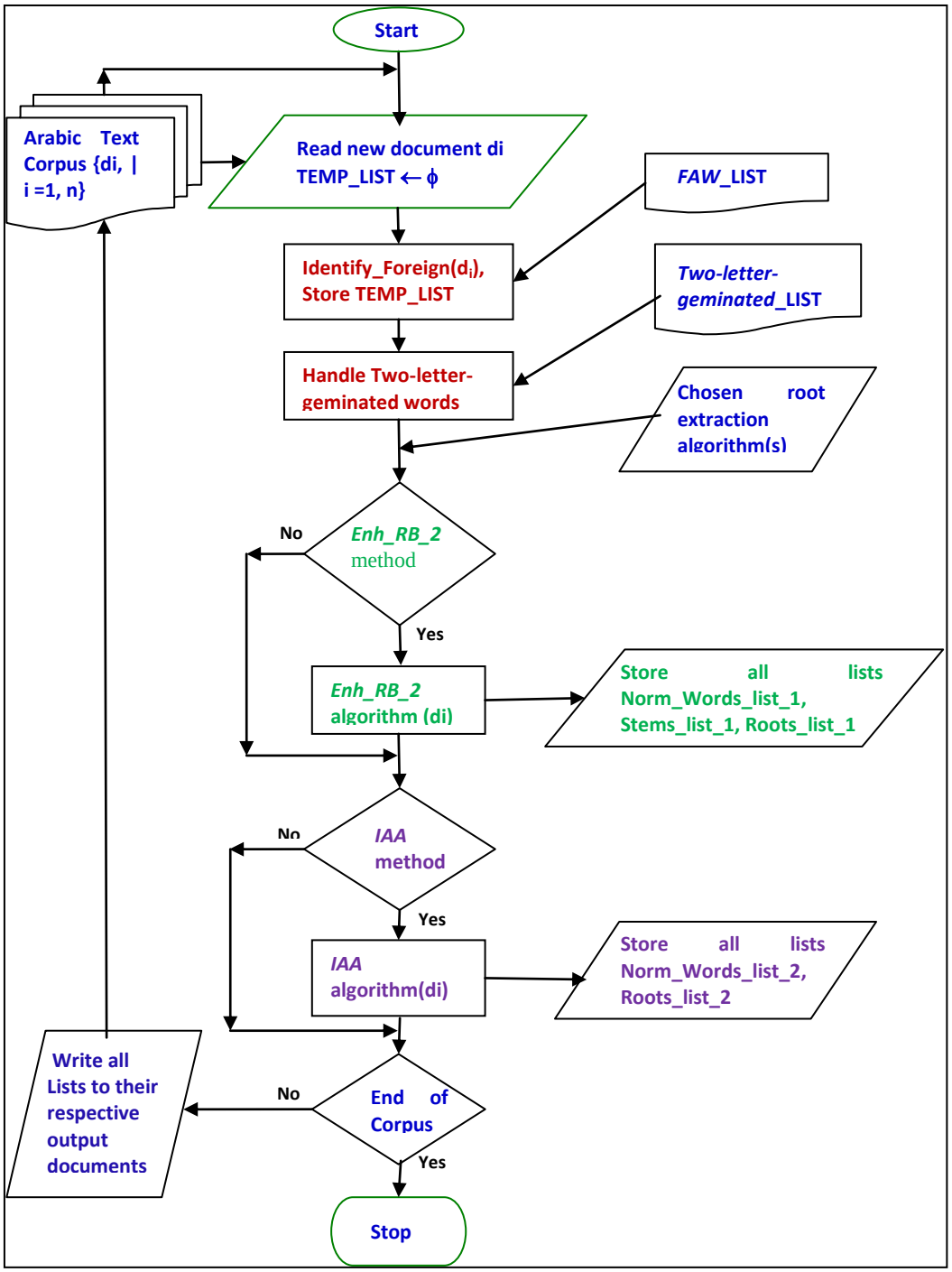


Figure 28: The Flowchart of Final Proposed Root Extraction System

LACC corpus was also used, which had much more words than AT8 collection, for implementing both ***Rule-based*** and ***Adjusted Al-Shalabi*** approaches and their ***Enhanced*** algorithms. The accuracy results of the two algorithms agree highly, as was shown above, with those when using AT8 collection. Execution time results for these algorithms is interesting since the efficiency of both algorithms in terms of the

effect of ***Correction*** algorithm is investigated by finding the percentage of the difference of execution time for each algorithm (i.e. with ***Correction*** one to the execution time for that algorithm without ***Correction*** as length increases). It was shown that the effect of ***Correction*** algorithm is similar in both algorithms for text lengths less than about 5,500. However, for length range 5,500 - 8,300 the effect of ***Correction*** algorithm is higher in ***Rule-Based*** algorithm than in ***Adjusted Al-Shalabi*** one since its percentage is less. Nevertheless, this effect is the opposite for lengths more than 8,300.

In future, the ***Rule-based*** approach can be improved by including more patterns in the infix remover, and handling the prefix-suffix paradigm. This can be performed by taking all possible prefix-suffix combinations and then deciding which is most appropriate according to a previously determined statistical value. In the weight-based algorithms, it is clear from the experimental results that the two proposed grouping of letters and their respective weights did not provide in general higher accuracy values. However, it was noted that for some words ***EWBM1*** method gave the correct root which for other words ***EWBM2*** provided their correct roots but in many others ***Adjusted Al-Shalabi*** method provided the correct root. This emphasizes the fuzzy nature of some letters in *sOltmwnyhA* and indicates perhaps that by using fuzzy sets to handle their grouping and weighting might provide higher accuracy and thus handle the prefix-suffix dilemma better. ***Correction*** algorithm is a promising efficient algorithm since it's highest reported improvement of the performance of original algorithms was 14%,. Its improvement can be increased by adding further rules and restrictions. Also, the proposed root extraction system is to be tested.

Chapter 5: Arabic Single-Label Text Classification Methods

5.1 Introduction

This chapter investigates the effect of implementing various classifiers for six different VSM representations on TC performance. Such representations are for the developed single-labeled Arabic corpus (presented in Chapter 3). The features' choices for VSM representations in this thesis are separately normalized words, stems, roots or extending such features by including their respective phrases. Also, such features in all representations here are weighted by a proposed TFIDF variant. Figure 29 briefly presents the steps needed for the implementation of TC methods. Part of the needed preprocessing steps in Figure 29 such as removing function words and extracting stems and roots for words were presented in Chapters 3 and 4.

This chapter is organized as follows: Section 5.2 describes preprocessing steps taken in order to provide the VSM representations of text documents and assign weights for features used/proposed for single-label TC. Section 5.3 briefly presents implemented classifiers as well as software tools for single-label TC. Section 5.4 presents the results of such implementations. Finally, it concludes with a brief presentation of such results. The detailed analysis of TC results will be presented in Chapter 6.

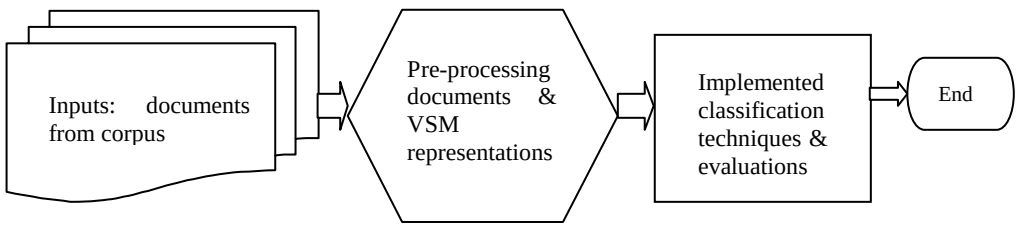


Figure 29: Basic Steps for Arabic TC classification

5.2 Pre-processing Steps

Applying classifiers on text documents requires first a preparation step of the documents. In this thesis, this is done by performing the following steps: 1- remove function words, punctuation marks, and numerals as was explained in Chapter 3, 2- perform a root extraction process to extract normalized words, their stems and roots as was presented and implemented in Chapter 4, 3- include the results of step 2 into normalized word, stem, and root lists as well as word phrases, stem phrases, and root phrases lists in order to further assign for each feature a weight and present each document in VSM representation, and 4- finally, present all documents in a single Attribute Relation File Format (ARFF) to be used by the Waikato Environment for Knowledge Analysis (WEKA) [91] software for classification (version 3.6.6). Steps 3 & 4 implemented in this thesis and mentioned above are explained next.

5.2.1 The Proposed Variant TFIDF Term Weighting Method

It was noticed that the generality among classes is not constant in the developed single-label corpus (presented in Chapter 3). This would have an impact on the process of developing the lists of words/stems/roots necessary to produce an ARFF file in terms of including the number of times each feature appeared in corpus and in each specific class. Thus, Table 28 below illustrates briefly the availability of these words in single-labeled documents. As noticed in Table 28, many words in corpus are not frequent. So, using the well-known *TFIDF* method for term weighting in document representation (presented in subsection 2.4.1.1) will result in weighting values of such words that do not reflect their presence among such classes. Thus, a variant-*TFIDF* method for weighting terms that includes such effect is proposed and implemented in this thesis as presented in eq. (1) below.

$$\text{var\_}tfidf\ (t_k, d_j) = \#(t_k, d_j).\log\left(\frac{N_i}{\#_{C_i}(t_k)}\right) \quad (1)$$

Where $\#(t_k,d_j)$: number of times term t_k occurs in d_j,
$\#_{Ci}(t_k)$ (called document frequency ($df(t_k)$): number of documents in class C_i that t_k occurs in,
N_i: number of documents in class C_i,

It is noteworthy that although Do and Ng [58] mentioned the $\frac{N_i}{\#_{C_i}(t_k)}$ part in eq. (1) above as one of possible methods for weighting terms but did not implement it. As far as is known, this proposed simple variant of *TFIDF* is implemented for term weighting here for the first time to be used in Arabic single-label TC. Results of implementing such term weighting are presented in section 5.4.

5.2.2 Document Representation

5.2.2.1 Features Implemented Using Single Terms

The works that used stemming for DR on Arabic texts, such as [63], [108], [135], [146], [147], [148], [154] and [171], as was explained in section 2.4.1.2, compared the effect of using words, stems or roots on Arabic TC performance. However, these works did not reach the same conclusion regarding the effect of such features on Arabic single-label TC performance. None of these works reported significance testing especially those that concluded that using roots provided best results for TC compared to that when using stems or words. Thus, here the use of different VSM representations using separately normalized words, stems or roots for features and the comparison of the effect of feature choice on TC performance is performed. If an improvement occurred, then significance testing is performed.

The results of constructing the lists of normalized words, stems and roots for the Arabic corpus 804 texts is presented in Table 28. The number of different normalized words is 117,724, the number of different stems is 18,019, and the number of different roots is 11,063. So, finally three ARFF files are ready to be used

for single-label TC with dimensions 804 x 117,724, 804 x 18,019, and 804 x 11,063 for normalized words, stems, and roots respectively.

List	# Terms	Ratio (%)	List (DF >1)	# Terms	Reduction of Terms (%)
Roots	**11,063**	**9.4 (to words list)**	**Roots**	**7,294**	**34.1**
Stems	**18,019**	**15.3 (to words list)**	**Stems**	**12,079**	**32.96**
Words	**117,724**	**-**	**Words**	**54,140**	**54**

Table 28: Number of different original implemented terms available in feature lists processed from corpus

From Table 28, the ratio of roots to words is 1 to 10.64 whereas the ratio of stems to words is 1 to 6.53. This is in agreement with the percentages presented in section 2.3.4. The results of implementing such representations are presented in section 5.4. Also, an extension of the above VSM representations is performed in this Chapter by adding their respective phrases and an investigation of their effect on TC performance is conducted. This will be discussed next.

5.2.2.2 Extending VSM Representation Using Phrases

As can be seen from Table 28 DF of words in Arabic texts is relatively low. This lead us to propose representing features in documents by phrases instead of words and investigate this representation' effect on single-label TC performance.

The method that was implemented in order to extract phrases is as follows: each three consecutive normalized words in a text (after removing function words, punctuation marks, etc) are presented by three two-word phrases. An example presenting this idea is for the phrase "استبعد رئيس الحكومة الإسرائيلية أيهود اولمرت": (transliterated *AstbEd r}ys AlHkwmp Al<srA}ylyp >yhwd Awlmrt*) this phrase is presented by the following six phrases "استبعد رئيس" ، "رئيس الحكومة" ، "استبعد الحكومة" ، "الحكومة الإسرائيلية" ، "الحكومة أيهود" ، "الإسرائيلية أيهود" ، "أيهود اولمرت" respectively (transliterated *AstbEd r}ys* , *r}ys AlHkwmp* , *AstbEd AlHkwmp* , *AlHkwmp Al<srA}ylyp* , *AlHkwmp >yhwd* , *Al<srA}ylyp >yhwd* , *>yhwd Awlmrt*) respectively. Table 29 presents an example of how two-word

phrases are chosen from a paragraph in a text. The same is performed for texts presented by roots as well as those presented by stems.

استبعد رئيس الحكومة الإسرائيلية أيهود اولمرت حدوث تصعيد في الأوضاع الأمنية على الجبهة والحدود الشمالية لإسرائيل في المستقبل القريب بسبب انشغال "حزب الله" بالأزمات السياسية الداخلية في لبنان ، مبينا أن "حزب الله" لم يقم حتى ألان بالرد على مقتل القائد العسكري في الحزب عماد مغنية خشية من رد فعل إسرائيلي على تنفيذ أي عملية انتقام
(a)
استبعد رئيس الحكومة الإسرائيلية أيهود اولمرت حدوث تصعيد الأوضاع الأمنية الجبهة والحدود الشمالية لإسرائيل المستقبل القريب انشغال حزب الله بالأزمات السياسية الداخلية لبنان مبينا حزب الله يقم بالرد مقتل القائد العسكري الحزب عماد مغنية خشية رد فعل إسرائيلي تنفيذ عملية انتقام
(b)
"استبعد رئيس" "رئيس الحكومة" "استبعد الحكومة" "الحكومة الإسرائيلية" "الحكومة أيهود" "الإسرائيلية أيهود" "أيهود اولمرت" "أيهود حدوث" "اولمرت حدوث" "حدوث تصعيد" "تصعيد الأوضاع" "حدوث الأوضاع" "الأوضاع الأمنية" "الأمنية الجبهة" "الأوضاع الجبهة" "الجبهة والحدود" "والحدود الشمالية " "الجبهة الشمالية " "الشمالية لإسرائيل" "لإسرائيل المستقبل" "الشمالية المستقبل" "المستقبل القريب" "القريب انشغال" "المستقبل انشغال" "انشغال حزب" "حزب الله" "انشغال الله" "الله بالأزمات" "بالأزمات السياسية" "الله السياسية" "السياسية الداخلية" " الداخلية لبنان" "السياسية لبنان" "لبنان مبينا" "مبينا حزب" "لبنان حزب" "حزب الله" "الله يقم" "حزب يقم" "يقم بالرد" "بالرد مقتل" "يقم مقتل" "مقتل القائد" "القائد العسكري" "مقتل العسكري" "العسكري الحزب" "الحزب عماد" "العسكري عماد" "عماد مغنية" "مغنية خشية" "عماد خشية" "خشية رد" "رد فعل" "خشية فعل" "فعل إسرائيلي" "إسرائيلي تنفيذ" "فعل تنفيذ" "تنفيذ عملية" "عملية انتقام" "تنفيذ انتقام"
(c)

Table 29: A paragraph taken from Addustour newspaper: (a) original paragraph (55 words), (b) paragraph (40 words, 19 three-word phrases) after removal of function words, punctuation marks, short vowels and/or numerals, (c) paragraph after words are put into two-word phrases (60 phrases) *(here phrases are put between double quotes for illustration)*

After such phrases are chosen, these phrases are included in lists to investigate their numbers in such texts and along categories and corpus. The results are presented briefly in Table 30. Also, as conducted for the lists of roots, stems, and words, an investigation of the document frequency for such phrases in corpus is performed.

List	# Terms	Ratio (%)	List (DF >1)	# Terms	Reduction of Terms (%)
RP	**655,923**	**83.8 (from WP)**	**RP**	**39,028**	**94**
SP	**799,314**	**102 (from WP)**	**SP**	**25,236**	**96.8**
WP	**782,969**	**-**	**WP**	**6,873**	**99**

Table 30: Number of different proposed terms available in feature lists processed from corpus

From Tables 28 & 30, the ratio of reduced roots to reduced root phrases is 1 to 5.4; the ratio of reduced stems to reduced stem phrases is 1 to 2.1, whereas the ratio of reduced words to reduced word phrases is 1 to 0.13. The number of phrases, stem-phrases, or root-phrases is very large to be used instead of words, stems, or roots respectively. Also, from Tables 28 and 30, the ratio of roots to root phrases is 1 to 59.3, the ratio of stems to stem phrases is 1 to 44.4, and the ratio of words to word phrases is 1 to 6.7.

The relatively much smaller number of reduced phrases, stem-phrases, or root-phrases (i.e. with DF>1) lead us, instead of representing documents by phrases, to propose to extend the original representation of features in texts through including at the end of each VSM the representation of their respective reduced phrases. An example illustrating this proposal is suppose a VSM representation of a document using words would be <politics, **0, 0, 1.34, 0, 3.87, 0, 0, 0, 0, 7.39**> and the VSM for the same document using only phrases with DF greater than 1 be <politics, *0, 0, 0, 3.25, 11.11, 0, 0, 4*>, then the proposed VSM representation for this document would be <politics, **0, 0, 1.34, 0, 3.87, 0, 0, 0, 0, 7.39**, *0, 0, 0, 3.25, 11.11, 0, 0, 4*>. So, finally three ARFF files are ready to be used for single-label TC with dimensions 804 x 124,598, 804 x 43,256, and 804 x 50,091 for normalized words and phrases, stems and stem phrases, and roots and root phrases respectively. The results of implementing such representations are presented in section 5.4.

5.2.3 Implemented Feature Subset Selection Method

There was consensus among research works for Arabic TC that implemented various FSS methods, that using Chi-square method for FSS improved single-label TC performance [135], [137], [136], [8], [98], [174], [146], [147], [148] and [184] (as was presented in subsection 2.4.1.2). Thus, it was decided to use this method here for FSS. The results of using this FSS method is presented in section 5.4.

5.2.3.1 Chi-square Method

Chi-square function [161] is defined as shown in eq. 2 below

$$\chi^2(t_k, c_i) = \frac{|Tr| .[P(t_k, c_i).P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i).P(\bar{t}_k, c_i)]^2}{P(t_k).P(\bar{t}_k).P(c_i).P(\bar{c}_i)} \qquad (2)$$

Where $|Tr|$: number of documents in training set,
$p(t_k, c_i)$ is Probability that kth term of document d_j occurs from class c_i

This function is available in WEKA software and is implemented here where using the outputs of this function, the effect of varying the number of selected features on single-label TC performance is investigated.

5.3 Applied Text Classification Methods

Seventy five classifiers that are available in WEKA software are implemented here where the developed Arabic corpus (presented in Chapter 3) is used. The results of only forty seven classifiers under six types are presented here. The remaining results are not included either because the performance of such classifiers is poor for F1-measure (i.e. << 0.5), or couldn't be implemented due to either such classifiers are not available in the WEKA version used or these require different representation or binary labeling. Lazy learners such as k-NN and Multi-Instance (MI) learners in WEKA are not among the presented ones in this thesis since either their classifiers were tested to have poor performance or are not applicable. Thus, only six types of classifiers are investigated here. Examples of poor F1 values for such classifiers are (0.1) for k-NN[32] [4] classifier, and (around 0.12) for Classification Via Clustering classifier (CVC)[33].

5.3.1 Single-Label Classification Methods

In this part two experiments are conducted. The criteria of tested classifiers are consistent in both experiments (see appendix IV for performance criteria used for such classifiers). All used classifiers are presented briefly in this section, and these classifiers are further presented and compared in Chapter 6 where their results in original research works are presented in order to compare the results in this thesis with those of such classifiers in previous works. The first experiment tests the six

[32] k-NN classifier is named in WEKA IBK, further info can be found at: http://weka.sourceforge.net/doc/weka/classifiers/lazy/IBk.html
[33] Further info on CVC classifier can be found at: http://wiki.pentaho.com/display/DATAMINING/ClassificationViaClustering

ARFF VSM representations discussed above using forty seven classifiers (using the explorer in WEKA software) according to their type. In this experiment, the effect of varying the number of selected features on TC performance is investigated. This is performed since the proposed term weighting method discussed above has not been implemented before and as far as known many of the classifiers used here were not performed previously for Arabic such as the rule learner NNge[34] [130].

The second experiment investigates further the same forty seven classifiers (used in first experiment), in performance to conclude which of the: a) classifiers, b) six VSM representations provided the best performance. This is performed in three parts and implemented using the experimenter in WEKA software and results are tested for significance. The first part compares the performance of classifiers of same type. FSS is performed using Chi-square on all those VSM representations and only best 1000 and 5000 selected features are maintained here. In the second part, the best two performing classifiers among representations for each type are chosen and the comparison among types are performed and tested for significance. The third part presents briefly the results of significance testing for each classifier between root and RRP representations, stem and SSP representations, and finally word and WP representations. In this experiment, two-tail statistical (corrected) t-test is conducted with significance level of 0.05.

5.3.1.1 Implemented Classifiers

Forty seven different classifiers are applied on the six ARFF files. **Five of these classifiers are Function** classifiers, namely SMO [110]; [142], Logistic [123], Multi Layer Perceptron[35] (MLP), Simple Logistic [119], and RBF network[36] classifiers.

[34] Further info on NNge can be found at: http://weka.sourceforge.net/doc.packages/NNge/weka/classifiers/rules/NNge.html.
[35] Further info on MLP can be found at: http://weka.sourceforge.net/doc/weka/classifiers/functions/MultilayerPerceptron.html.

Six Rule classifiers are also applied, namely the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) rule learning classifier [47] (in WEKA it is named JRip)[37], the rule learning algorithm [75] that builds C4.5 partial tree (named in WEKA PART)[38], RIpple-DOwn Rule learner (named in WEKA Ridor) [80], a classifier that builds and uses a 1R classifier (named in WEKA OneR) [102], Nearest-Neighbor like algorithm using non-nested Generalized Exemplars (named in WEKA NNge) [130], and Decision Table [114].

Another type of classifiers that is implemented here is the one based on **Bayes theory**, namely Naïve Bayes (NB) [106], Bayes Net (BN) [181], Naïve Bayes Multinomial (NBM) [132], Complement NB [149], NBMUpdatable [132], and NBUpdatable [106].

Eleven Tree classifiers are also implemented, namely the Reduced Error Pruning Tree algorithm (named in WEKA REPTree)[39], the Random Forest tree learner (in WEKA is named RandomForest)[40] [38] which is a classifier that consists of a collection of tree-structured classifiers with no pruning, the C4.5 tree classifier (named in WEKA J48)[41], Best-First decision Trees (named in WEKA BF Tree) [164]; [77], Functional Trees (named in WEKA FT) [82], grafted C4.5 decision tree (named in WEKA J48 graft) [180], Logit boost Alternating Decision Tree (name in WEKA LAD Tree) [101], Logistic Model Trees (named in WEKA LMT) [118];

---

[36] further ino regarding RBFnetwork can be found at: http://weka.sourceforge.net/doc/weka/classifiers/functions/RBFNetwork.html

[37] Further info on JRip can be found at: http://classes.engr.oregonstate.edu/eecs/winter2003/cs534/weka/weka-3-3-4/doc/weka.classifiers.rules.JRip.html [last accessed 7/5/2012].

[38] Further info on PART can be found at: http://weka.sourceforge.net/doc.dev/weka/classifiers/rule/PART.html [last accessed 7/5/2012].

[39] Further info on RepTree can be found at: http://classes.engr.oregonstate.edu/eecs/winter2003/cs534/weka/weka-3-3-4/doc/weka.classifiers.trees.REPTree.html [last accessed 7/5/2012].

[40] Further info on RandomForest can be found at: http://www.hsc.wvu.edu/mbrcc/fs/GuoLab/pdfs/Software%202.pdf [last accessed 7/5/2012].

[41] Further info on J48 (C4.5) can be found at: http://weka.sourceforge.net/doc/weka/classifiers/trees/J48.html [last accessed 7/5/2012]

[119], Naive Bayes Tree (named in WEKA NB Tree) [115], Random Tree[42], and Classification And Regression Trees (named in WEKA Simple Cart) [40].

Two Miscellaneous classifiers are used here namely Voting Feature Interval Classifier (named in WEKA VFI)[43], and Hyper Pipes[44] (HP and used in [65]).

Finally, **seventeen Meta** classifiers are used here, namely AdaBoost.M1 [78], Attribute Selected Classifier[45] (ASC), Bagging [39], Classification Via Regression (CVR) [76], Dagging [177], Decorate [133]; [134], END [59]; [74], Filtered Classifier[46] (FC), Logit Boost (LB) [77], Multi Class Classifier[47], Class Balanced Nested Dichotomies (named in WEKA ClassBalancedND) (CBND) [59]; [74], DataNearBalanced ND (DNBND) [59]; [74], Nested Dichotomies (ND) [59]; [74], Ordinal Class Classifier (OCC) [73], a classifier that consists of multiple trees constructed pseudo randomly selecting subsets of components of feature vector (named in WEKA RandomSubSpace) (RSS) [99], Random Committee[48] (RC), and Rotation Forest (RF) [150].

Results of implementing experiments are presented in section 5.4.

5.4 Results of Implementations

The implementation of single-label TC on proposed and prepared VSM representations discussed in subsection 5.2.2 is performed through two experiments. The first experiment applies forty seven classifiers (using the explorer of the WEKA

[42] further info on Random tree can be found at: http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomTree.html.
[43] Further info on VFI can be found at: http://roust.gotdns.com/weka-doc/weka/classifiers/misc/VFI.html.
[44] Further info on HyperPipes can be found at: http://weka.sourceforge.net/doc/weka/classifiers/misc/HyperPipes.html.
[45] further info on Attribute selected classifier can be found at:
http://weka.sourceforge.net/doc/weka/classifiers/meta/AttributeSelectedClassifier.html.
[46] further info on Filtered classifier can be found at:
http://weka.sourceforge.net/doc/weka/classifiers/meta/FilteredClassifier.html.
[47] further info on MultiClas Classifier is found at:
http://weka.sourceforge.net/doc.stable/weka/classifiers/meta/MultiClassClassifier.html
[48] further info on Random committee can be found at:
http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/RandomCommittee.html.

software) on the six representations where the effect of varying the number of best selected features on TC performance is investigated. The results of such application in terms of weighted-F1 are presented in Figures 30 to 35 as well as Tables 31 and 32. The second experiment investigates further: 1- which of the forty seven classifiers among the six representations provides the best performance, and 2- which of representations provides best significant performance results. This is implemented using the experimenter of the WEKA software and results are tested for significance. Results of $F1^M$ measure for these classifiers are presented in Tables 33 and 34.

5.4.1 Results of Implemented Single-Label Classification Methods

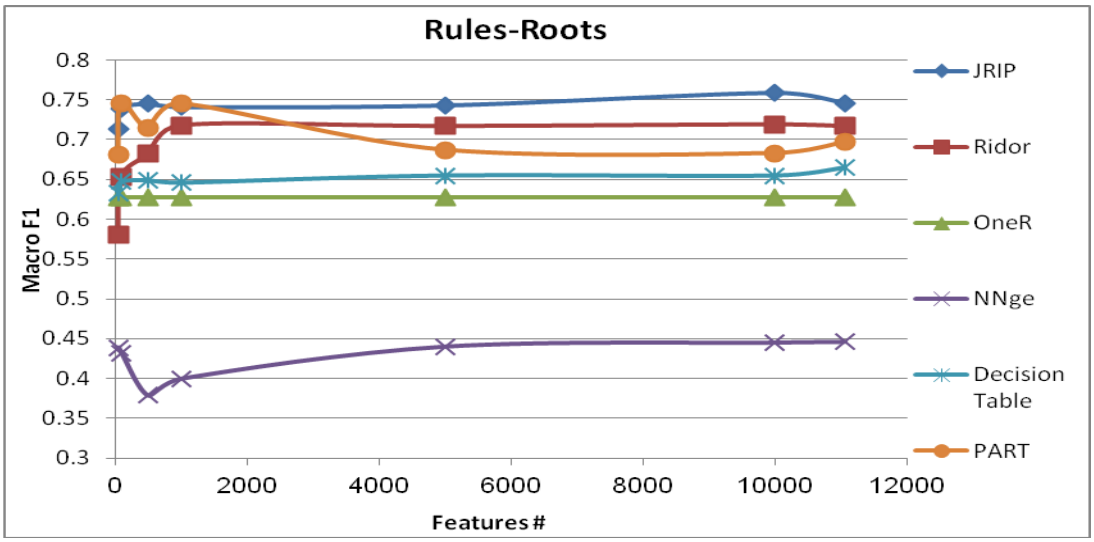
In the two experiments, stratified 10-fold stratified cross validation is used for all implemented classifiers on all six representations. The choice of stratified 10-fold cross validation is the same as some research works on Arabic TC. This method is chosen here due to: the relatively small size of corpus, the different number of texts among categories, and the method's relatively low bias and variance [92]. The criteria chosen for these classifiers such as the number of epochs, stopping criteria,..etc whether for the first experiment or second are shown in appendix IV. For evaluation, weighted-macro average F1-measure is used to compare the performance of such classifiers in the first experiment (among others as training time, root mean square error, percent correct, .. etc) whereas Macro F1-measure is used, among others, in the second experiment. The other performance measures effects are presented and critically analyzed in Chapter 6.

5.4.1.1 First Experiment

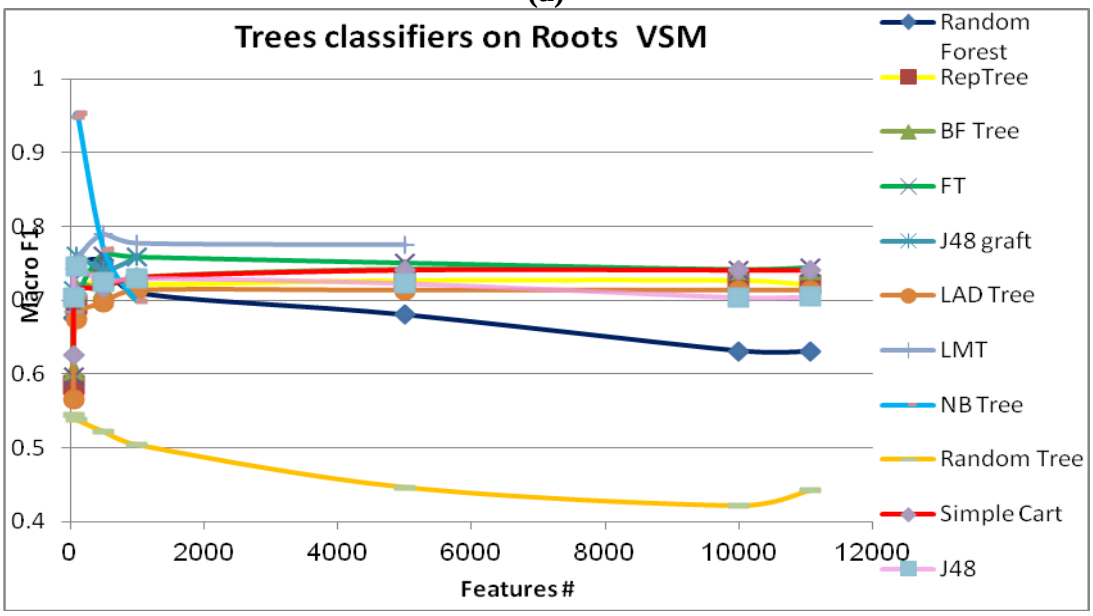
This experiment is composed of two parts. The first part investigates and compares the performance of classifiers of the same type for each representation as the number

of best selected features is increased (presented in Figures 30-35 below), and the second part investigates and compares the performance of each classifier among the six representations also as the number of best selected features is increased (presented in Figure 36 and appendix IV). As far as is known, no reports were presented for most of these classifiers on Arabic text classification studies. Also, among the categories and for only 1000 selected features, the performance of these classifiers is presented in appendix IV where due to space limitations the rest of results are not shown. The reason why only this was performed for 1000 features is explained at end of this subsection. Further analysis of results of Figures 30-36 are shown in Tables 31 and 32.

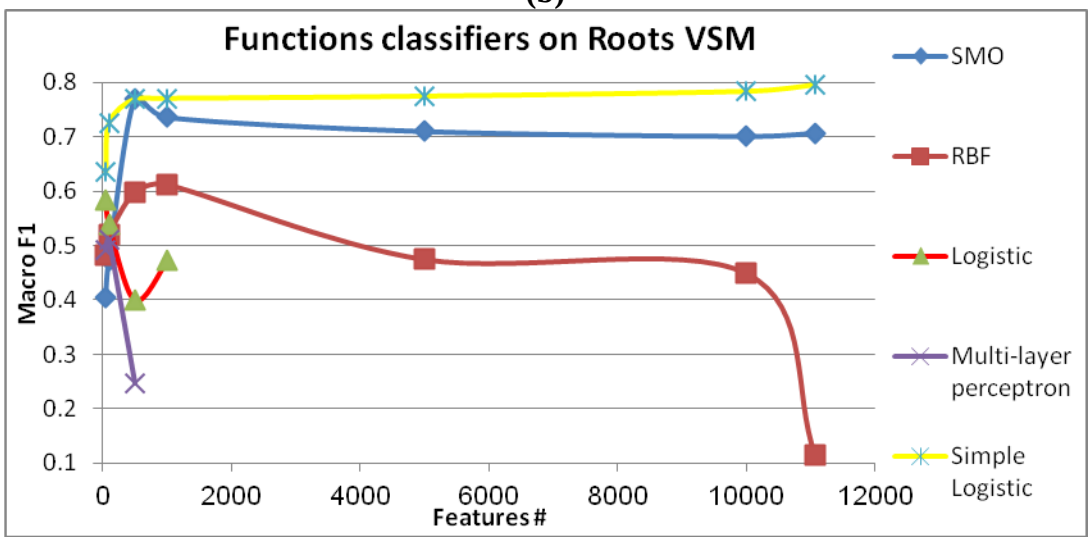
The performances of classifiers according to their type are introduced separately as shown in Figures 30-35. Figure 30 presents the performance of classifiers according to their type for Root representation, Figure 31 for Stem representation, Figure 32 for Word representation, Figure 33 for RRP representation, Figure 34 for SSP representation, and Figure 35 for WP representation (kindly refer to appendix IV for the display of all classifiers performance for each representation). It can be concluded that the performance of some of these classifiers degrade when terms are extended by including their respective phrases while for others the performance improve. Also, since some of the VSMs used here have high dimensionality and as such some classifiers require extensive calculations, then such implementation of classifiers is limited by available RAM and PC speed. This resulted in that some of these classifiers didn’t provide results for such representations as number of selected features was increased such as the NB Tree, Logistic, or MLP classifiers.


Rules-Roots
Macro F1
Features #
JRIP
Ridor
OneR
NNge
Decision Table
PART

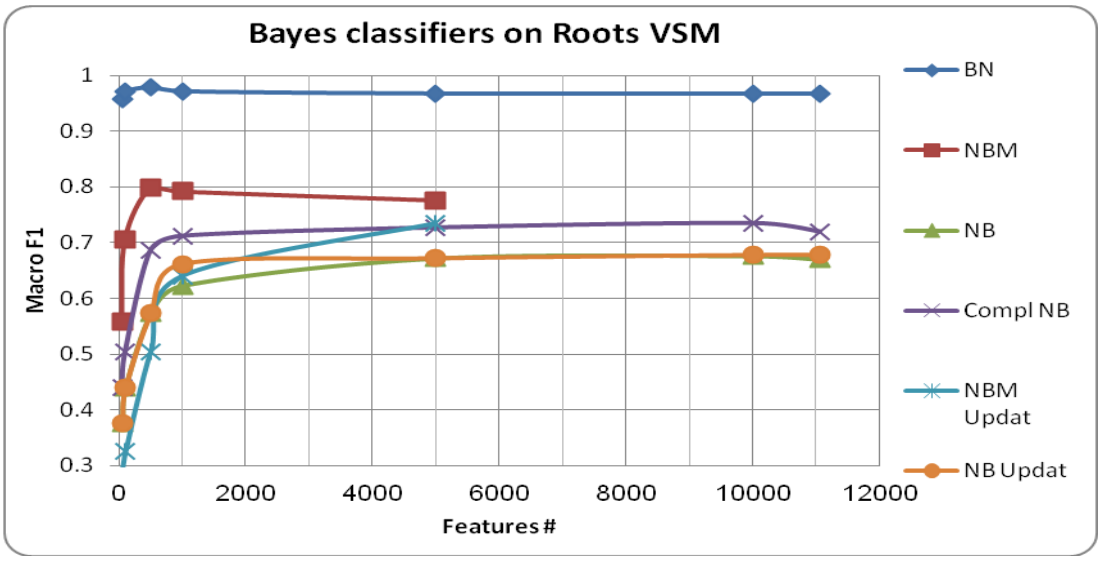
(a)


Trees classifiers on Roots VSM
Macro F1
Features #
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree
Random Tree
Simple Cart
J48

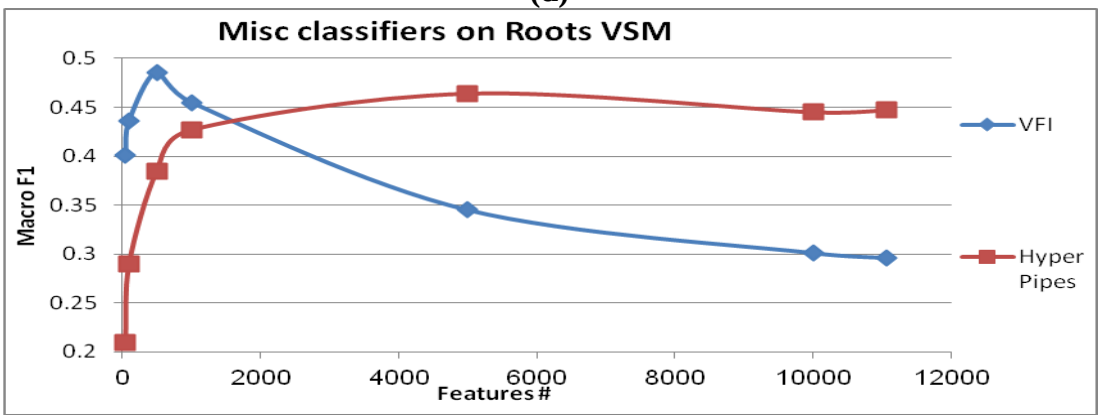
(b)


Functions classifiers on Roots VSM
Macro F1
Features #
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic

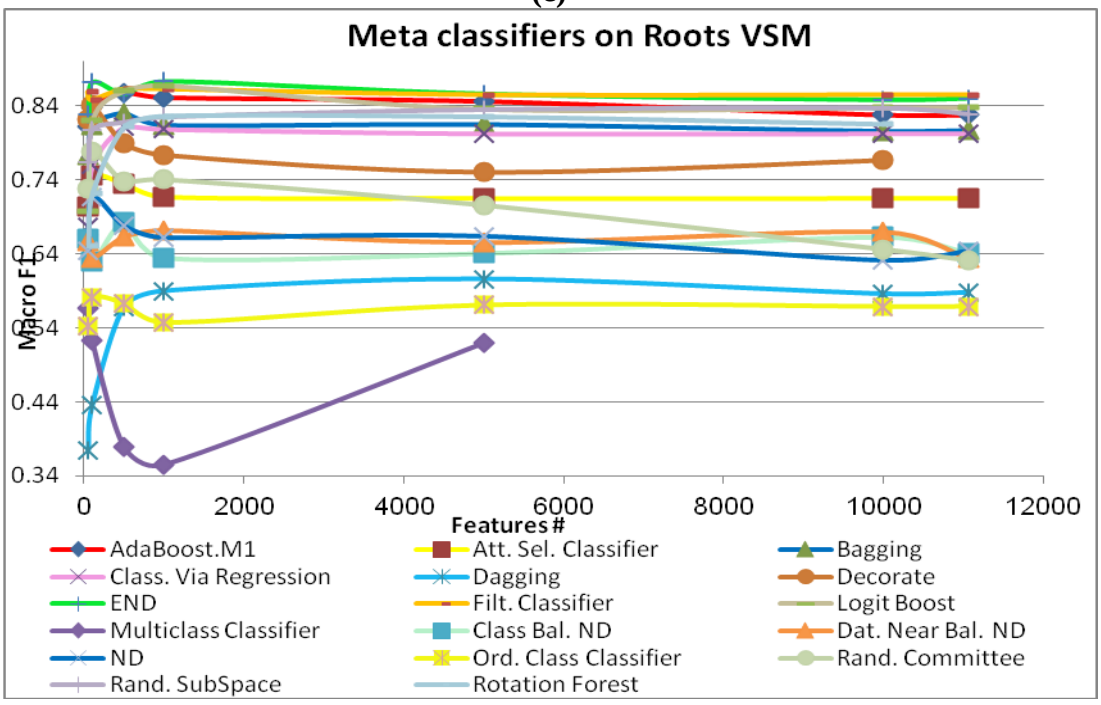
(c)



(d)



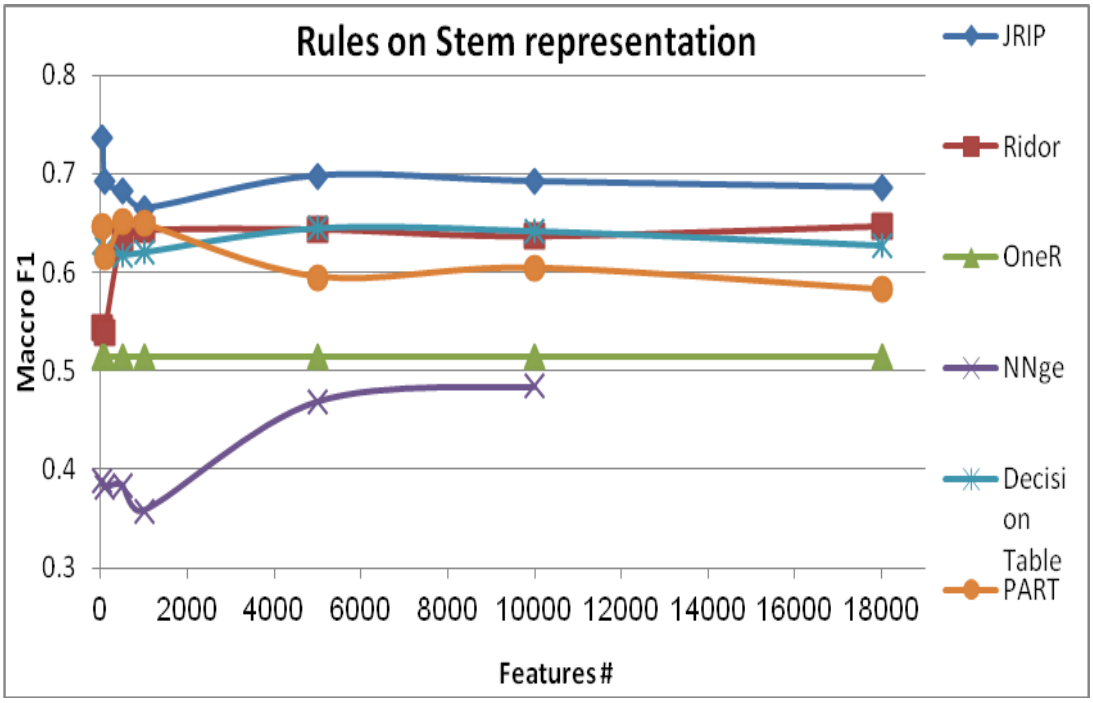
(e)



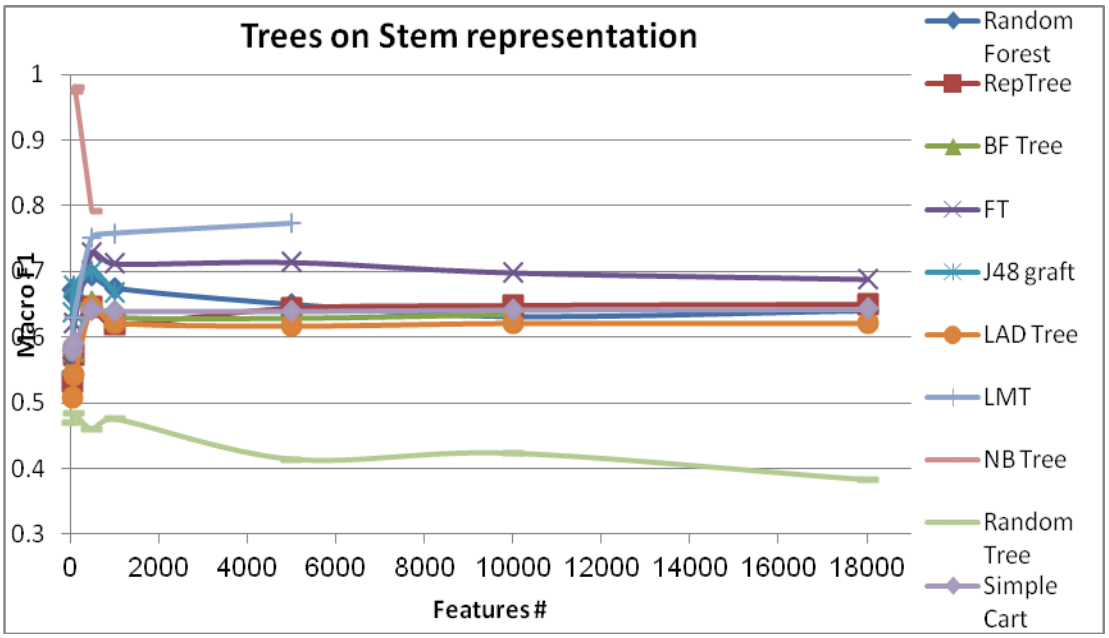
(f)

Figure 30: Comparison between classifiers' performance for Root VSM representation according to their type (a) rules, (b) trees, (c) functions, (d) Bayes-based, (e) miscellaneous, (f) meta

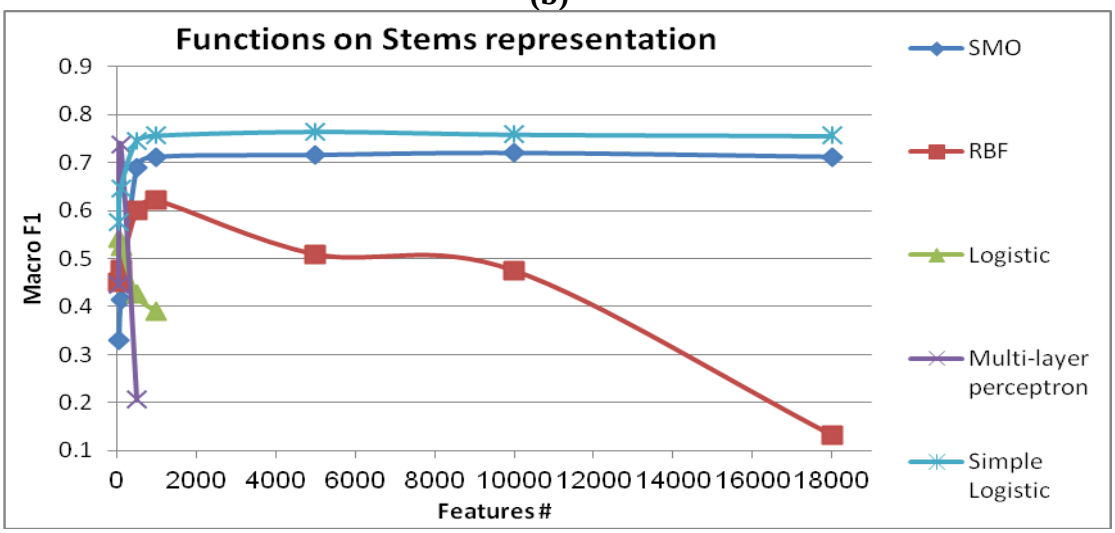
Figure 30 presents the performance of all classifiers for the Root representation. From Figure 30a, the three classifiers with best performance among rule-learners are JRip, Ridor, and PART respectively. In Figure 30b, the four classifiers with best performance among tree learners are LMT, FT, Simple Cart, and RepTree respectively. However, from Figure 30c, the best performance of function learners is for the two classifiers Simple Logistic and SMO respectively. In Figure 31d, the performance of Bayes-based learners is compared. The three classifiers with best performance are BN, NBM, and Complement NB respectively. Miscellaneous learners are presented in Figure 30e and among the two learners, HP classifier performs better. In Figure 30f, Meta classifiers performances are presented and the best seven classifiers in performance are END, FC, RSS, LB, AdaBoost.M1, RF, and Bagging.



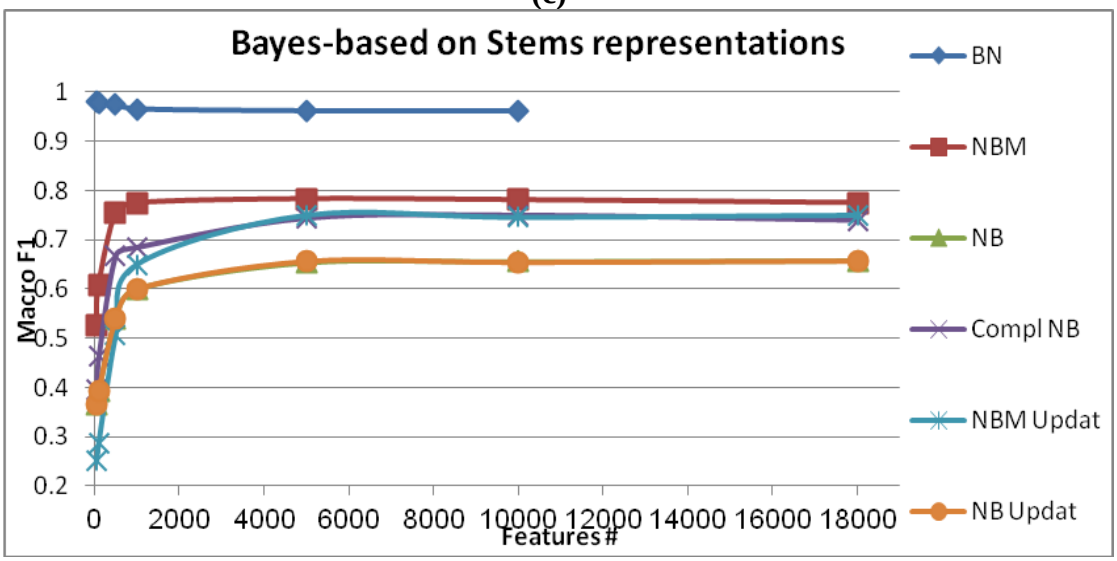
(a)


Trees on Stem representation
Macro F1
Features #
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree
Random Tree
Simple Cart

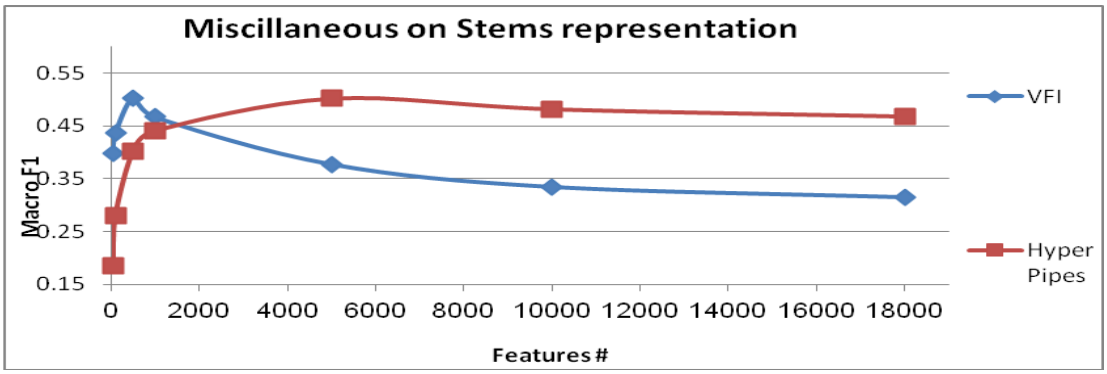
(b)


Functions on Stems representation
Macro F1
Features #
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic

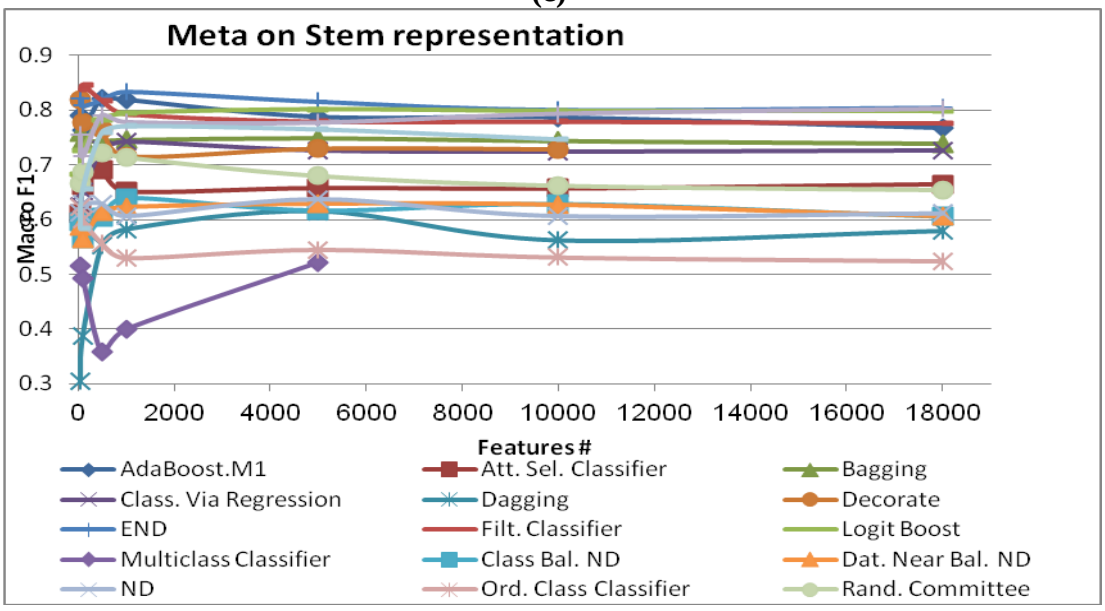
(c)


Bayes-based on Stems representations
Macro F1
Features #
BN
NBM
NB
Compl NB
NBM Updat
NB Updat

(d)



(e)

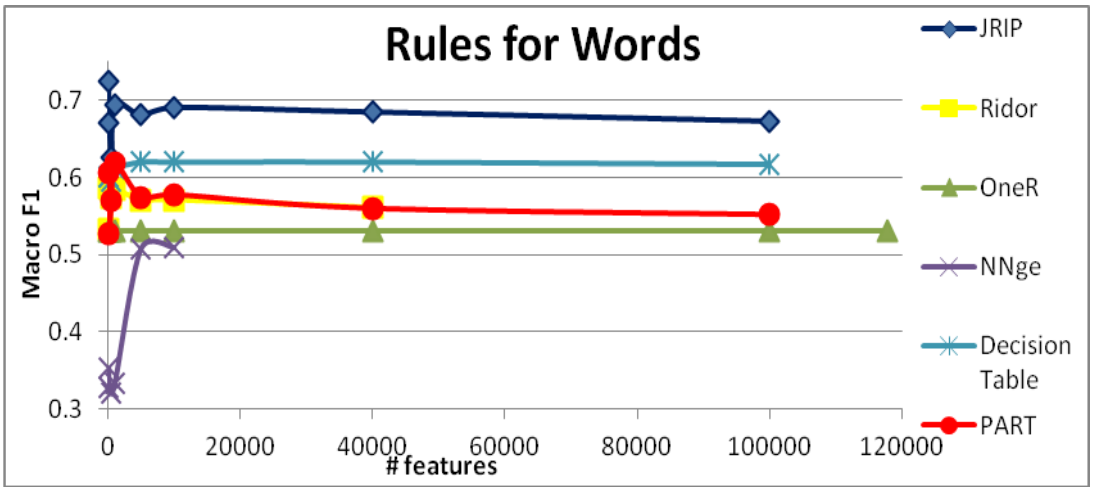


(f)

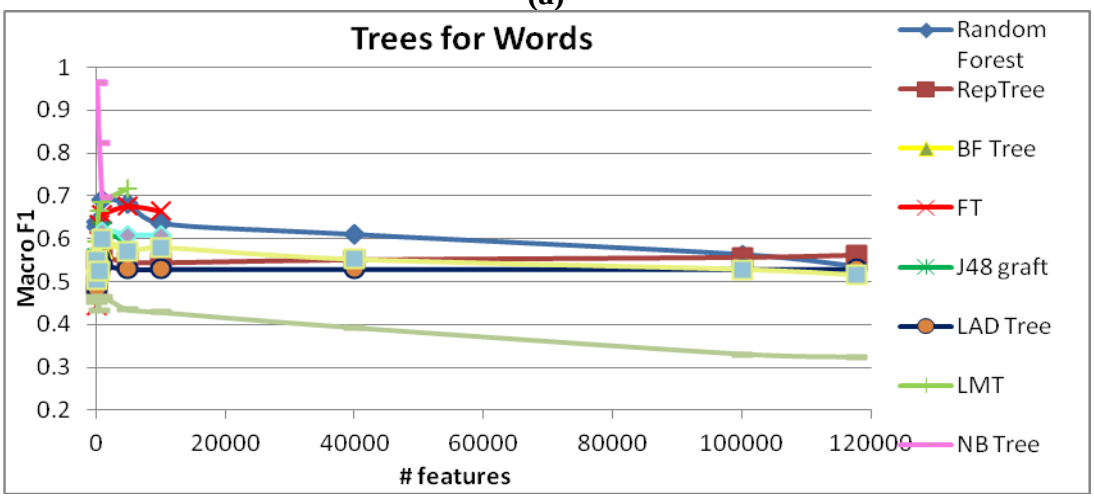
Figure 31: Comparison between classifiers' performance for Stem VSM representation according to their type (a) rules, (b) trees, (c) functions, (d) Bayes-based, (e) miscellaneous, (f) meta

Figure 31 presents the performance of all classifiers for Stem representation. From Figure 31a, the three classifiers with best performance among rule-learners are JRip, Decision Table, and Ridor respectively. In Figure 31b, the four classifiers with best performance among tree learners are LMT, FT, RepTree, and Simple Cart respectively. However, from Figure 31c, the best performance of function learners is for the two classifiers Simple Logistic and SMO respectively. In Figure 31d, the performance of Bayes-based learners is compared. The three classifiers with best performance are BN, NBM, and NBMU respectively. Miscellaneous learners are presented in Figure 31e and among the two learners, HP classifier performs better. In

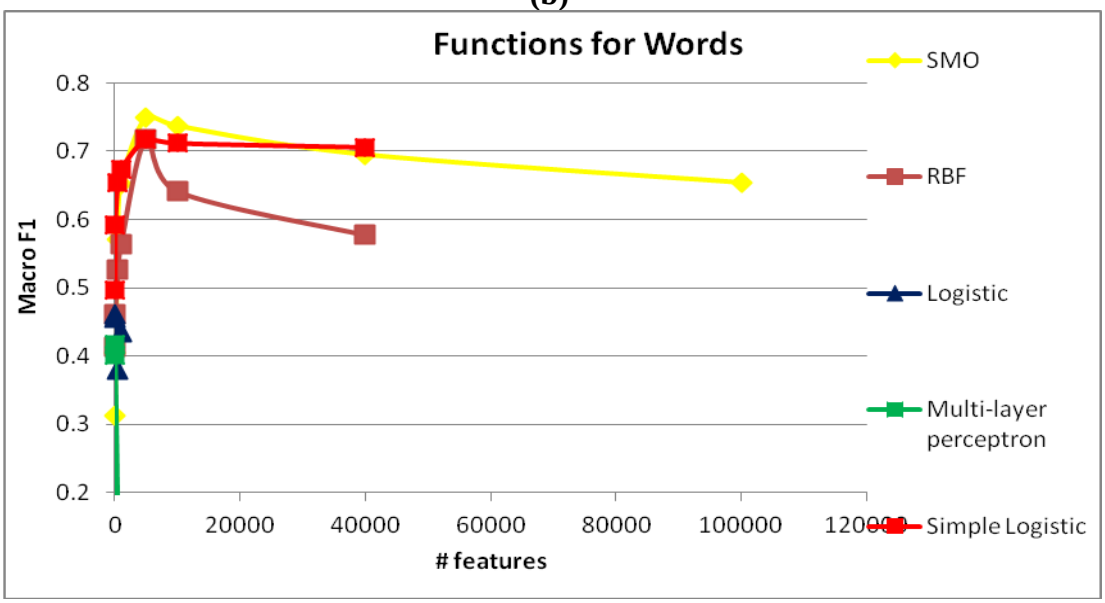
Figure 31f, Meta classifiers performances are presented and the best seven classifiers in performance are END, LB, RSS, AdaBoost.M1, FC, RF, and Bagging.



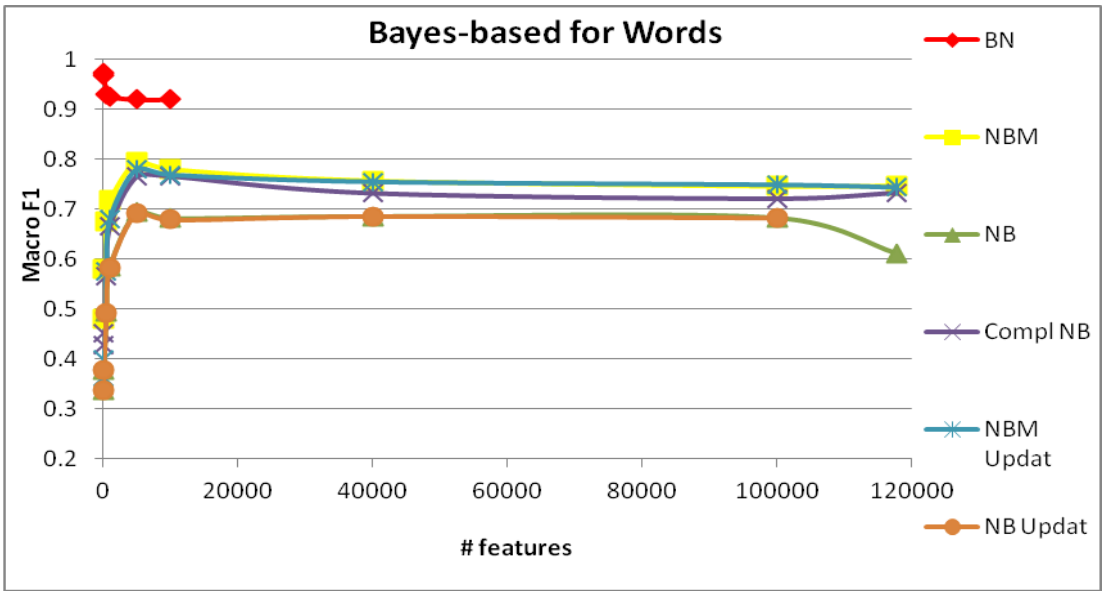
(a)



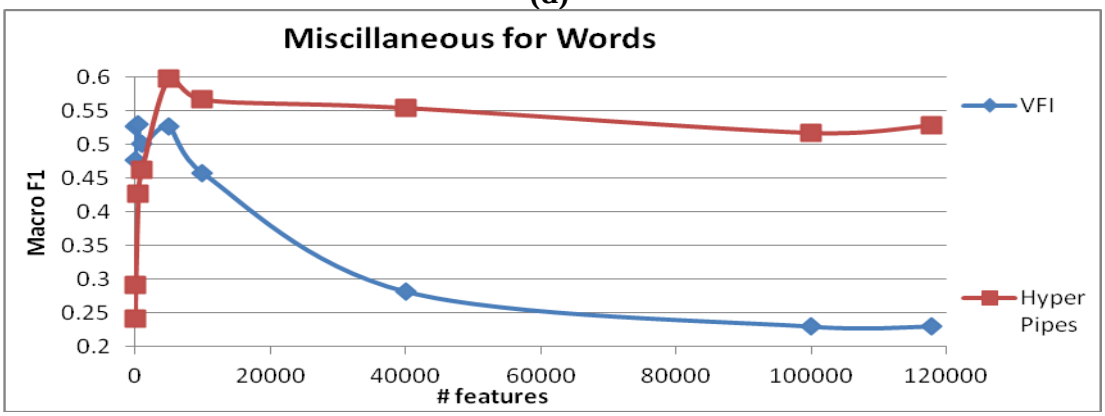
(b)



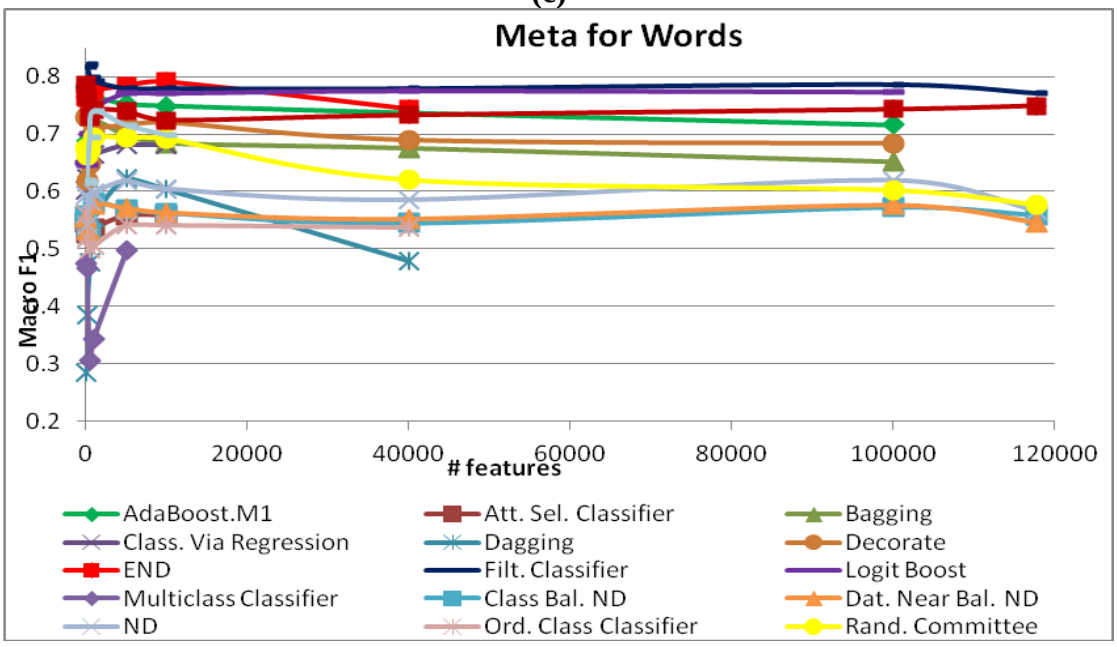
(c)



(d)



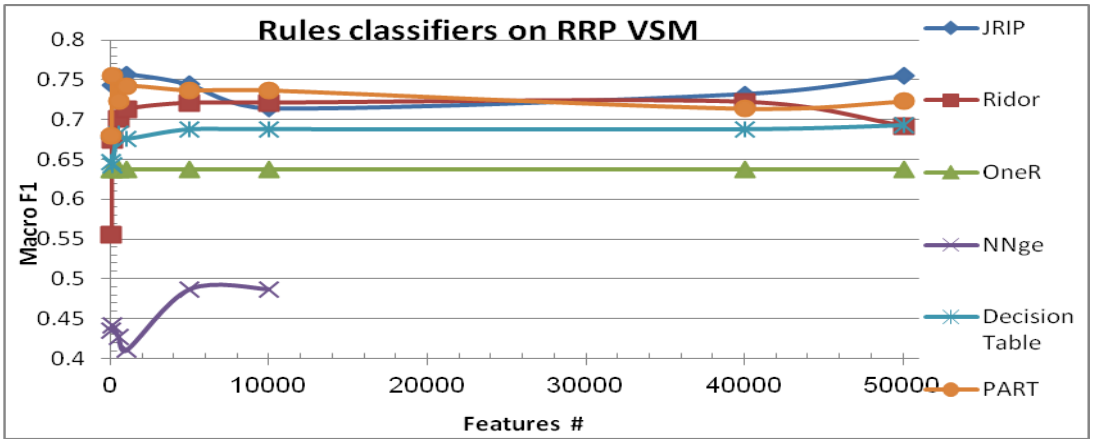
(e)



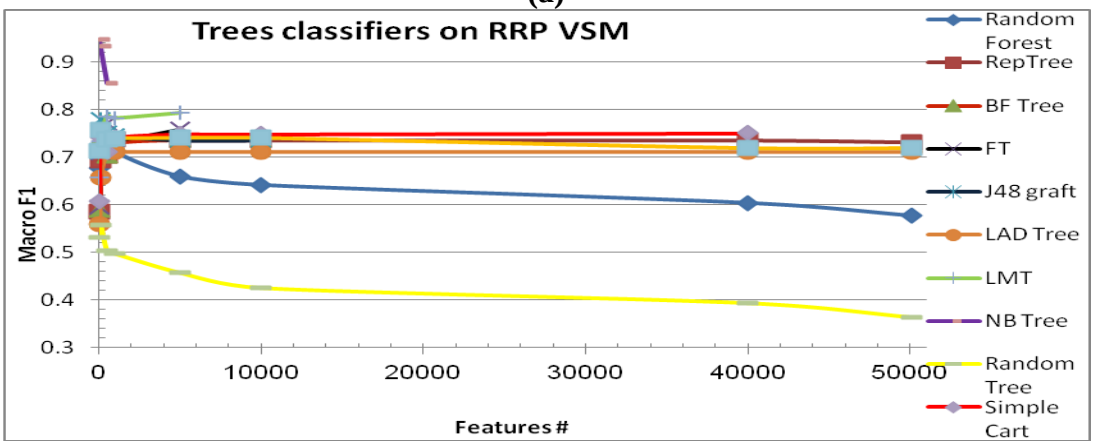
(f)

Figure 32: Comparison between classifiers' performance for Word VSM representation according to their type (a) rules, (b) trees, (c) functions, (d) Bayes-based, (e) miscellaneous, (f) meta

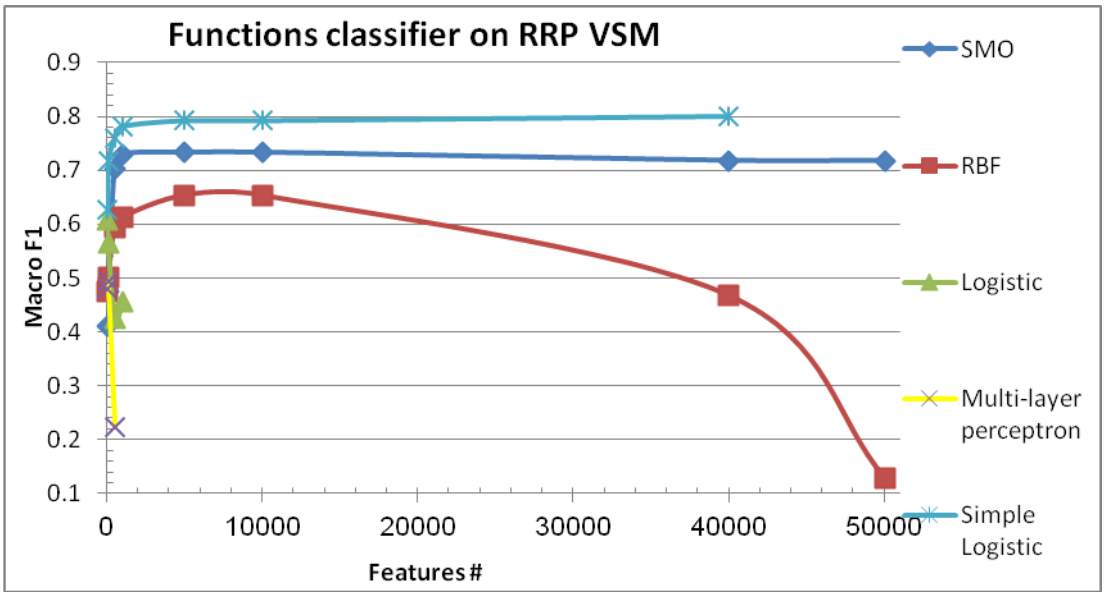
Figure 32 presents the performance of all classifiers for Stem representation. From Figure 32a, the three classifiers with best performance among rule-learners are JRip, Decision Table, and PART respectively. In Figure 32b, the four classifiers with best performance among tree learners are LMT, FT, Random Forest, and Simple Cart respectively. However, from Figure 32c, the best performance of function learners is for the two classifiers Simple Logistic and SMO respectively. In Figure 32d, the performance of Bayes-based learners is compared. The three classifiers with best performance are BN, NBM, and NBMU respectively. Miscellaneous learners are presented in Figure 32e and among the two learners, HP classifier performs better. In Figure 32f, Meta classifiers performances are presented and the best seven classifiers in performance are END, FC, LB, AdaBoost.M1, RSS, RF, and Decorate respectively.



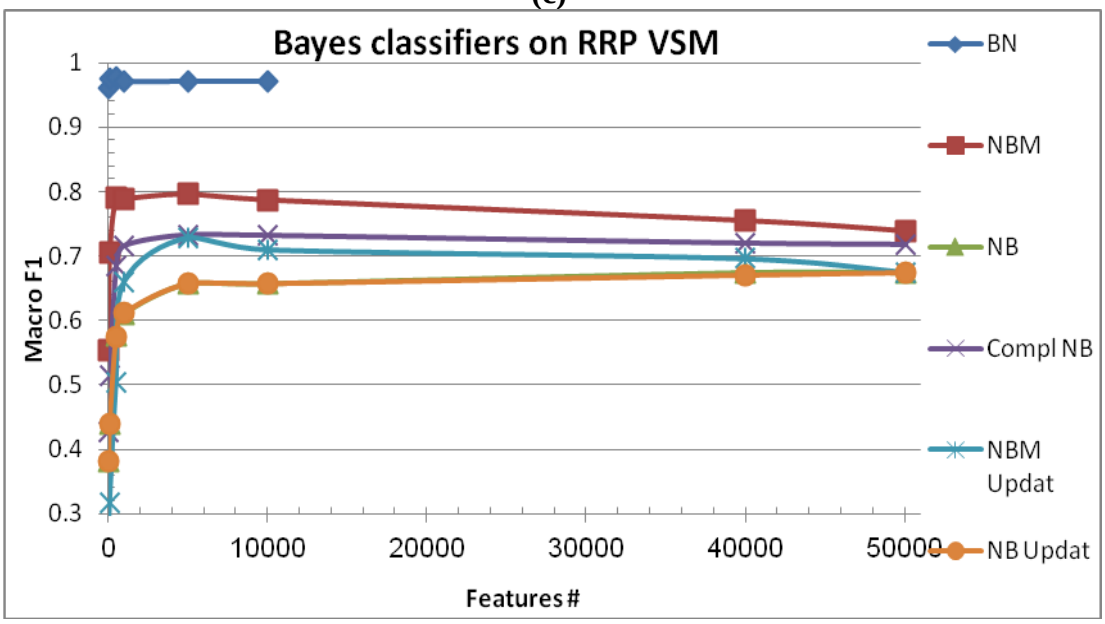
(a)



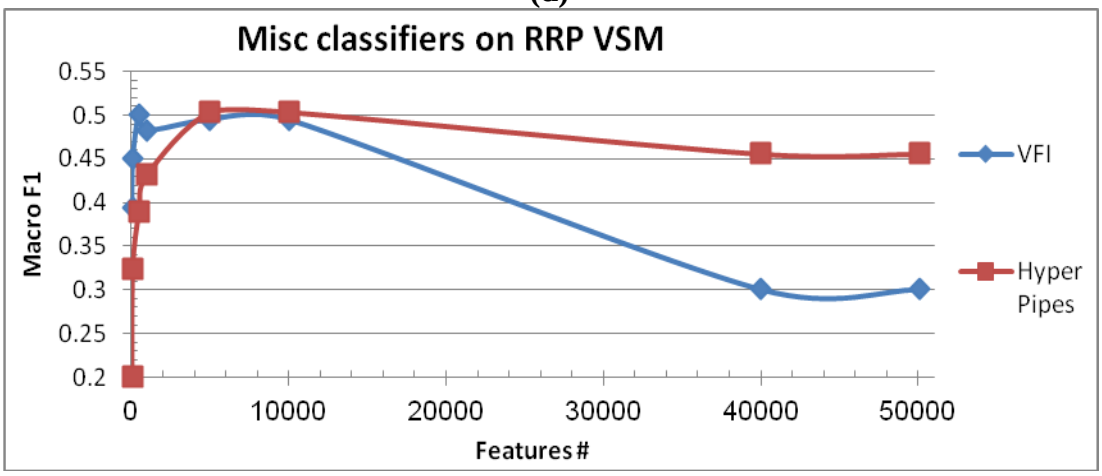
(b)


Functions classifier on RRP VSM
Macro F1
Features #
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic

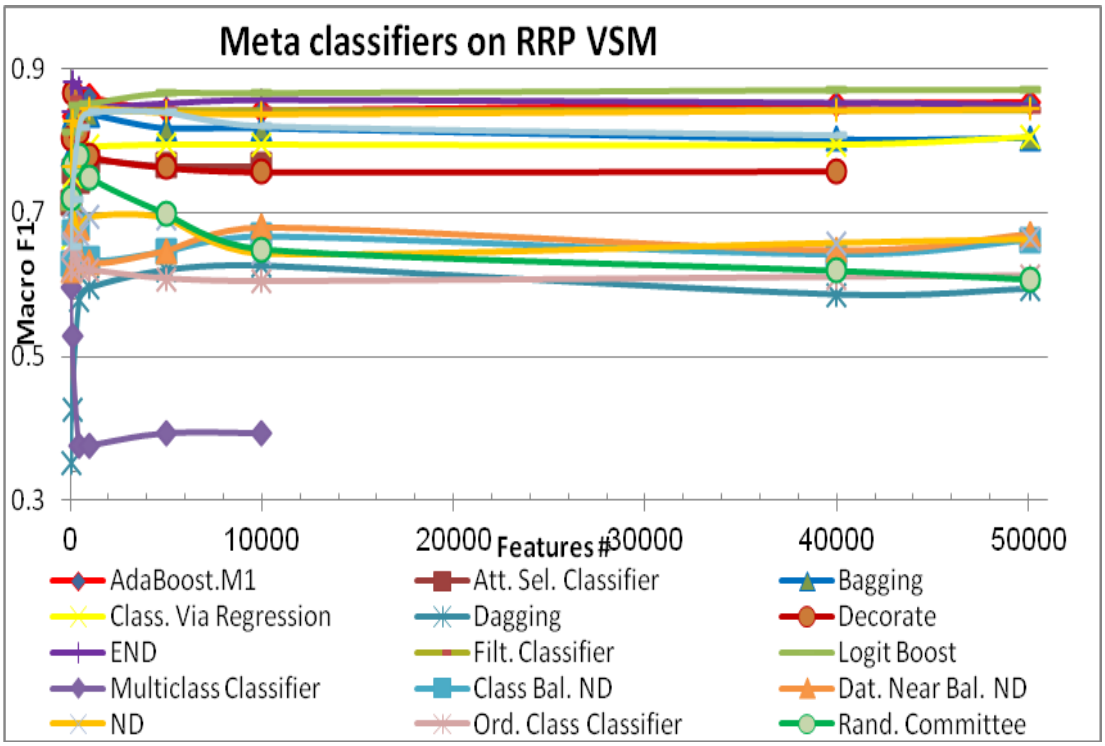
(c)


Bayes classifiers on RRP VSM
Macro F1
Features #
BN
NBM
NB
Compl NB
NBM Updat
NB Updat

(d)


Misc classifiers on RRP VSM
Macro F1
Features #
VFI
Hyper Pipes

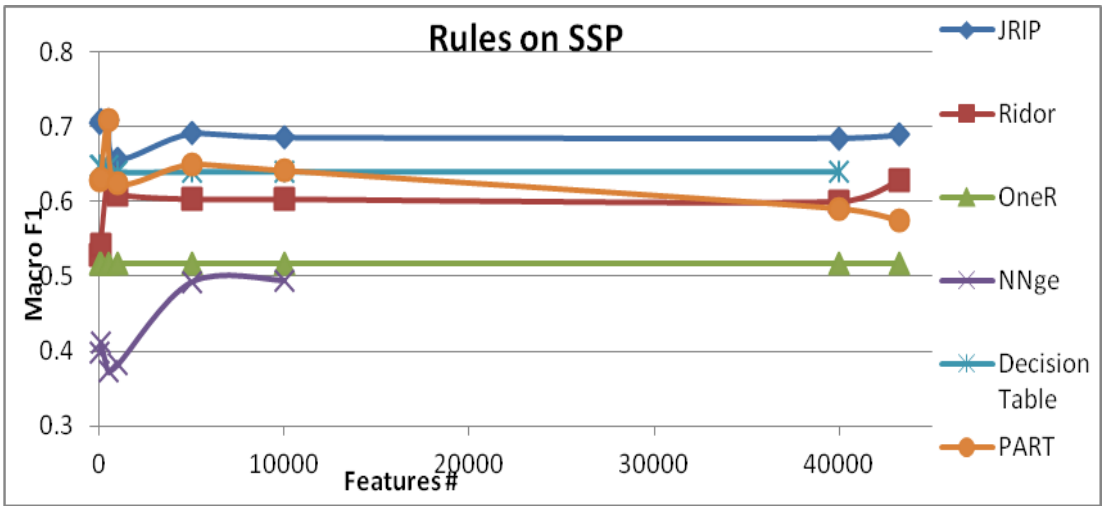
(e)



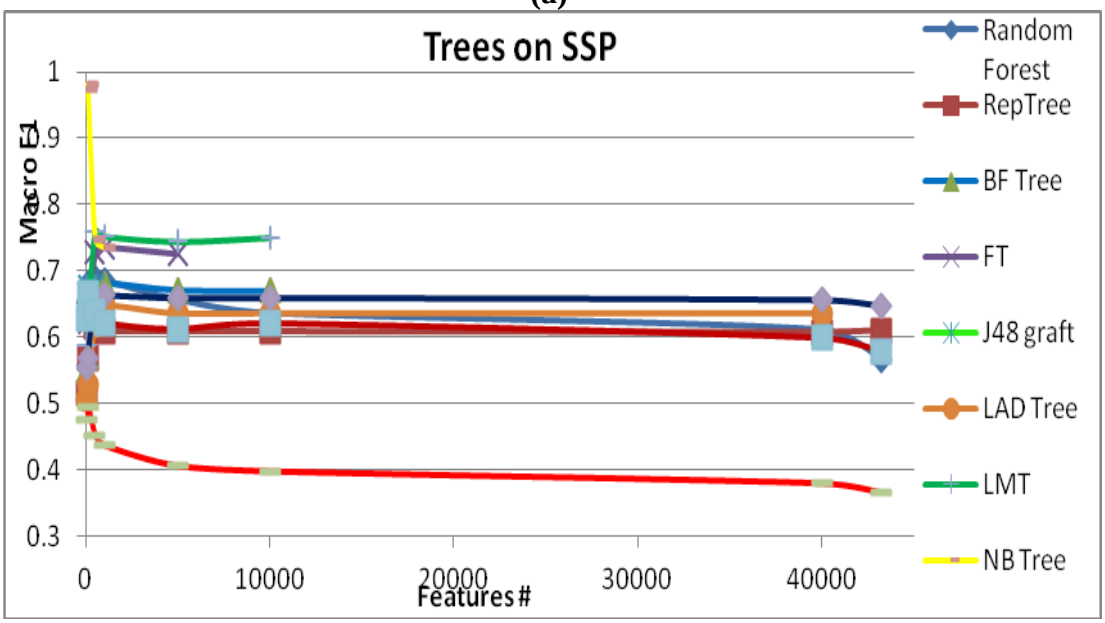
(f)

Figure 33: Comparison between classifiers' performance for RRP VSM representation according to their type (a) rules, (b) trees, (c) functions, (d) Bayes-based, (e) miscellaneous, (f) meta

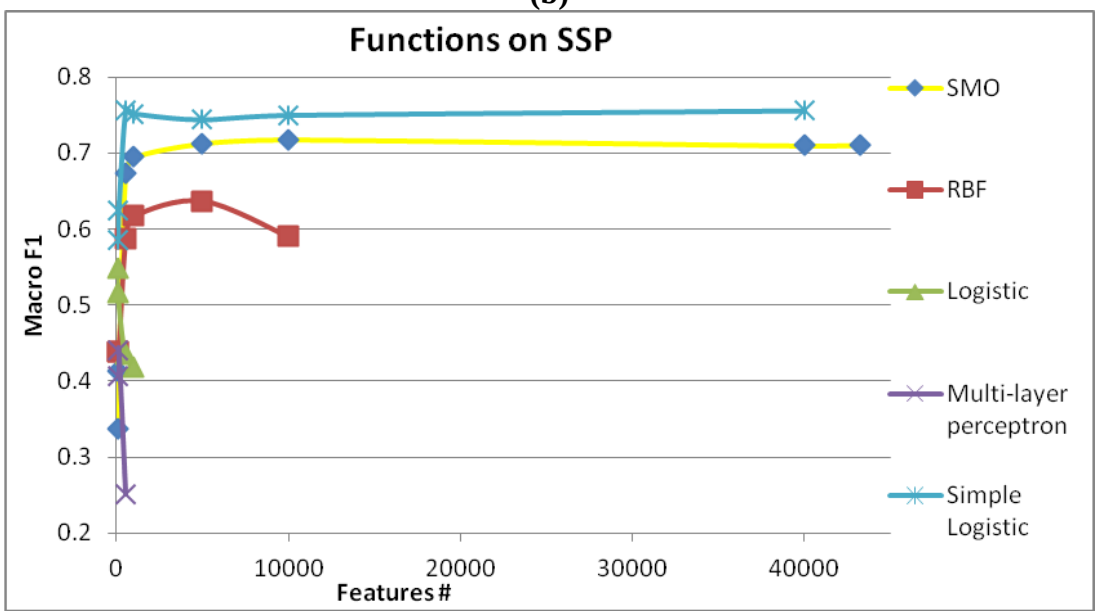
Figure 33 presents the performance of all classifiers for RRP representation. From Figure 33a, the three classifiers with best performance among rule-learners are PART, Ridor, and JRip respectively. In Figure 33b, the four classifiers with best performance among tree learners are LMT, FT, Simple Cart, and J48 respectively. However, from Figure 33c, the best performance of function learners is for the two classifiers Simple Logistic and SMO respectively. In Figure 33d, the performance of Bayes-based learners is compared. The three classifiers with best performance are BN, NBM, and Complement NB respectively. Miscellaneous learners are presented in Figure 33e and among the two learners, HP classifier performs better. In Figure 33f, Meta classifiers performances are presented and the best seven classifiers in performance are LB, END, AdaBoost.M1, FC, RSS, RF, and Bagging.


Rules on SSP
Macro F1
Features #
0.8
0.7
0.6
0.5
0.4
0.3
0
10000
20000
30000
40000
JRIP
Ridor
OneR
NNge
Decision Table
PART

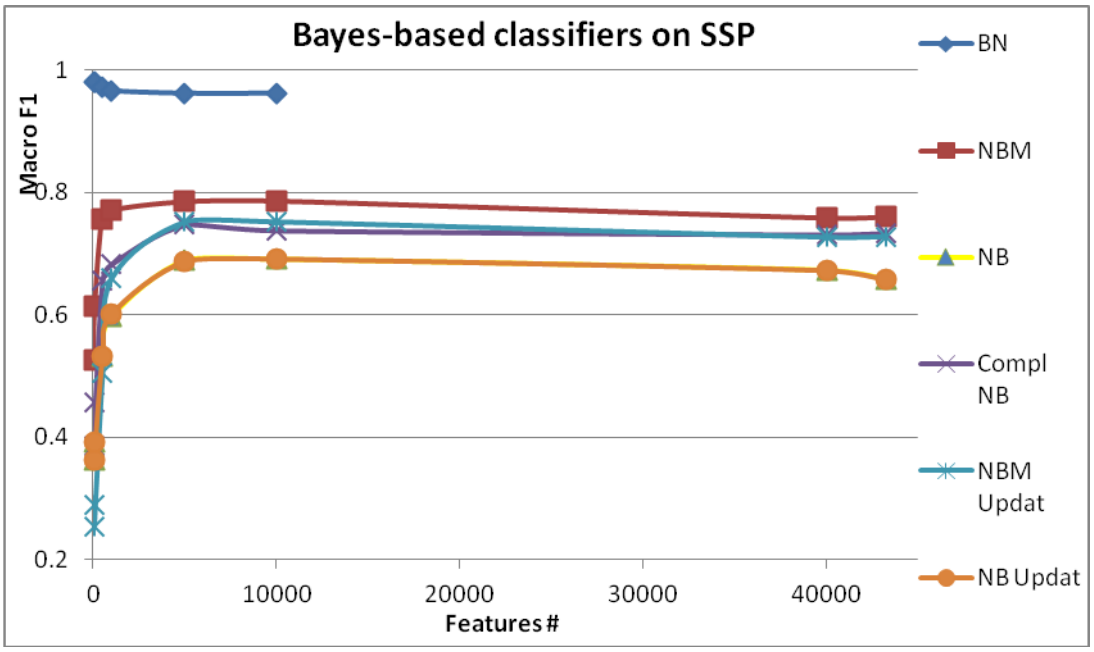
(a)


Trees on SSP
Macro F1
Features #
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10000
20000
30000
40000
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree

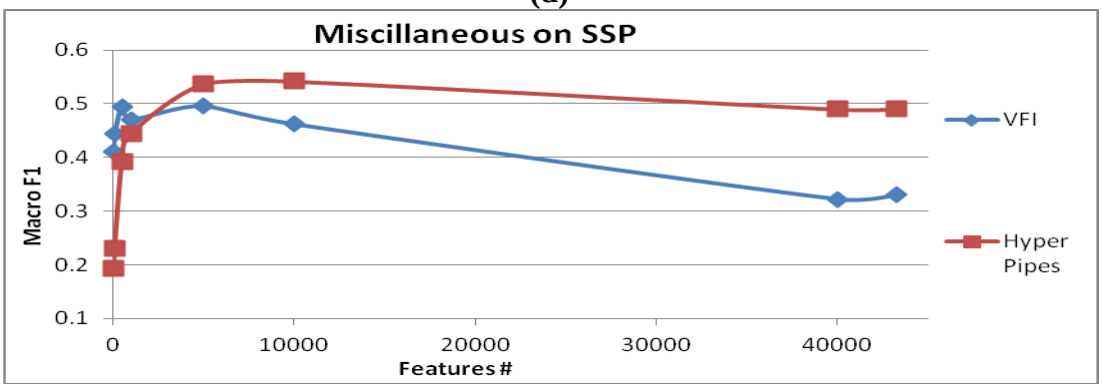
(b)


Functions on SSP
Macro F1
Features #
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10000
20000
30000
40000
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic

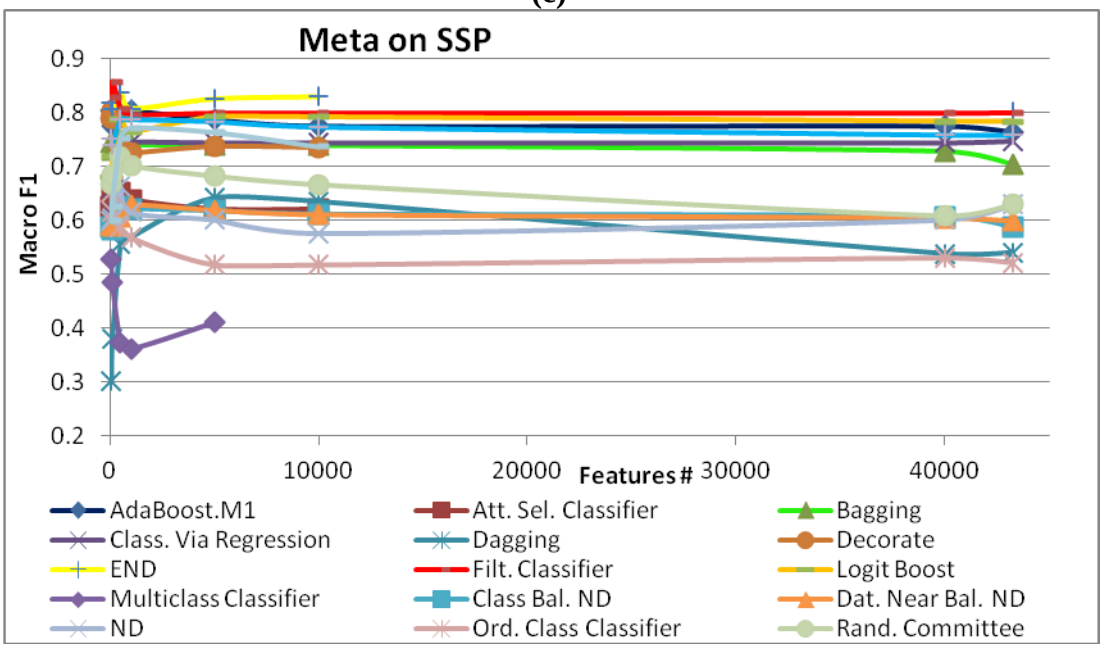
(c)



(d)



(e)

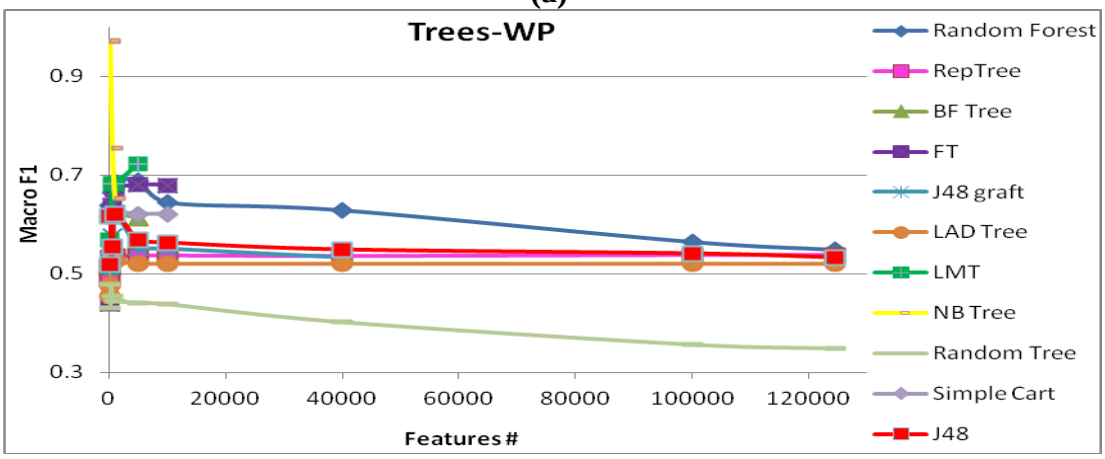


(f)

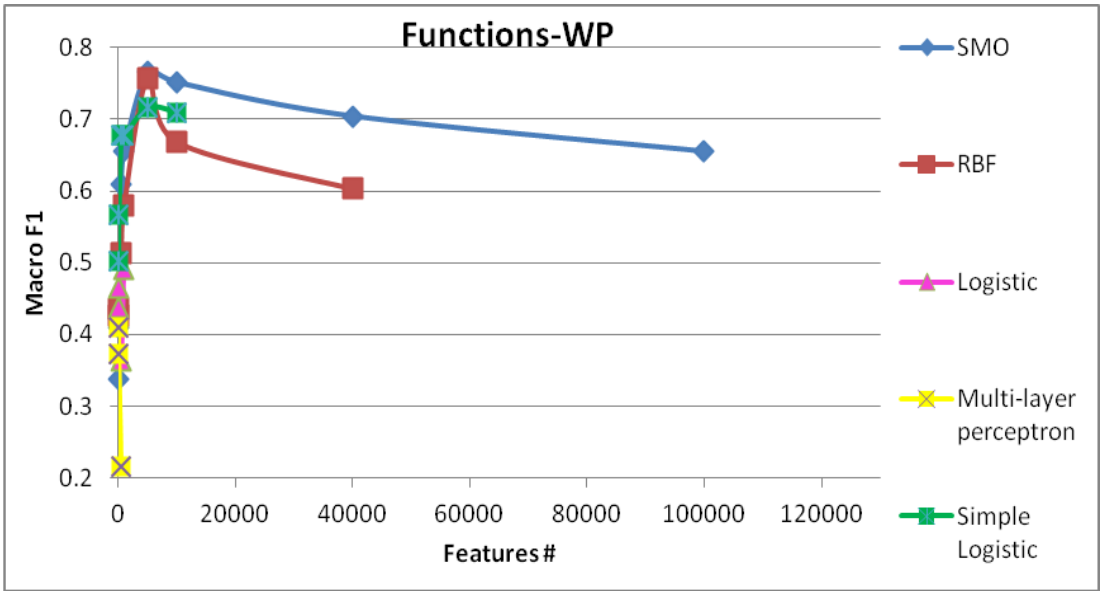
Figure 34: Comparison between classifiers' performance for SSP VSM representation according to their type (a) rules, (b) trees, (c) functions, (d) Bayes-based, (e) miscellaneous, (f) meta

Figure 34 presents the performance of all classifiers for SSP representation. From Figure 34a, the three classifiers with best performance among rule-learners are JRip, PART, and Decision Table respectively. In Figure 34b, the four classifiers with best performance among tree learners are LMT, FT, BF Tree, and Simple Cart respectively. However, from Figure 34c, the best performance of function learners is for the two classifiers Simple Logistic and SMO respectively. In Figure 34d, the performance of Bayes-based learners is compared. The three classifiers with best performance are BN, NBM, and NBMU respectively. Miscellaneous learners are presented in Figure 34e and among the two implemented learners, HP classifiers performs better. In Figure 34f, Meta classifiers performance are presented and the best seven classifiers in performance are END, FC, LB, RSS, AdaBoost.M1, RSS, RF, and CVR.

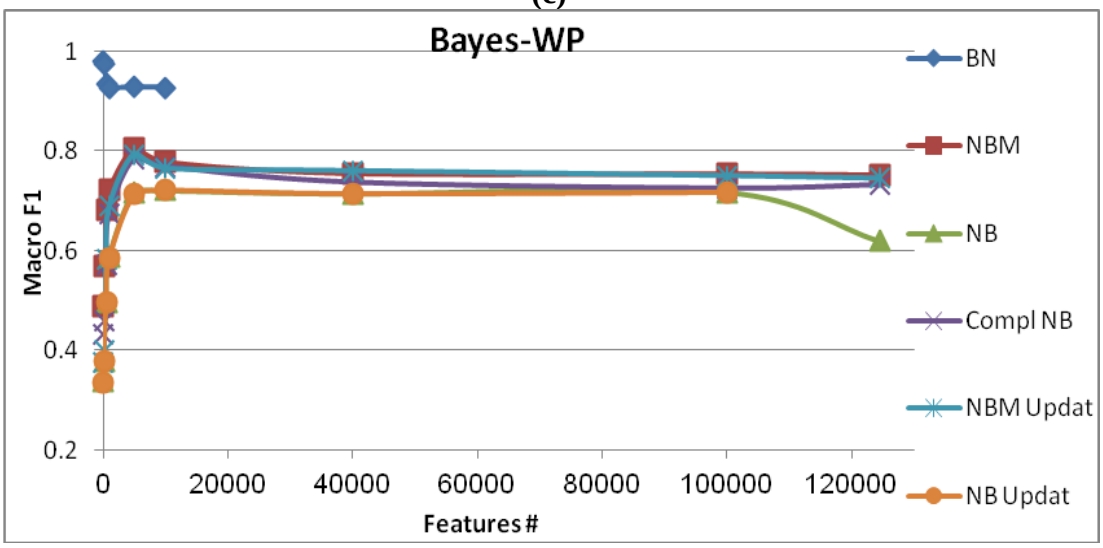
(a)



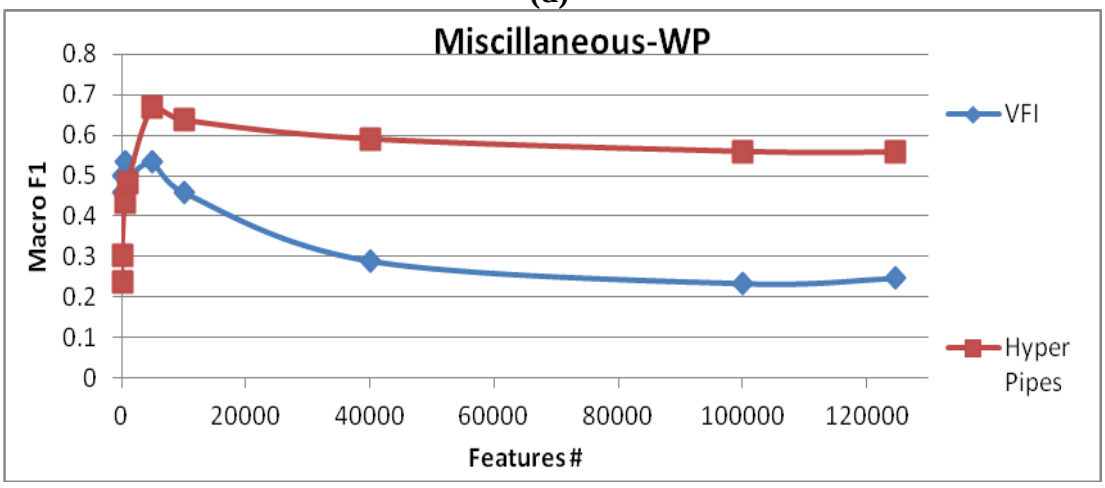
(b)


Functions-WP
Macro F1
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20000
40000
60000
80000
100000
120000
Features #
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic

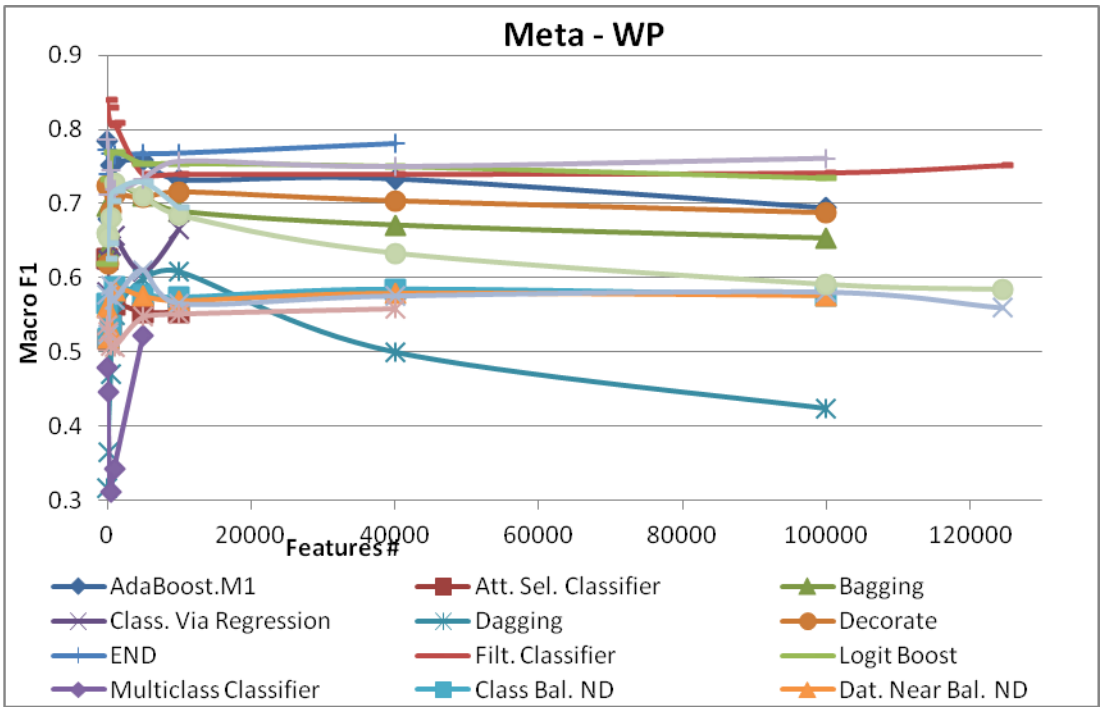
(c)


Bayes-WP
Macro F1
1
0.8
0.6
0.4
0.2
0
20000
40000
60000
80000
100000
120000
Features #
BN
NBM
NB
Compl NB
NBM Updat
NB Updat

(d)


Miscillaneous-WP
Macro F1
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20000
40000
60000
80000
100000
120000
Features #
VFI
Hyper Pipes

(e)



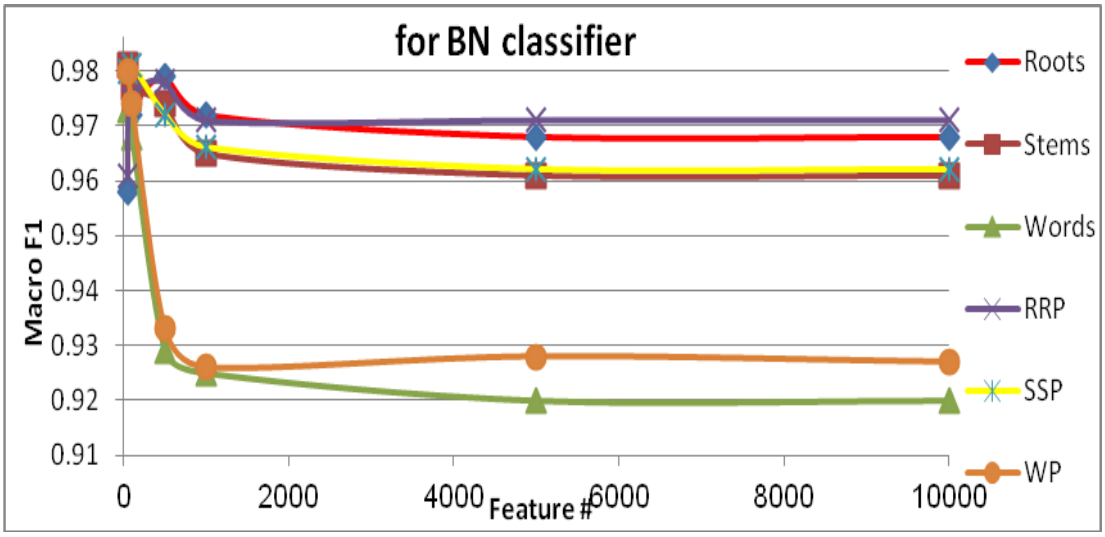
(f)

Figure 35: Comparison between classifiers' performance for WP VSM representation according to their type (a) rules, (b) trees, (c) functions, (d) Bayes-based, (e) miscellaneous, (f) meta

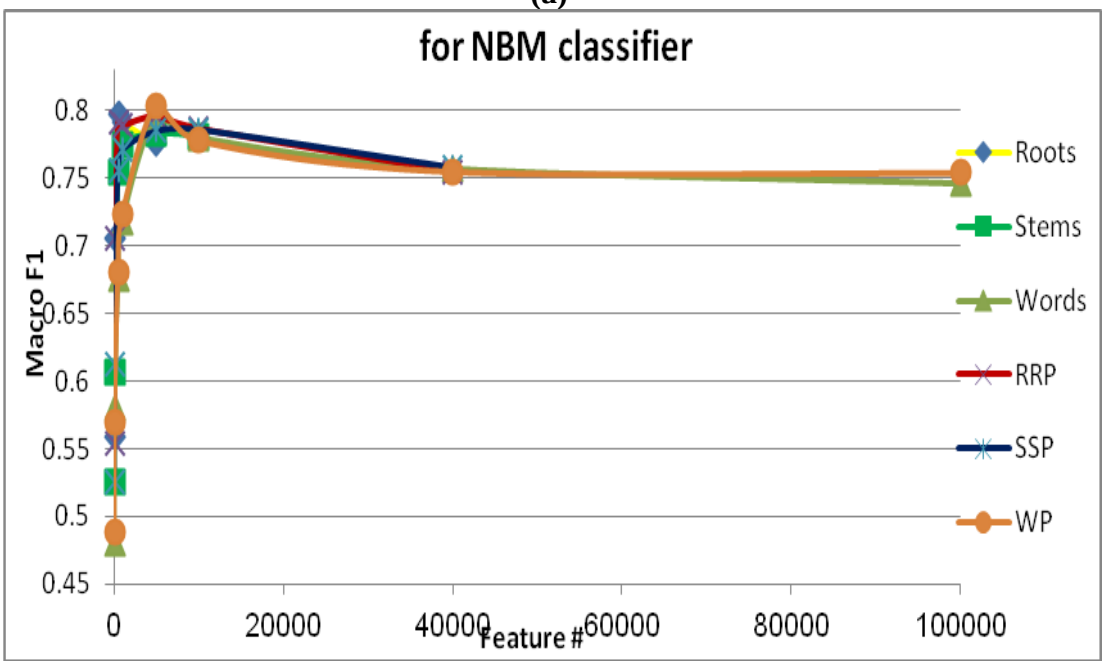
Figure 35 presents the performance of all classifiers for WP representation. From Figure 35a, the three classifiers with best performance among rule-learners are JRip, Ridor and Decision Table respectively. In Figure 35b, the four classifiers with best performance among tree learners are LMT, FT, Random Forest, and Simple Cart respectively. However, from Figure 35c, the best performance of function learners is for the two classifiers SMO and Simple Logistic respectively. In Figure 35d, the performance of Bayes-based learners is compared. The three classifiers with best performance are BN, NBM, and NBMU respectively. Miscellaneous learners are presented in Figure 35e and among the two implemented learners, HP classifiers performs better. In Figure 35f, Meta classifiers performances are presented and the best seven classifiers in performance are END, RSS, LB, FC, AdaBoost.M1, Decorate, and Bagging. It is clear from previous figures that the Bayes Net classifier has the highest weighted-F1 value among implemented classifiers for all VSM

representations with maximum weighted-$F1^M$ = 98.1%. The remaining implemented classifiers varied in their performances as the number of selected features varied for the used VSM representations.

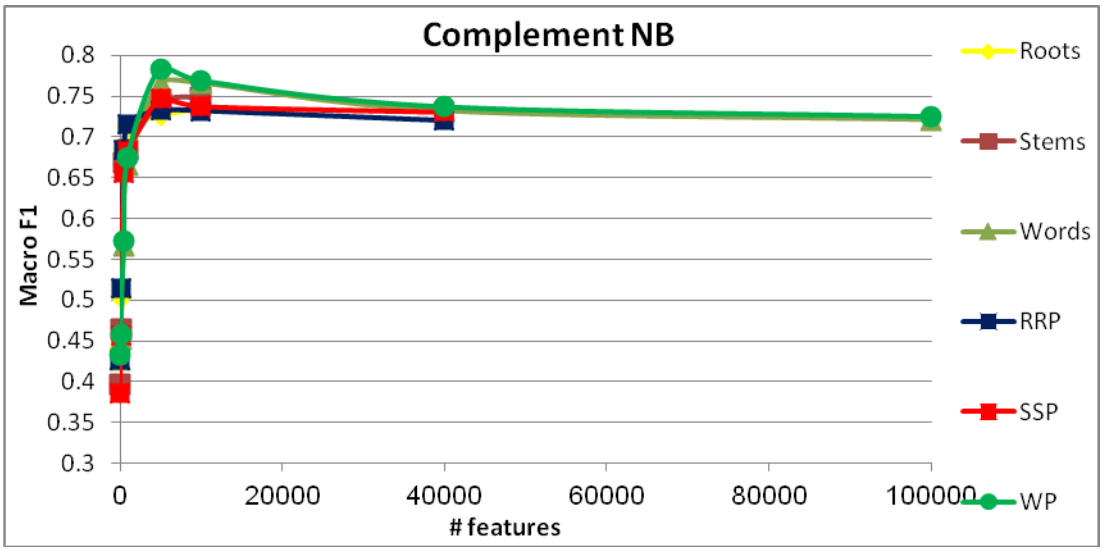
Figure 36 is presented in order to compare between the effect of implemented VSM representations for each classifier on TC performance (here only highest performing classifiers are presented and the rest of classifiers comparisons are in appendix IV).



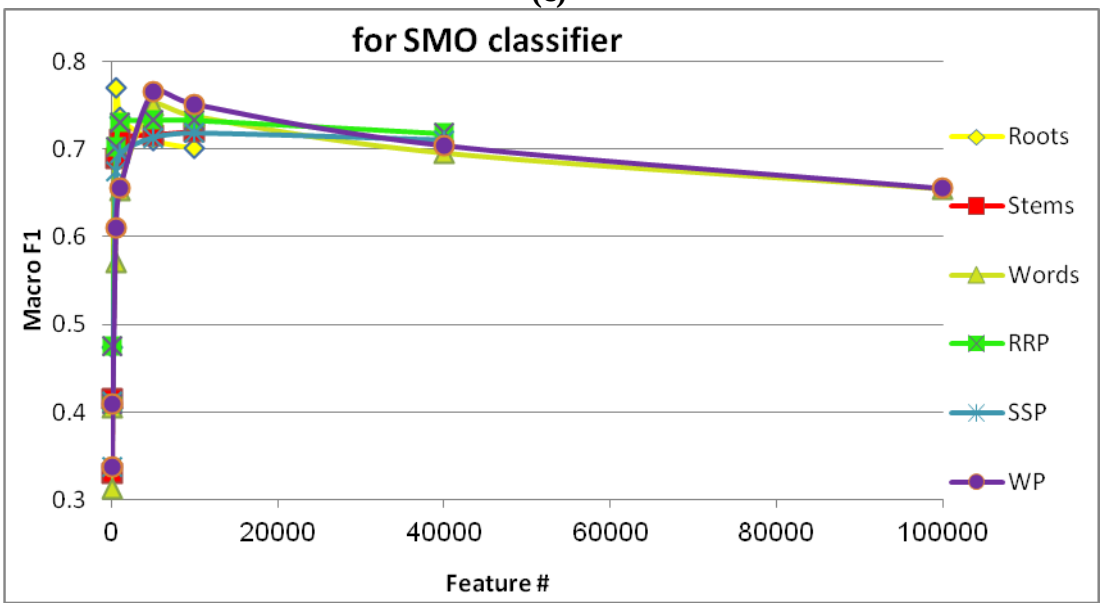
(a)



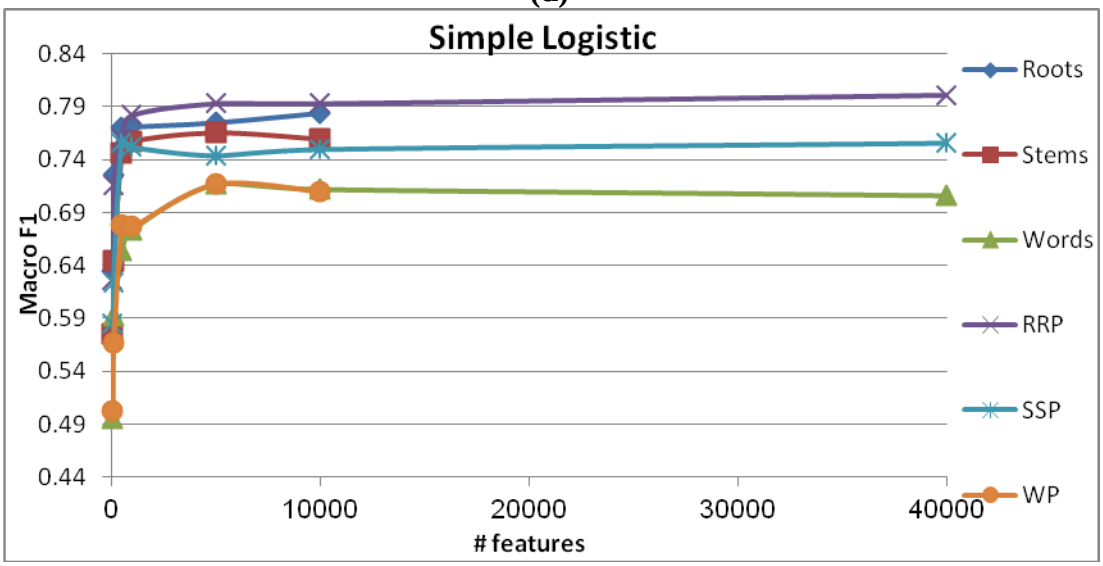
(b)


Complement NB
Macro F1
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

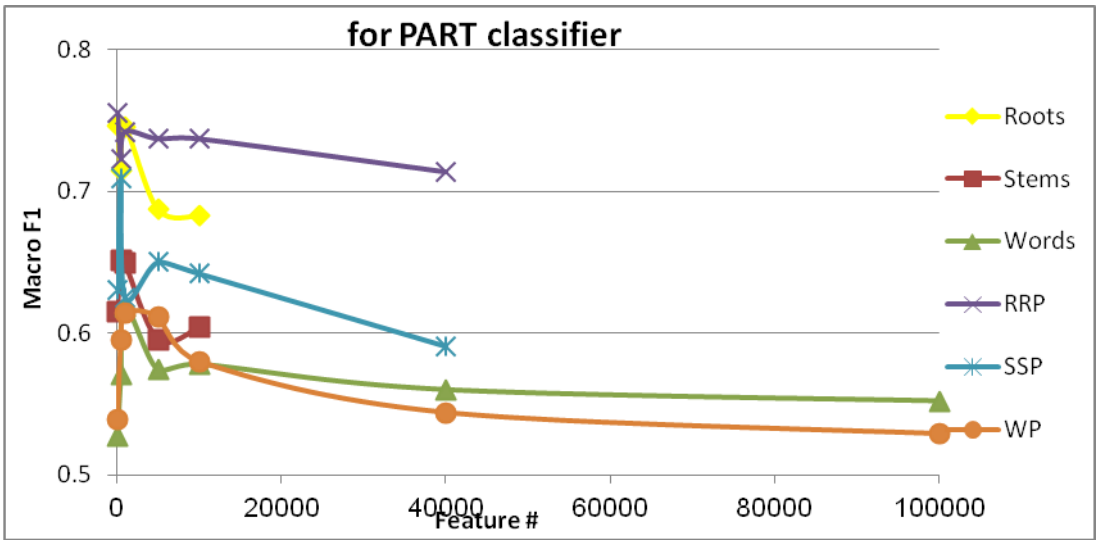
(c)


for SMO classifier
Macro F1
0.8
0.7
0.6
0.5
0.4
0.3
0
20000
40000
60000
80000
100000
Feature #
Roots
Stems
Words
RRP
SSP
WP

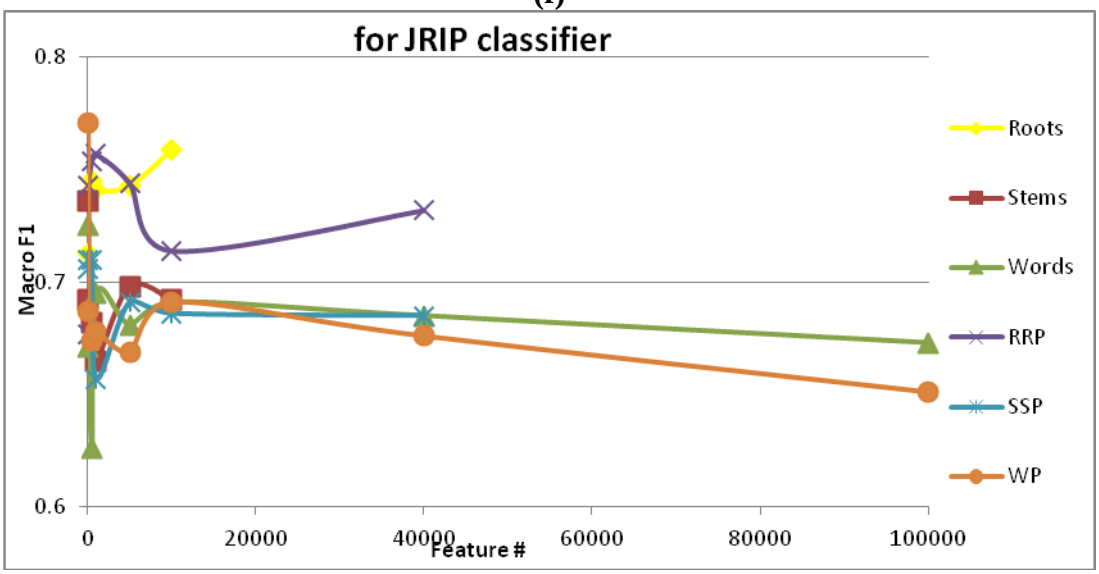
(d)


Simple Logistic
Macro F1
0.84
0.79
0.74
0.69
0.64
0.59
0.54
0.49
0.44
0
10000
20000
30000
40000
features
Roots
Stems
Words
RRP
SSP
WP

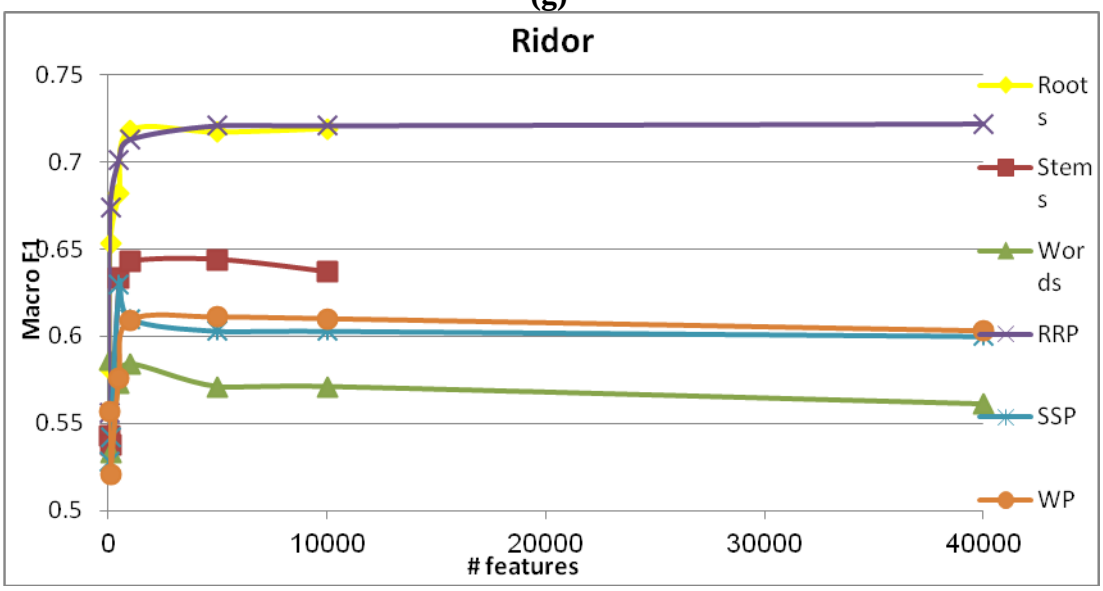
(e)


for PART classifier
0.8
0.7
0.6
0.5
Macro F1
0
20000
40000
60000
80000
100000
Feature #
Roots
Stems
Words
RRP
SSP
WP

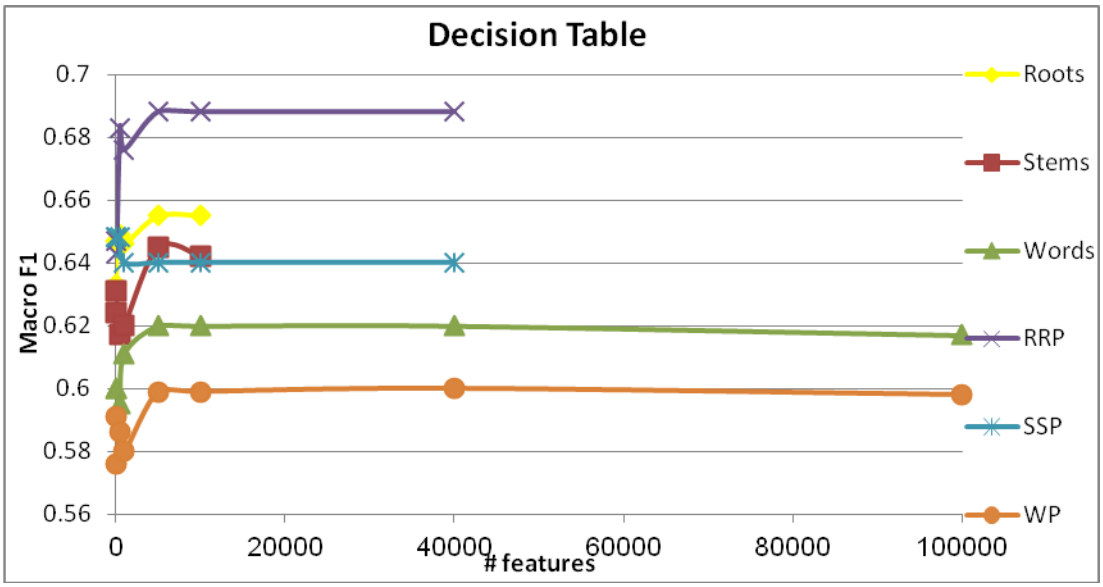
(f)


for JRIP classifier
0.8
0.7
0.6
Macro F1
0
20000
40000
60000
80000
100000
Feature #
Roots
Stems
Words
RRP
SSP
WP

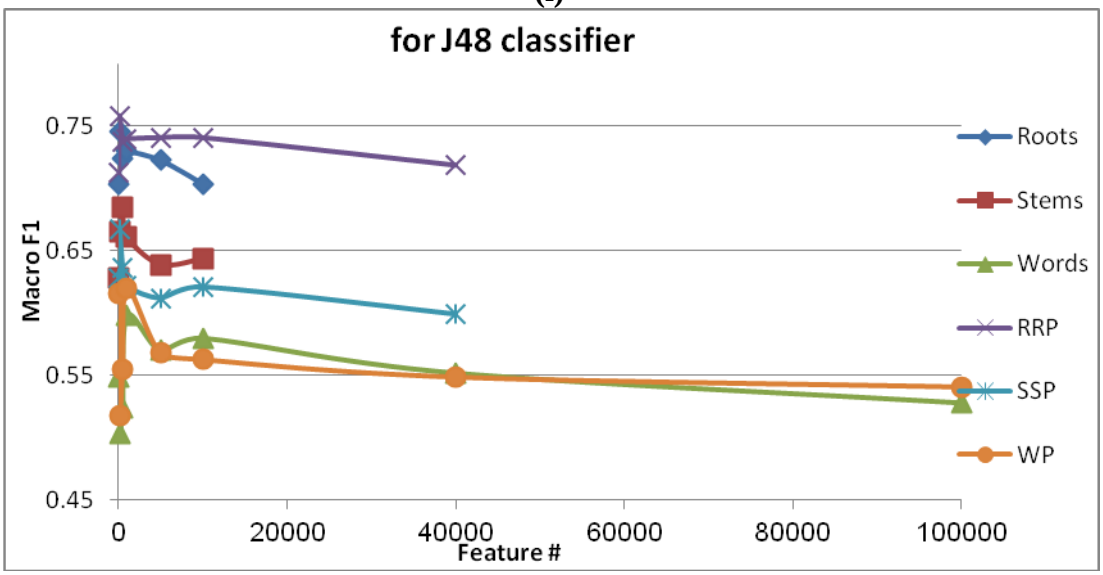
(g)


Ridor
0.75
0.7
0.65
0.6
0.55
0.5
Macro F1
0
10000
20000
30000
40000
features
Roots
Stems
Words
RRP
SSP
WP

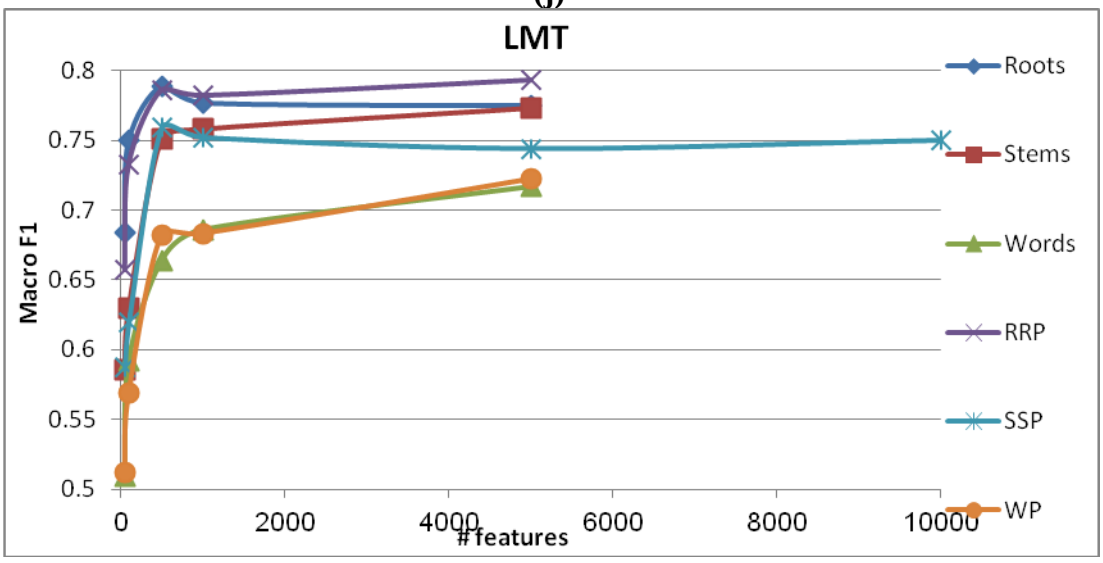
(h)


Decision Table
Macro F1
0.7
0.68
0.66
0.64
0.62
0.6
0.58
0.56
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

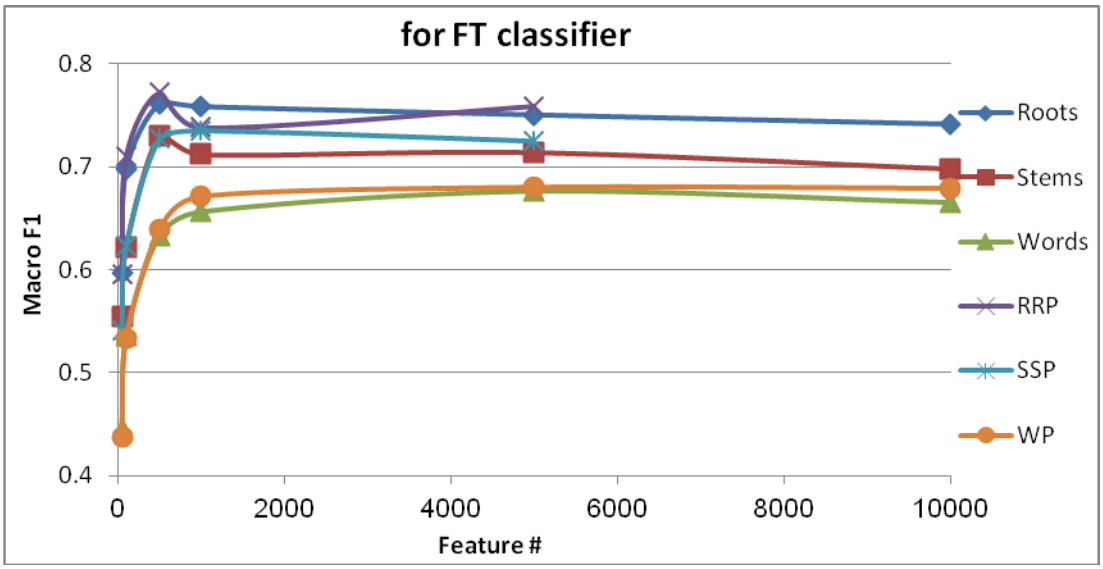
(i)


for J48 classifier
Macro F1
0.75
0.65
0.55
0.45
0
20000
40000
60000
80000
100000
Feature #
Roots
Stems
Words
RRP
SSP
WP

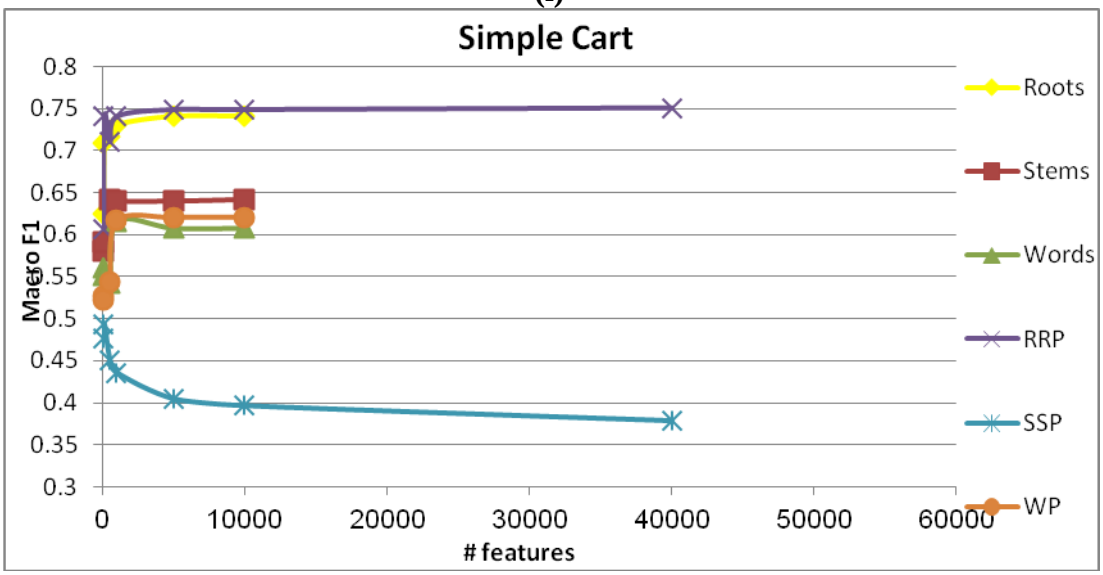
(j)


LMT
Macro F1
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
2000
4000
6000
8000
10000
features
Roots
Stems
Words
RRP
SSP
WP

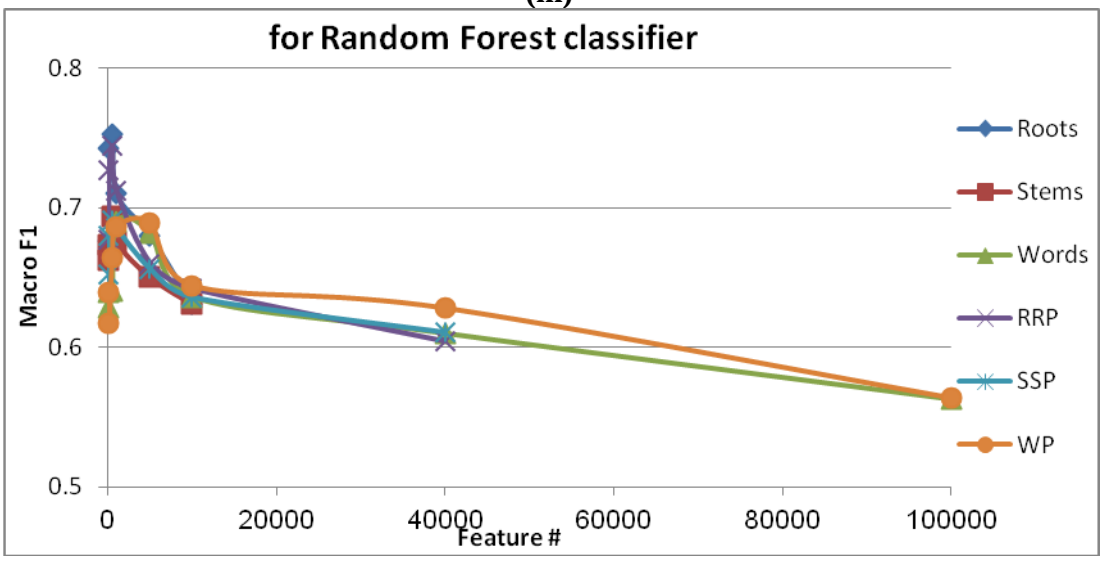
(k)


for FT classifier
Macro F1
Feature #
Roots
Stems
Words
RRP
SSP
WP

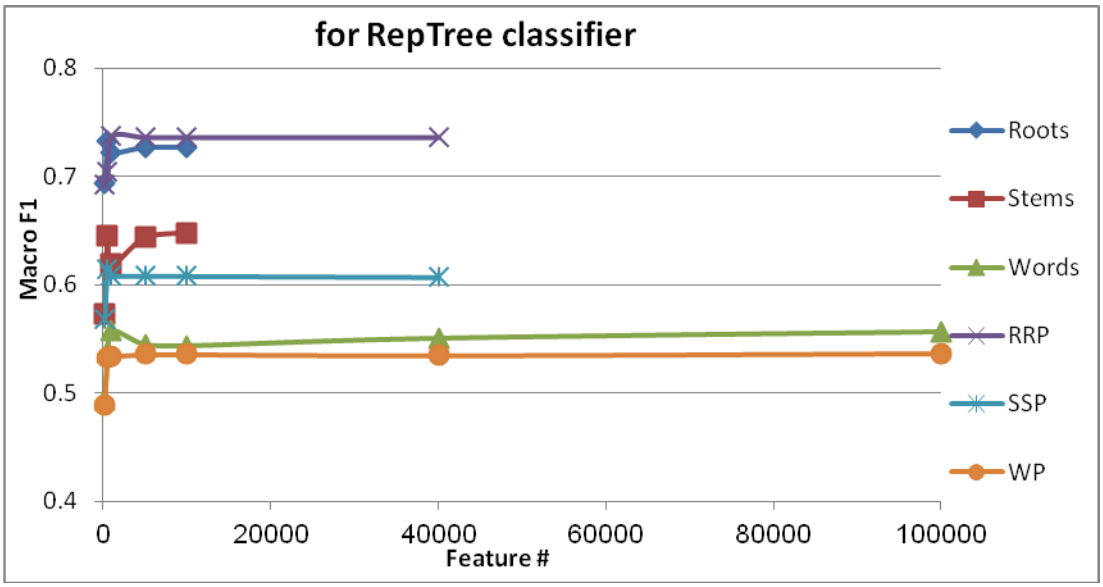
(l)


Simple Cart
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

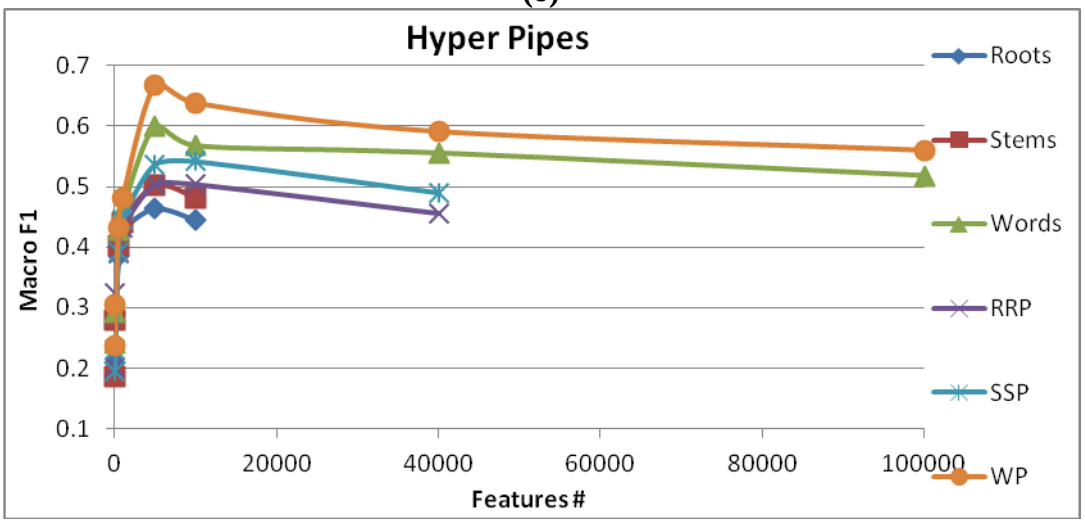
(m)


for Random Forest classifier
Macro F1
Feature #
Roots
Stems
Words
RRP
SSP
WP

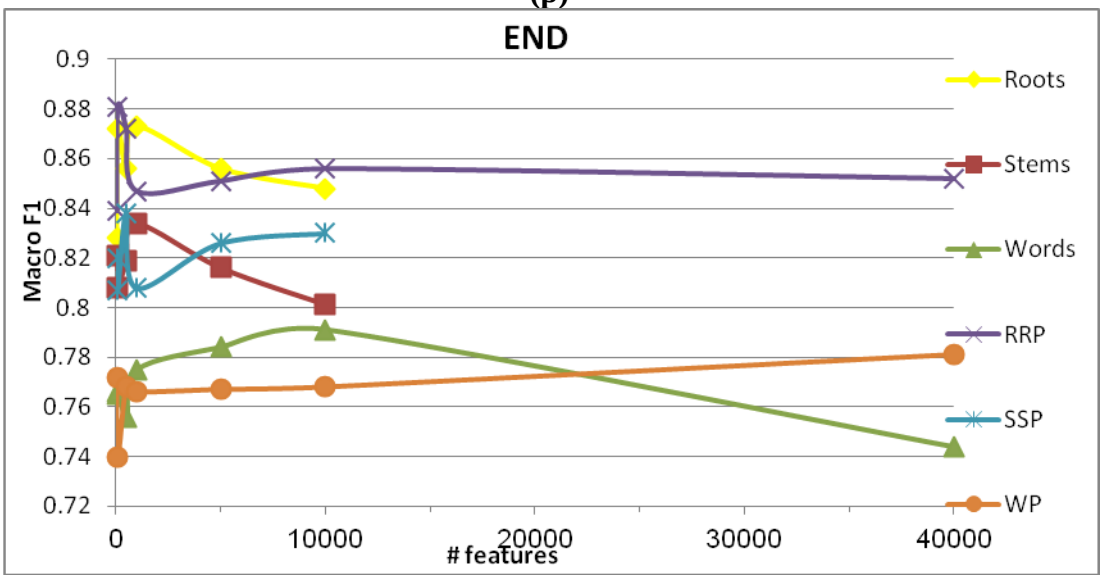
(n)


for RepTree classifier
Macro F1
0.8
0.7
0.6
0.5
0.4
0
20000
40000
60000
80000
100000
Feature #
Roots
Stems
Words
RRP
SSP
WP

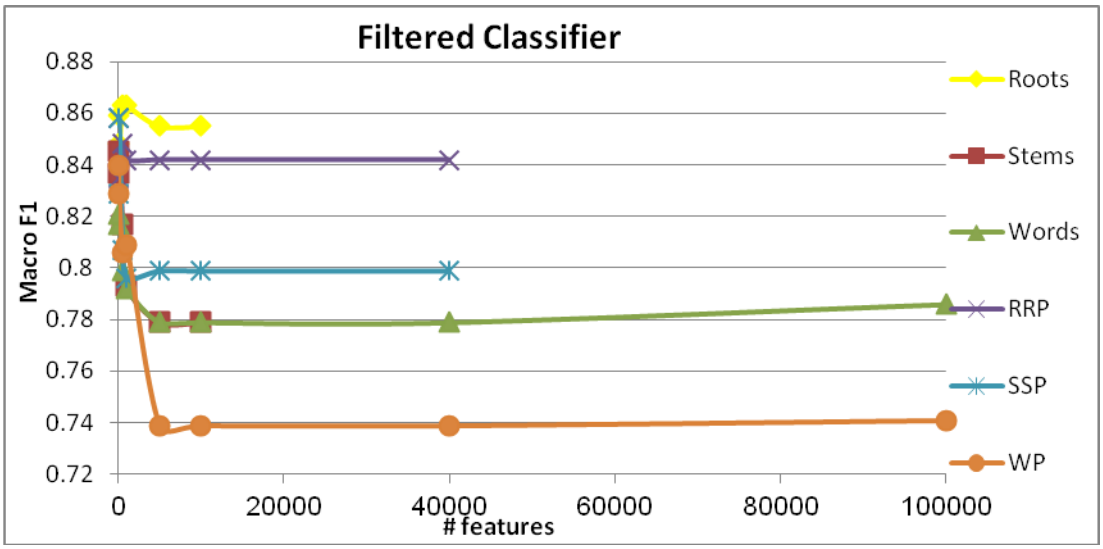
(o)


Hyper Pipes
Macro F1
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20000
40000
60000
80000
100000
Features #
Roots
Stems
Words
RRP
SSP
WP

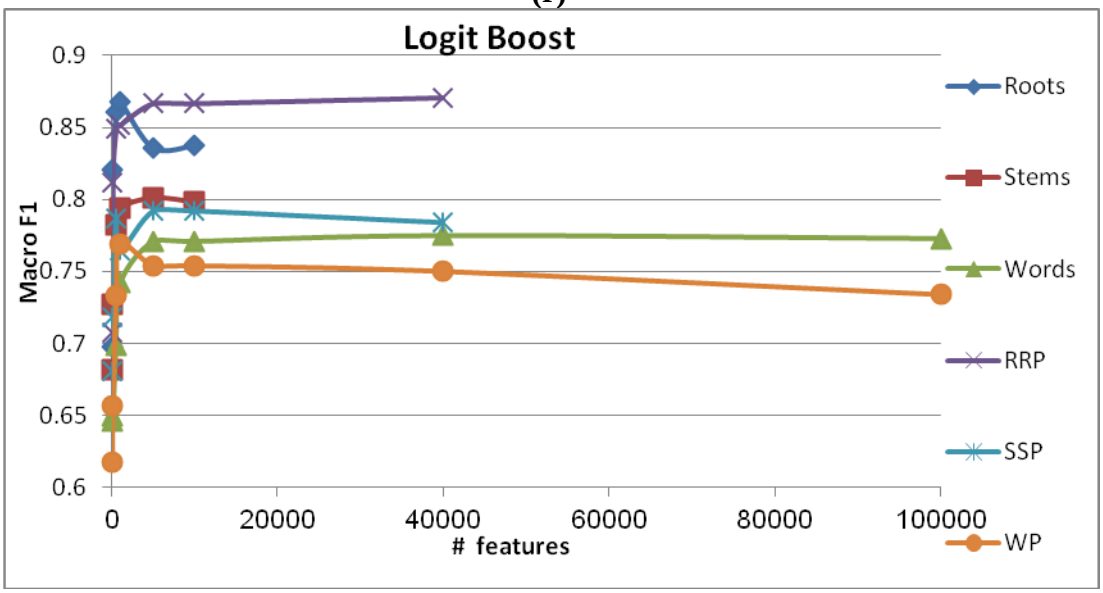
(p)


END
Macro F1
0.9
0.88
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0
10000
20000
30000
40000
features
Roots
Stems
Words
RRP
SSP
WP

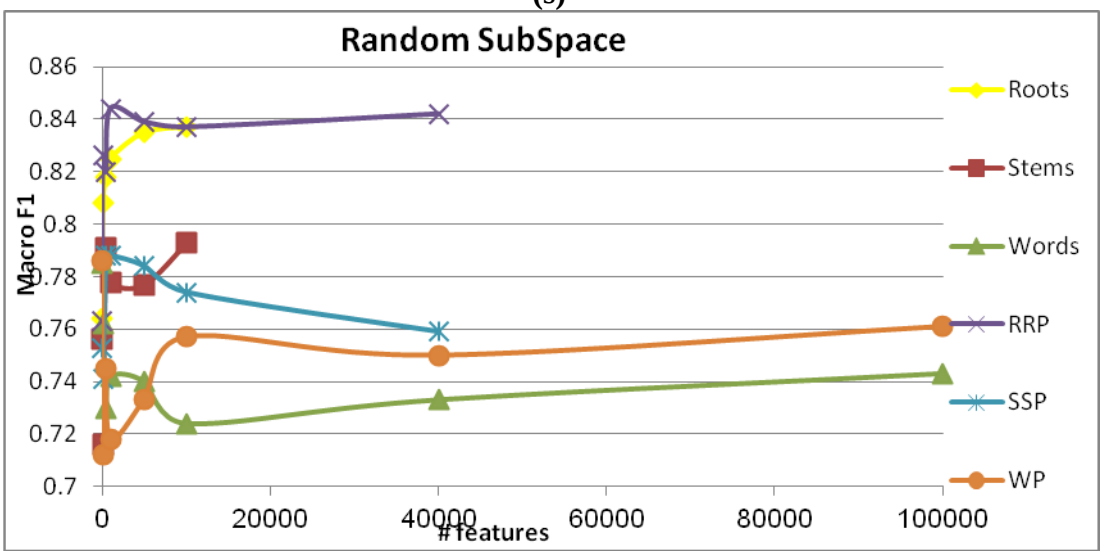
(q)


Filtered Classifier
Macro F1
0.88
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

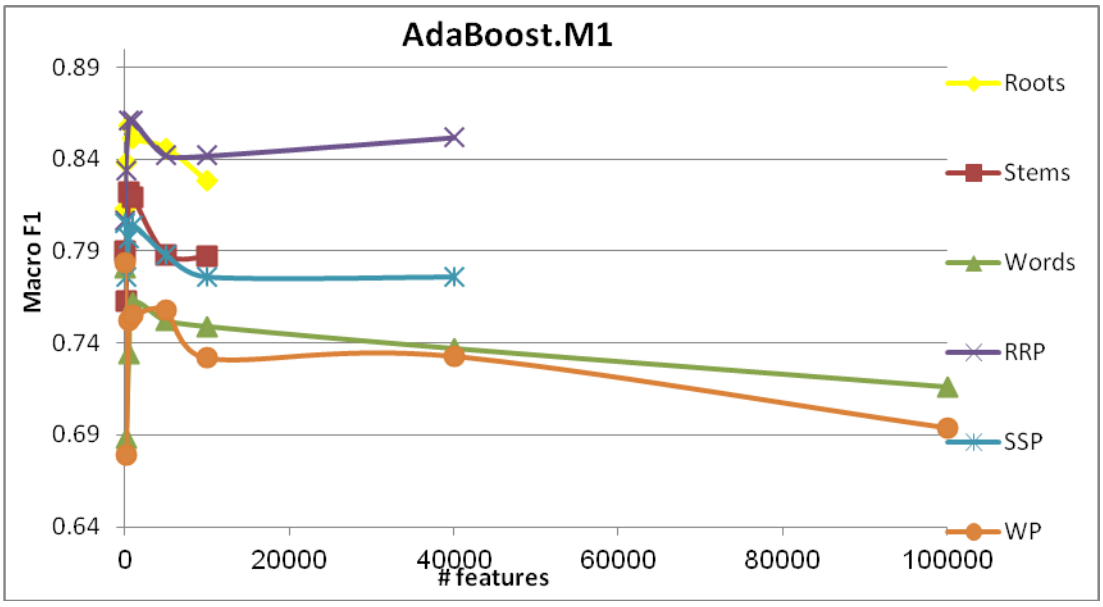
(r)


Logit Boost
Macro F1
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

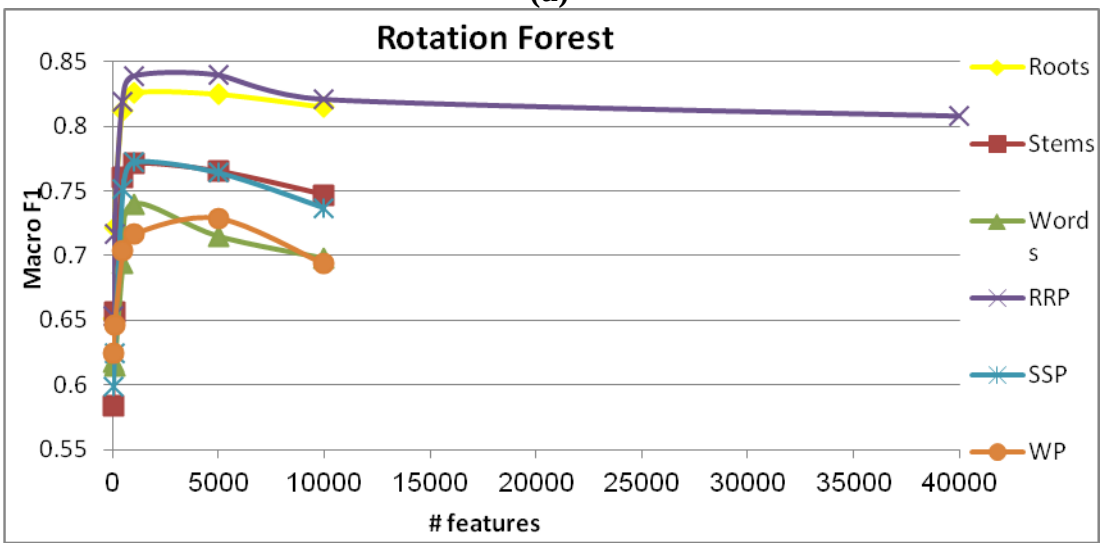
(s)


Random SubSpace
Macro F1
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0.7
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

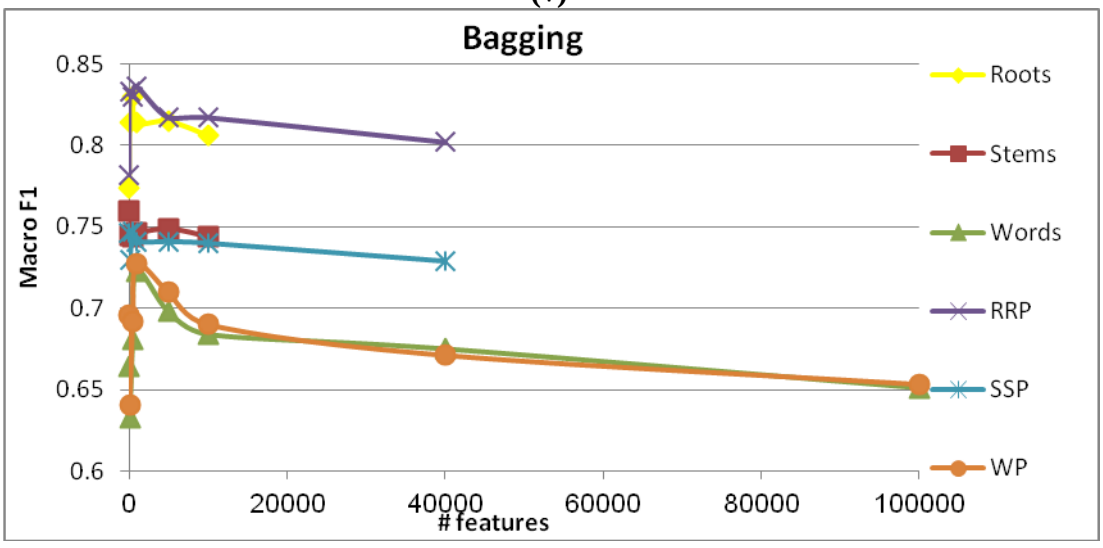
(t)


AdaBoost.M1
Macro F1
0.89
0.84
0.79
0.74
0.69
0.64
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

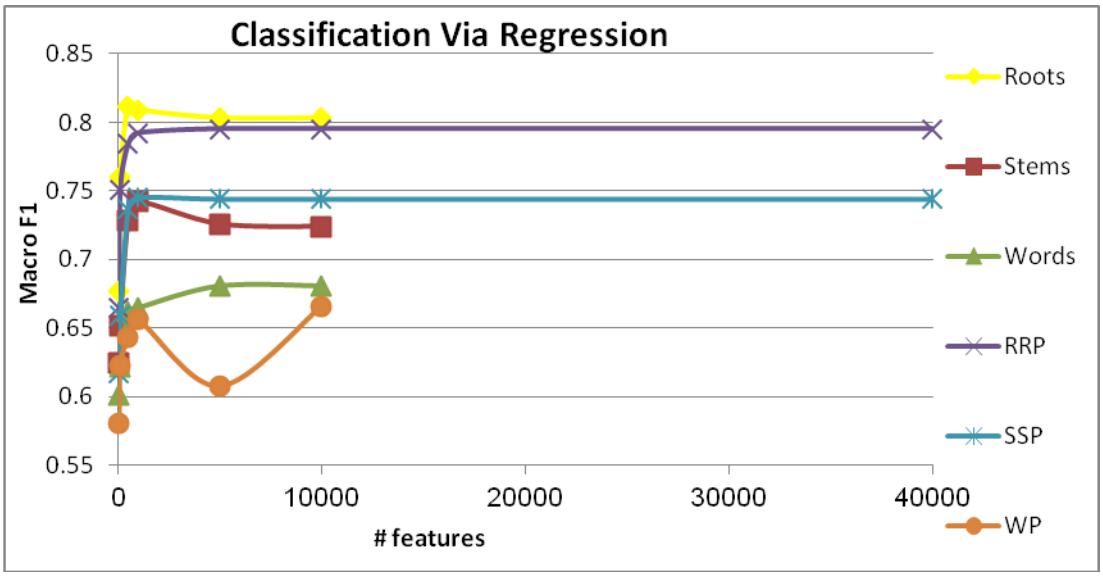
(u)


Rotation Forest
Macro F1
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
5000
10000
15000
20000
25000
30000
35000
40000
features
Roots
Stems
Word
s
RRP
SSP
WP

(v)


Bagging
Macro F1
0.85
0.8
0.75
0.7
0.65
0.6
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

(w)



(x)

Figure 36: Performance of different VSM representations as number of selected features varied using classifier (a) BN, (b) NBM, (c) Compl NB, (d) SMO, (e) Simple Logistic, (f) PART, (g) JRIP, (h) Ridor, (i) Decision Table, (j) J48, (k) LMT, (l) FT, (m) Simple Cart, (n) Random Forest, (o) Rep Tree, (p) Hyper Pipes, (q) END, (r) Filtered Classifier, (s) Logit Boost, (t) Random SubSpace, (u) AdaBoost.M1, (v) Rotation Forest, (w) Bagging, (x) Classification Via Regression

From Figure 36, BN performance is affected very slightly with choosing different VSM representations. Such small variation indicates that representing terms by roots or stems of respective words and including phrases in such representations have slight effect on its performance. However, the performance of this classifier is highest for Roots representation. Also, as the number of selected features varies the performance of this classifier reduces by about 5% when using Words representation while for Roots and Stems representations the reduction is about 2%. Other classifiers in Figure 36 performed in a different way than the BN classifier.

In order to establish at which number of features each classifier has highest performance, Table 31 is presented. From this table, 44 classifiers' performances are highest for Root and RRP representations especially for features numbers in the range [50, 5000]. It is noticed that most classifiers have highest performance at 1000 or 5000 features. In general, the addition of phrases to original VSM representation

improved/degraded TC performance slightly especially as feature numbers are above 5000.

Classifier Name	Information	VSM representation					
		Roots	Stems	Words	RRP	SSP	WP
JRip	# features	10,000	5,000	1,000	1,000	500	10,000
	$F1^M$	0.759	0.698	0.695	0.757	0.71	0.619
PART	# features	1,000	500	1,000	100	500	1,000
	$F1^M$	0.745	0.651	0.619	0.755	0.709	0.614
Decision Table	# features	5000	5000	5000	5000	50	40000
	$F1^M$	0.655	0.645	0.62	0.688	0.648	0.6
Ridor	# features	10000	5000	1000	5000	500	5000
	$F1^M$	0.719	0.644	0.584	0.721	0.63	0.611
RepTree	# features	500	10,000	1,000	1,000	500	5,000
	$F1^M$	0.733	0.648	0.558	0.737	0.615	0.536
J48	# features	100	500	1,000	100	100	1,000
	$F1^M$	0.746	0.685	0.598	0.758	0.667	0.62
LMT	# features	500	5000	5000	5000	500	5000
	$F1^M$	0.789	0.773	0.717	0.793	0.759	0.722
FT	# features	1000	500	5000	500	1000	5000
	$F1^M$	0.758	0.73	0.676	0.772	0.735	0.68
Simple Cart	# features	5000	500	1000	40000	100	5000
	$F1^M$	0.741	0.642	0.616	0.751	0.494	0.621
Random Forest	# features	500	500	1000	500	500	5000
	$F1^M$	0.753	0.694	0.69	0.744	0.691	0.689
Simple Logistic	# features	10000	5000	5000	5000	500	5000
	$F1^M$	0.784	0.764	0.717	0.793	0.756	0.717
SMO	# features	500	5,000	5,000	5,000	10,000	5,000
	$F1^M$	0.77	0.716	0.75	0.733	0.718	0.766
Hyper Pipes	# features	5000	5000	5000	5000	1000	5000
	$F1^M$	0.464	0.502	0.599	0.504	0.542	0.668
END	# features	1000	1000	10000	500	500	40000
	$F1^M$	0.873	0.834	0.791	0.872	0.838	0.781
Filter Classifier	# features	500	100	500	50	100	50
	$F1^M$	0.863	0.845	0.799	0.858	0.858	0.84
Logit Boost	# features	1000	5000	40000	40000	5000	1000
	$F1^M$	0.868	0.801	0.775	0.871	0.792	0.769
Rand SubSpace	# features	10000	10000	50	1000	500	50
	$F1^M$	0.837	0.793	0.785	0.844	0.788	0.786
AdaBoost.M1	# features	500	500	50	500	1000	50
	$F1^M$	0.858	0.822	0.781	0.861	0.804	0.784
Rotation Forest	# features	1000	1000	1000	5000	1000	5000
	$F1^M$	0.826	0.771	0.74	0.84	0.772	0.729
Bagging	# features	500	5000	5000	1000	500	1000
	$F1^M$	0.83	0.749	0.622	0.836	0.747	0.727
CVR	# features	500	1000	5000	5000	1000	10000
	$F1^M$	0.812	0.742	0.681	0.795	0.745	0.666
BN	# features	500	50	50	500	100	50
	$F1^M$	0.979	0.981	0.973	0.978	0.981	0.98
Complement NB	# features	10000	10000	5000	5000	5000	5000
	$F1^M$	0.735	0.749	0.766	0.733	0.747	0.784
NBM	# features	500	5,000	5,000	5,000	10,000	5,000
	$F1^M$	0.798	0.783	0.795	0.796	0.678	0.804

Table 31: Maximum $F1^M$ values at specific features number for implemented VSM representations along each classifier.

Table 31 illustrates that maximum weighted-$F1^M$ values are rather similar among most classifiers (remaining classifiers' performance comparisons are presented in

appendix IV). Also, since the maximum weighted-$F1^M$ values are achieved for different number of features among different VSM representations and classifiers. Thus, Table 32 was formed to present the amount of improvement/degradation of each representation using best performing classifiers (remaining classifiers' performance comparisons are presented in appendix IV).

Classifier	Max w-$F1^M$, VSM type	Improvement/degradation of first compared to second VSM type (%)			Improvement/degradation of second compared to first VSM type (%)			
		Roots, RRP	Stems, SSP	Words, WP	Roots, Stems	Roots, Words	RRP, SSP	RRP, WP
J48	0.758, RRP	+1.2	-1.8	+2.2	+6.1	+14.8	+9.1	+13.8
SMO	0.77, Roots	-6.7	+0.2	+1.6	+5.4	+2	+1.5	-3.3
BN	0.981, Stems	-0.1	0	+0.2	+0.2	-0.6	+0.3	+0.2
NBM	0.804, WP	-0.2	-10.5	+0.9	+1.5	+0.3	+11.8	-0.8
RepTree	0.737, RRP	+0.4	-3.3	-2.2	+8.5	+17.5	+12.2	+20.1
JRip	0.759, Roots	-0.2	+1.2	-0.4	+6.1	+6.4	+4.7	+6.6
PART	0.755, RRP	+1.0	+5.8	-0.5	+9.4	+12.6	+4.6	+14.1
Decision Table	0.688, RRP	+3.3	+0.3	-2	+1	+3.5	+4	+8.8
Ridor	0.721, RRP	+0.3	-1.4	+2.7	+7.5	+13.5	+9.2	+11.0
LMT	0.793, RRP	+0.4	-1.4	+0.5	+1.6	+7.2	+3.4	+7.1
FT	0.772, RRP	+1.1	+0.5	+0.4	+3.1	+8.5	+3.7	+9.2
Simple Cart	0.751, RRP	+1	-14.8	+0.5	+9.9	+12.5	+25.7	+13.0
Rand Forest	0.753, Roots	-0.9	-0.3	-0.1	+5.9	+6.3	+5.3	+5.5
S Logistic	0.793, RRP	+0.9	-0.9	0	+1.9	+6.7	+3.7	+7.6
Hyper Pipes	0.668, WP	+4	+4	+6.8	+3.8	-13.5	+3.8	-16.4
END	0.873, Roots	+0.8	+0.4	-1	+3.9	+8.2	+4.3	+9.1
Fil Classifier	0.863, Roots	-0.5	+1.3	+4.1	+1.8	+6.4	0	+1.8
Logit Boost	0.871, RRP	+0.3	-0.9	-0.6	+6.7	+9.3	+7.9	+10.2
RSS	0.844, RRP	+0.7	-0.3	+0.1	+4.6	+5.2	+5.6	+5.8
AdaBoost.M1	0.861, RRP	+0.3	-1.7	+0.3	+3.6	+7.7	+5.6	+7.7
Rotation Forest	0.84, RRP	+1.4	+0.1	-1.1	+5.5	+8.6	+6.8	+11.1
Bagging	0.836, RRP	+0.6	-1.3	+10.5	+7	+20.8	+8.9	+10.9
CVR	0.812, Roots	-1.7	+0.3	-1.5	+7	+13.1	+5	+12.9
Compl NB	0.784, WP	-0.2	-0.2	+1.8	-1.4	-3.1	-1.4	-5.1

Table 32: $F1^M$ Improvement/Degradation by comparing implemented VSM representations performances at feature numbers presented in Table 31 for each classifier.

It is clear from Table 32 that using RRP representation provides highest performance for 23 classifiers and using Roots representation provides highest performance for 11 classifiers compared to using other representations. Such improvement varies from 0.2% to 25.7%. However, improvement/degradation of performance when including phrases for all original VSM representations varies among classifiers and such variation is not clear to be indeed an improvement or degradation or none. This is investigated in the second experiment.

From Table 32 and results in appendix IV, the performance of 29 classifiers is higher when using RRP representation than when using Roots representation. The

performance of 23 classifiers is higher when using SSP representation than when using Stems representation. The performance of 30 classifiers is higher when using WP representation than when using Words representation. Also, the performance of 42 classifiers is higher when using Roots representation than when using Stems representation. The performance of 36 classifiers is higher when using Roots representation than when using Words representation. The performance of 41 classifiers is higher when using RRP representation than when using SSP representation and the performance of 37 classifiers is higher when using RRP representation than when using WP representation. Also, the performance of most classifiers among categories for 1000 best FSS is presented in appendix IV. Next is a detailed description of the second experiment.

5.4.1.2 Second Experiment

The second experiment performs a comparison between the forty seven classifiers in three parts using the experimenter in WEKA and tests for significance using a two-tailed (corrected) T test ($\alpha = 0.05$) along many criteria as F-measure, and Percent correct. The criteria chosen for these classifiers such as the number of epochs, stopping criteria, ..etc are shown in appendix IV. As such, the experiment's inputs are the representations used with the number of selected features required for chosen classifiers (more specifically 1000 and 5000 features only). Stratified 10-fold cross validation is used and the chosen number of repetitions is 5 for all parts of this experiment. The outputs are the contingency matrix, kappa statistics, percent correct, and macro F1 measures among others. However, only macro F1 values are shown here.

The first part compares the performance of classifiers of same type. FSS is performed using Chi-square on all those VSM representations and only best 1000

and 5000 selected features are maintained here. Results shown in Table 33 are for best 1000 features only (for 5000 features see appendix IV). For the second part, the best performing classifiers among representations for each type shown in Table 33 are chosen and the comparison among types is performed and tested for significance (results are shown in Table 34). The third part presents briefly the results of significance testing for classifiers between Root and RRP representations, Stem and SSP representations, and finally Word and WP representations (some results are presented in appendix IV).

Classifiers	Roots	RRP	Stems	SSP	Words	WP	v/ /*
Bayes based, significance relative to NBM							
BN	0.99 (0.03) v	0.98 (0.03) v	0.99 (0.03) v	0.99 (0.03) v	0.96 (0.06) v	0.96 (0.06) v	6/0/0
NB	0.65 (0.14) *	0.63 (0.13) *	0.58 (0.13) *	0.58 (0.12) *	0.52 (0.16) *	0.52 (0.12) *	0/0/6
NBM	0.78 (0.11)	0.78 (0.11)	0.72 (0.11)	0.72 (0.10)	0.67 (0.11)	0.68 (0.12)	
Complement NB	0.71 (0.11) *	0.72 (0.14)	0.70 (0.13)	0.69 (0.13)	0.65 (0.13)	0.67 (0.12)	0/5/1
NBMU	0.72 (0.15)	0.71 (0.11)	0.69 (0.13)	0.68 (0.16)	0.66 (0.13)	0.68 (0.14)	0/6/0
NBU	0.65 (0.14) *	0.63 (0.13) *	0.58 (0.13) *	0.58 (0.12) *	0.52 (0.16) *	0.52 (0.12) *	0/0/6
Functions, significance relative to SMO							
SMO	0.68 (0.13)	0.68 (0.11)	0.64 (0.12)	0.64 (0.11)	0.59 (0.16)	0.56 (0.15)	
Simple Logistic	0.83 (0.09) v	0.83 (0.11) v	0.79 (0.13) v	0.78 (0.11) v	0.63 (0.15)	0.63 (0.14)	4/2/0
RBF	0.64 (0.09)	0.57 (0.18) *	0.60 (0.14)	0.60 (0.17)	0.57 (0.14)	0.51 (0.14)	0/5/1
Rules, significance relative to PART							
JRip	0.87 (0.09)	0.85 (0.11)	0.73 (0.14)	0.74 (0.13)	0.69 (0.16) v	0.67 (0.16) v	2/4/0
PART	0.83 (0.10)	0.84 (0.11)	0.68 (0.13)	0.70 (0.12)	0.49 (0.17)	0.50 (0.14)	
Ridor	0.86 (0.08)	0.85 (0.09)	0.69 (0.11)	0.69 (0.13)	0.58 (0.15)	0.57 (0.14)	0/6/0
OneR	0.79 (0.15)	0.80 (0.14)	0.51 (0.17) *	0.52 (0.17) *	0.54 (0.20)	0.53 (0.18)	0/4/2
NNge	0.43 (0.20) *	0.42 (0.18) *	0.33 (0.19) *	0.34 (0.20) *	0.31 (0.19) *	0.32 (0.21) *	0/0/6
Decision Table	0.54 (0.18) *	0.55 (0.15) *	0.50 (0.17) *	0.48 (0.16) *	0.48 (0.12)	0.47 (0.16)	0/2/4
Trees, significance relative to J48							
J48	0.86 (0.08)	0.87 (0.09)	0.70 (0.13)	0.69 (0.11)	0.47 (0.16)	0.46 (0.16)	
Random Forest	0.69 (0.14) *	0.70 (0.13) *	0.62 (0.14)	0.63 (0.15)	0.61 (0.14) v	0.59 (0.16)	1/3/2
RepTree	0.87 (0.09)	0.86 (0.10)	0.64 (0.11)	0.67 (0.10)	0.52 (0.18)	0.53 (0.16)	0/6/0
BF Tree	0.88 (0.07)	0.88 (0.08)	0.69 (0.09)	0.68 (0.12)	0.55 (0.15)	0.52 (0.15)	0/6/0
FT	0.81 (0.10)	0.81 (0.08)	0.73 (0.13)	0.72 (0.11)	0.64 (0.16) v	0.64 (0.13) v	2/4/0
J48 graft	0.89 (0.08)	0.89 (0.08)	0.71 (0.12)	0.71 (0.10)	0.50 (0.16)	0.49 (0.15)	0/6/0
LAD Tree	0.84 (0.10)	0.84 (0.10)	0.57 (0.16)	0.59 (0.15)	0.52 (0.17)	0.52 (0.16)	0/6/0
LMT	0.83 (0.10)	0.83 (0.10)	0.79 (0.13)	0.78 (0.11)	0.64 (0.15) v	0.65 (0.14) v	2/4/0
Random Tree	0.43 (0.17) *	0.40 (0.17) *	0.37 (0.14) *	0.32 (0.15) *	0.34 (0.15)	0.37 (0.16)	0/2/4
Simple Cart	0.86 (0.08)	0.85 (0.10)	0.70 (0.10)	0.68 (0.10)	0.60 (0.16)	0.57 (0.15)	0/6/0
Miscellaneous, significance relative to VFI							
VFI	0.51 (0.21)	0.53 (0.15)	0.56 (0.15)	0.54 (0.18)	0.70 (0.17)	0.68 (0.14)	
Hyper Pipes	0.49 (0.19)	0.52 (0.17)	0.53 (0.18)	0.56 (0.17)	0.51 (0.18) *	0.52 (0.19)	0/5/1
Meta, significance relative to AdaBoost.M1							
AdaBoost.M1	0.92 (0.07)	0.93 (0.05)	0.84 (0.09)	0.82 (0.11)	0.71 (0.12)	0.74 (0.11)	
Attr Sel	0.84 (0.08) *	0.86 (0.09) *	0.68 (0.13) *	0.67 (0.15) *	0.54 (0.15) *	0.55 (0.12) *	0/0/6
Bagging	0.91 (0.06)	0.92 (0.07)	0.75 (0.11) *	0.74 (0.11) *	0.65 (0.15)	0.67 (0.13)	0/4/2
Class. Via	0.90 (0.09)	0.89 (0.07)	0.76 (0.12)	0.78 (0.11)	0.63 (0.15)	0.60 (0.12) *	0/5/1
Dagging	0.50 (0.23) *	0.52 (0.18) *	0.53 (0.13) *	0.50 (0.19) *	0.48 (0.19) *	0.46 (0.19) *	0/06
Decorate	0.83 (0.09) *	0.85 (0.10) *	0.74 (0.13) *	0.76 (0.12)	0.66 (0.14)	0.66 (0.11)	0/3/3
END	0.92 (0.07)	0.92 (0.06)	0.85 (0.08)	0.85 (0.09)	0.70 (0.16)	0.74 (0.12)	0/6/0
Filtered	0.83 (0.09) *	0.84 (0.08) *	0.81 (0.10)	0.81 (0.11)	0.80 (0.12)	0.80 (0.13)	0/4/2
Logit Boost	0.94 (0.06)	0.95 (0.05)	0.80 (0.12)	0.80 (0.10)	0.69 (0.12)	0.72 (0.12)	0/6/0
Multi Class	0.33 (0.11) *	0.34 (0.13) *	0.38 (0.14) *	0.37 (0.09) *	0.27 (0.12) *	0.31 (0.11) *	0/0/6
CBND	0.68 (0.16) *	0.71 (0.15) *	0.64 (0.13) *	0.62 (0.14) *	0.49 (0.15) *	0.49 (0.19) *	0/0/6
DNBND	0.70 (0.16) *	0.72 (0.16) *	0.64 (0.13) *	0.63 (0.15) *	0.48 (0.14) *	0.49 (0.19) *	0/0/6
ND	0.69 (0.15) *	0.73 (0.17) *	0.63 (0.15) *	0.63 (0.16) *	0.49 (0.17) *	0.47 (0.14) *	0/0/6
OCC	0.70 (0.14) *	0.75 (0.13) *	0.55 (0.18) *	0.57 (0.15) *	0.42 (0.19) *	0.43 (0.17) *	0/0/6
Random	0.73 (0.14) *	0.74 (0.12) *	0.65 (0.12) *	0.61 (0.14) *	0.60 (0.11) *	0.61 (0.15) *	0/0/6
RSS	0.93 (0.07)	0.93 (0.07)	0.83 (0.08)	0.85 (0.09)	0.72 (0.14)	0.71 (0.14)	0/6/0
Rotation Forest	0.90 (0.08)	0.90 (0.09)	0.81 (0.10)	0.82 (0.11)	0.66 (0.14)	0.68 (0.13)	0/6/0

Numbers in brackets are for standard deviation, win/ tie/ loose is abbr. as v/ /*

Table 33: Performance of implemented classifiers along different representations by selecting best 1000 features.

Table 33 presents the results of implementing different classifiers on all VSM representations for only the best selected 1000 features and testing for significance (for 5000 see appendix IV).

Results shown in Table 33 show that:

1- For Bayes-based classifiers, BN classifier performance is significantly the best among others (relative to NBM) for all representations (F1=99%) followed by NBM classifier (F1=78%);

2- For Function classifiers, Simple Logistic classifier performance is significantly the best among others (relative to SMO) for root, RRP, stem, SSP representations (F1=83%) followed by SMO classifier (F1=68%);

3- For Rule classifiers, JRip classifier performance is significantly the best among others (relative to PART) for word and WP representations (F1=87%) followed by Ridor classifier (F1=86%) but with no significance, then by PART classifier (83%);

4- For Tree classifiers, LMT classifier performance is significantly the best among others (relative to J48) for word and WP representations (F1=83%) followed by FT classifier (F1=81%) performance is significantly the best among others also for word and WP representations, then by BF Tree (88%) and RepTree (87%) classifiers with no significance; and

5- For Meta classifiers, Logit boost classifier performance is the best among others (relative to AdaBoost.M1) for all representations with no significance (F1=94%) followed by RSS classifier (F1=93%) with no significance, then followed by END classifier (92%) with no significance, then by AdaBoost.M1 classifier (92%).

It is noticed here that the performance of all best classifiers mentioned above is better than the performance of SMO classifier. The SMO classifier implemented here uses linear kernel. Other types of kernels, although not presented here, are tested for this classifier and results of using these types did not improve the classifier's performance but actually degraded it.

Table 34 re-represents best results of classifiers shown in Table 33 among types (except Miscellaneous) and tests for significance with respect to C4.5 classifier.

Type	Classifiers	Roots	RRP	Stems	SSP	Words	WP	v/ /*
Bayes-based	BN	0.99 (0.03)	0.98 (0.03)	0.99 (0.03)	0.99 (0.03)	0.96 (0.06)	0.96 (0.06)	6/0/0
	NBM	0.78 (0.11)	0.78 (0.11)	0.72 (0.11)	0.72 (0.10)	0.67 (0.11)	0.68 (0.12)	2/3/1
Functions	Simple Logistic	0.83 (0.09)	0.83 (0.11)	0.79 (0.13)	0.78 (0.11)	0.63 (0.15)	0.63 (0.14)	2/4/0
	SMO	0.68 (0.13)	0.68 (0.11)	0.64 (0.12)	0.64 (0.11)	0.59 (0.16)	0.56 (0.15)	0/4/2
Rules	JRip	0.87 (0.09)	0.85 (0.11)	0.73 (0.14)	0.74 (0.13)	0.69 (0.16)	0.67 (0.16)	2/4/0
	Ridor	0.86 (0.08)	0.85 (0.09)	0.69 (0.11)	0.69 (0.13)	0.58 (0.15)	0.57 (0.14)	0/6/0
	PART	0.83 (0.10)	0.84 (0.11)	0.68 (0.13)	0.70 (0.12)	0.49 (0.17)	0.50 (0.14)	0/6/0
Trees	LMT	0.83 (0.10)	0.83 (0.10)	0.79 (0.13)	0.78 (0.11)	0.64 (0.15)	0.65 (0.14)	2/4/0
	FT	0.81 (0.10)	0.81 (0.08)	0.73 (0.13)	0.72 (0.11)	0.64 (0.16)	0.64 (0.13)	2/4/0
	BF Tree	0.88 (0.07)	0.88 (0.08)	0.69 (0.09)	0.68 (0.12)	0.55 (0.15)	0.52 (0.15)	0/6/0
	RepTree	0.87 (0.09)	0.86 (0.10)	0.64 (0.11)	0.67 (0.10)	0.52 (0.18)	0.53 (0.16)	0/6/0
	J48	0.86 (0.08)	0.87 (0.09)	0.70 (0.13)	0.69 (0.11)	0.47 (0.16)	0.46 (0.16)	
Meta	Logit boost	0.94 (0.06)	0.95 (0.05)	0.80 (0.12)	0.80 (0.10)	0.69 (0.12)	0.72 (0.12)	5/1/0
	RSS	0.93 (0.07)	0.93 (0.07)	0.83 (0.08)	0.85 (0.09)	0.72 (0.14)	0.71 (0.14)	5/1/0
	END	0.92 (0.07)	0.92 (0.06)	0.85 (0.08)	0.85 (0.09)	0.70 (0.16)	0.74 (0.12)	4/2/0
	AdaBoost.M1	0.92 (0.07)	0.93 (0.05)	0.84 (0.09)	0.82 (0.11)	0.71 (0.12)	0.74 (0.11)	4/2/0
	Filt Classifier	0.83 (0.09)	0.84 (0.08)	0.81 (0.10)	0.81 (0.11)	0.80 (0.12)	0.80 (0.13)	3/3/0

Table 34: Performance of best two classifiers among types for different representations by selecting best 1000 features *(significance results are relative to J48)*.

C4.5 classifier [164]; [92]; [143] is one of the frequently studied classifiers that usually provide good results. This is why it is chosen here for significance comparison (in appendix IV significance testing for classifiers in Table 34 relative LMT and BN classifiers). Results in Table 34 show that BN classifier is significantly the best classifier among those used in this thesis and for all representations.

The third part compares the performance of the classifiers between: a) Roots and RRP representations, b) Stems and SSP representations, and c) Words and WP

representations. It tests their performance for significance. FSS is performed using Chi-square on all those representations and best 1000 and 5000 selected features are maintained (some results are presented in appendix IV). It is evident that although a slight improvement/degradation was obtained for some classifiers when the original VSM was extended, yet such results are not significant.

5.5 Conclusions

The results of implementing the proposed variant *TFIDF* and VSM representations for various classifiers have been presented in this chapter and can be briefed as:

1- The comparison between the performance results of most classifiers showed that using Roots representation significantly improved their performance than when using Stems or Words representations.

2- A comparison between the performances of those classifiers showed that using RRP representation significantly improved most of their performances than when using SSP or WP representations.

3- It is evident that although a slight improvement/degradation was obtained for some classifiers when the original VSM was extended, as explained above, yet such results are not significant.

4- It was noticed that the performance of BN was the best among implemented classifiers for all representations with $F1^{M} = 0.99$ and the effect of the variation of the number of selected features on its performance was the minimal among all classifiers.

5- Using the proposed variant of *TFIDF*, the classifiers' results in Table 34 showed that the best classifier was BN, followed by, in decreasing order, Logit Boost, Random Sub Space, END, Filtered Classifier Meta classifiers when features' number is 1000. The same can be concluded when features' number is 5000.

6- The high improvement in classification performance for most implemented classifiers when using roots representation compared to when using words representation provides an increase in knowledge obtained by using roots representation.

The overall analysis and critical review of the work and results for the two experiments reported in this chapter are explored in the next Chapter.

Chapter 6: Critical Analysis of Text Classification Methods' Performances

6.1 Introduction

This chapter provides critical analysis of the results of implementing various classifiers for six different VSM representations on TC performance whether for the first or second experiment as presented in Chapter 5.

This chapter is organized as follows: Section 6.2 compares the results of using phrases in document representation on Arabic TC performance shown in Chapter 5 with those implemented for English TC and further analyzes these results. Section 6.3 further compares between the results of implemented classifiers here according to their types in terms of their errors, training time and size (if tree learners) or number of rules (if rule learners). It also relates to the results of the same classifiers in other studies for English TC besides those for Arabic (if any). Finally, it concludes with the outcome of such analysis and comparison.

6.2 Effect of Using Phrases on Classification Performance

Since there are no reports of using phrases as representatives of features for Arabic TC, as was mentioned in subsection 2.4.1.1, this thesis extends the single term representation, as was explained in subsection 5.2.2.2, by including phrases with DF>1 to such representations. The results of the effect of this extension on TC performance were presented in section 5.4.1. Such results are in agreement with reported studies for English TC as [79]; [139]. However, in reported English TC, the effect of such phrases on TC performance was further investigated by including [79]

syntactic heuristics in phrase development or the [139] syntactic category of the word (using a POS-tagger) then extracting two levels of phrases from texts: a) proper nouns, b) complex nominal that express domain concepts, then word senses were used in place of simple words. In both studies [79]; [139] the detailed and explicit investigation of each case on different and large corpora showed that the effect of using phrases, word senses .., etc doesn't improve TC performance for English.

When word VSM representation was extended for English in (Moschitti and Basili, 2004) it seems to not improve TC. The new idea here (in this thesis) is to extend roots and stems representations with their respective phrases and studying their effect on TC performance. In this thesis, phrases were constructed as explained in section 5.2 and although results of including phrases for Arabic TC is similar to those for English TC and that the construction of Arabic phrases includes most of above cases for English, yet such linguistic options were not separately studied. This is due to the fact that there are no available online resources that would provide such options for Arabic except for the AWN software that can provide word senses but unlike WN it does not provide the percentage of each sense for a given word and as such cannot be used here.

Although, the method of extracting phrases implemented here for Arabic is slightly different than the method for extracting phrases for English, yet results of extending single terms with their respective phrases for Arabic is in consensus with those for English. Moschitti and Basili, (2004) [139] tended to explain the reasons for such results over English due to two possible properties of phrases: 1- Loss of coverage, and 2- poor effectiveness. The author tends to agree with such explanations here. Also, only one Arabic corpus was used here (rather small) which would lead to the

conclusion that further investigation of this matter (i.e. including phrases, word senses) is the next step on larger Arabic corpora.

6.3 Comparison between Classifiers

As far as known there is no work that investigated all WEKA classifiers for Arabic TC. However, in order to provide indicative comparison, the results of two works are presented here briefly. These are the works of [83] and [43]. Gelbukh and Kolesnikova [83] reported using different classification methods for recognizing semantic types of Spanish verb-noun collocations. Maximum reported F-measure values for such algorithms were: 0.903 for PART, 0.903 for JRip, 0.888 for Ridor, 0.908 for BF Tree, 0.915 for Simple Cart, 0.915 for FT, 0.893 for REPTree, 0.759 for NB, 0.783 for IB1, and 0.933 for SMO algorithms. Also, Chakraborty et al, [43] reported using various methods for classifying accounting literature. Maximum reported Accuracy values for such algorithms were: 0.832 for BN, 0.8531 for Complement NB, 0.8042 for NB, 0.8 for NBM, 0.6334 for NB Updatable, 0.8 for NBM Updatable, 0.74 for J48, 0.74 for J48 graft, 0.5 for LAD Tree, 0.73 for Random Forest, 0.57 for Random Tree, 0.8 for RepTree, 0.75 for Simple Cart, 0.64 for BF Tree, 0.65 for FT, 0.7 for LMT, 0.73 for ZeroR, 0.7136 for Ridor, 0.7136 for PART, 0.77 for OneR, 0.7433 for JRip, 0.8 for Decision Table, 0.6 for NNge, 0.7833 for CVR, 0.4333 for Multi Class Classifier, 0.7 for Simple Logistic, 0.5667 for SMO, 0.7833 for Attribute Selected Classifier, 0.7833 for Bagging, 0.6833 for Dagging, 0.7833 for Decorate, 0.8167 for END, 0.8 for FC, and 0.734 for Logit Boost.

Before comparing results obtained in this thesis with those presented above, further analysis of results for Arabic TC of each type of classifiers is presented next.

6.3.1 Function Classifiers

Function classifiers are classifiers that build models that use functions such as linear regression, logistic regression functions. The best two performing function classifiers here are further discussed, namely Simple Logistic and SMO.

Simple Logistic classifier [119] builds a logistic regression model using LogitBoost and incorporates attribute selection by fitting simple regression functions in LogitBoost. In [119] this method was compared with C4.5, AdaBoost, LMT, and two other algorithms and was tested on 32 data sets. Results show that its accuracy values are comparable with those of C4.5 and LMT. Although not presented here, Simple Logistic takes much less time than LMT. SMO classifier [110; 142] breaks large quadratic programming optimization problem into a series of smallest possible quadratic programming problems.

In [110; 142], SMO performance is better than linear SVM and faster. The SMO classifier tested here used linear kernel. Other types of kernels, although not presented here, were tested for this classifier and results of using these types did not improve the classifier's performance but actually degraded it. Also, the performance of this classifier for Arabic texts using macro-F1 in [146; 147; 148] is about 88-90% and highly different from its performance reported here for root representations (about 70%). In order to clarify the effect of variant *TFIDF* on this classifier performance further investigations are required.

It is worth mentioning that Logistic classifier [123] uses ridge estimator to reduce the effect of large number of covariates compared to number of observations and the high correlation of such covariates. The use of a ridge estimator would partially

explain the lower performance results of this classifier compared to Simple logistic and SMO for Arabic TC in this thesis.

Simple logistic performance in this thesis is significantly higher than the remaining function classifiers for all representations. For Words representation, the F1 values are 63% for Simple Logistic and 59% for SMO as is shown in Table 33, section 5.4. This is in general agreement with results obtained in [43]. However, for root representation the comparison between the performance of Simple Logistic with that of LMT, C4.5 and other classifiers, as shown in Table 34, section 5.4, shows their comparability. This conclusion is in agreement with that obtained in [119] although for different data sets.

6.3.2 Bayes-Based Classifiers

Bayes-based classifiers [92] are statistical classifiers. Studies investigating the performance of such classifiers found that NB classifier is comparable with decision trees and selected neural net work ones. Bayes-based classifiers have shown high accuracy and speed when applied on large databases. As far as known there is no comparison between various Bayes-based classifiers on Arabic texts and here a comparison is presented. Among the six classifiers investigated only the four highest in performance are further described.

BN classifier is [132] a classifier that uses a model that specifies a document to be represented by a vector of binary attributes. Thus, the number of times the word occur in the document is not captured and as such the probability of a document is found by multiplying the probability of all attribute values (occurring or not in document). The distribution then is based on multi-variate Bernoulli event model.

Thus, this model considers "... the document to be the event and the absence or presence of words to be attributes of the event".

NBM classifier [132] considers a model that specifies a document to be represented by the set of word occurrences from the document, i.e. word count in document is captured. The probability of a document is found by multiplying the probabilities of words that occur. In this thesis, the results of NBM and NBMU classifiers are highly comparable in terms of F1 values, time, and RMSE for all representations.

A comparison between the performance of BN and NBM classifiers as the number of features [132] was varied on five text corpora was investigated and it was found that NBM performance is almost uniformly better than BN and that NBM reduces error by an average of 27%. Also, BN had higher accuracy values than NBM for number of features generally less than 1000 (more near to 100 features). However, NBM is better in performance than BN above 1000 features. More specifically, in [132] BN accuracy is slightly higher than NBM by about (3 to 9)% for features number less than 1000 whereas in this thesis it is much higher by about 21%. Also, precision/recall breakeven point values for BN in [132] decreased largely as number of features increased whereas in this thesis F1 values for this classifier reduced slightly as number of features increased. The maximum precision/recall breakeven point values for BN in [132] for different corpora varied and were in the range 52-98% whereas in this thesis it was around 98% for all representations.

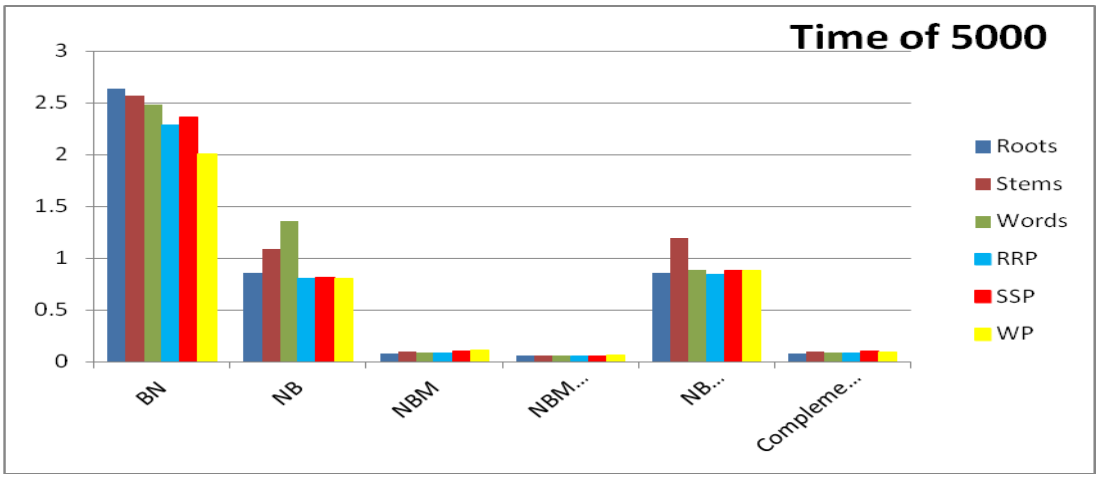
In this thesis, the results of BN performance above 10000 for features' numbers cannot be compared with those in [132] since it was not investigated due to hardware limitations. Unlike in [132] both BN and NBM classifiers in this thesis maintained an F1 value that reduced slightly as the number of features increased. Also, the FSS

method used in this thesis is the Chi-square method whereas in [132] it was mutual information.

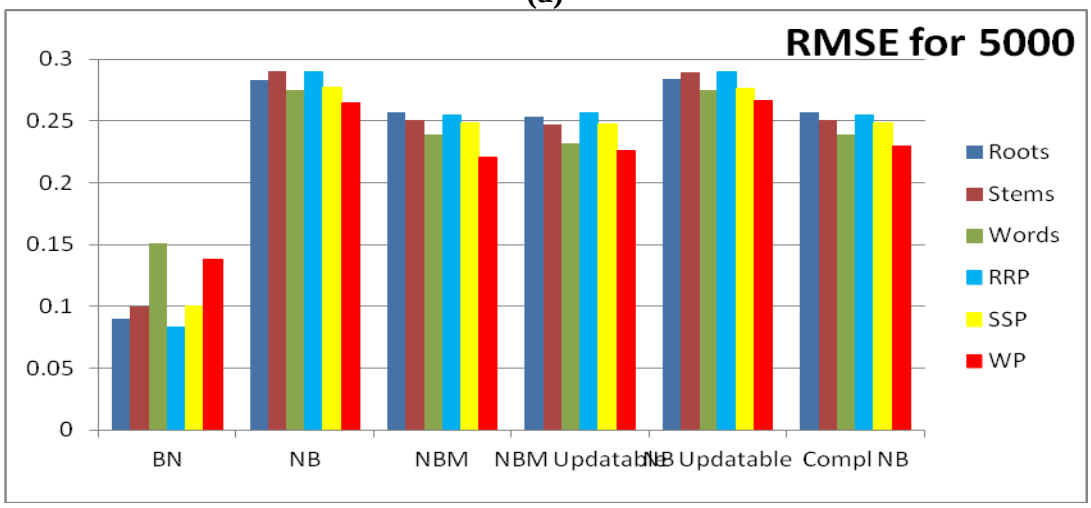
Complement NB classifier [149] modifies NBM classifier in four ways where it: 1- introduces a complement method to estimate the probability of a document, 2- normalizes weights of attributes, 3- uses a power law distribution to match term frequency distributions, and 4- uses two transformation pre-processing steps to improve the performance of NBM. This classifier was tested [149] on several text corpora and was compared with NBM and SVM classifiers. Its performance approaches that of SVM and outperforms NBM. Complement NB classifier performance was in this thesis less than that for NBM for all representations although comparable. This is different than that reported in [149].

In [43] the result of Complement NB was the highest among Bayes-based classifiers followed by BN, then by NBM. This is different than their performances reported in this thesis for word or WP representations.

As was presented in Table 33, section 5.4, the performance of Bayes-based classifiers was compared in terms of F1 values where the BN classifier's performance is significantly better than the remaining Bayes-based classifiers for all representations. In this chapter, their performance is also compared in terms of training time and RMSE as shown in Figure 37. Figure 37a presents the training time (in seconds) of such classifiers for all representations and the fastest one is the NBMU and NBM then Complement NB, and finally BN as the slowest among these classifiers (about 2 - 2.5 seconds). Figure 37b illustrates the error in their performance where BN classifier had the lowest error value followed by NBMU and NBM, and finally NBU classifier had the highest error value for all representations.



(a)



(b)

Figure 37: Comparison between Bayes-based classifiers' performance for all VSM representations according to (a) time, (b) RMSE

The results shown in Table 4 when testing Bayes-based classifiers especially the NBM one are lower than those reported in [146; 147; 148]. It is not evident at this stage whether the different performance of NBM classifier in [132] or that of Complement NB classifier in [149] is different than their performance in this thesis is due to the following factors: use of variant TFIDF weighting method, FSS method, text corpus, or all. In order to establish the cause of such difference, further research in future is required to study each factor separately.

6.3.3 Tree Classifiers

Decision Trees classifiers [92] are flowchart-like tree structures that are built from labeled data. The class of an unlabelled test instance is found by testing the attribute

values against the tree. Decision trees are [92] popular due to: 1- their handling high dimensionality of data, 2- their tree representation of knowledge is intuitive and easy to relate to by humans, and 3- their classification and learning steps are generally fast and simple. In this chapter, only the seven tree classifiers with best results are further discussed. Some of these trees are based on logistic regression. The efficiency [164] of tree classifiers is not only judged by their accuracy but also by other criteria as their errors and tree sizes. In this chapter all above criteria is further analyzed.

C4.5 classifier [143] generates a decision tree from a set of labeled instances by: 1- seeing if this set satisfies a stopping criterion, and if so the tree of this set is a leaf associated with the most frequent class in this set, then 2- using a test to recursively partition this set into smaller subsets. This algorithm uses the divide-and-conquer strategy in growing decision trees and adjusts Information Gain splitting criterion. This method was tested on 20 data sets (non-textual) and compared with its previous releases as well as other classifiers in terms of error rate and tree size using 10-fold cross validation. Results of this comparison [143] showed that this method produced smaller decision trees with higher accuracies and is superior to approaches that use global discretization.

LMT classifier [119; 170] is a method that combines linear logistic regression and tree induction by using such functions at tree leaves. In [170] LMT was compared in performance with AdaBoost classifier and their accuracy was in general comparable on 13 data sets. Yet, LMT training time was found to be much higher than that for AdaBoost. In [119] LMT was compared with C4.5, Simple Logistic, AdaBoost and others on 32 data sets in terms of accuracy and tree size. LMT in that work

outperformed C4.5 and was comparable to AdaBoost, Simple logistic and other classifiers but much lower in tree size than C4.5.

FT classifier [82; 119] is a method used for building trees that could use logistic regression functions at inner nodes and/or leaves. In [82] a comparison between performance of FT classifier and C4.5, linear Bayes, and Cruise [113] classifiers on 30 data sets was conducted. Results indicate that FT performance is generally comparable with linear Bayes and Cruise (for some datasets FT significantly wins while for others it loses).

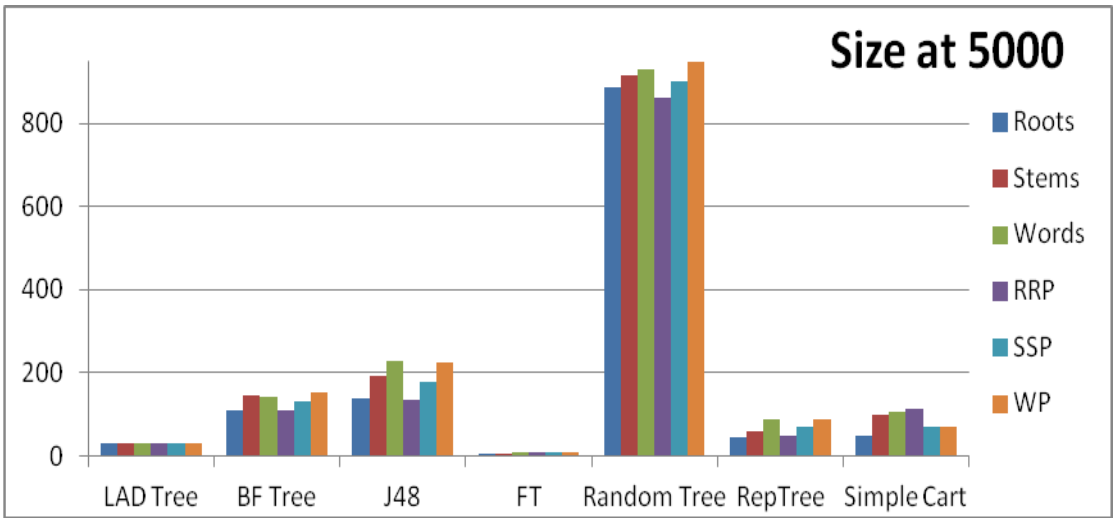
RepTree classifier is a fast method that builds a decision/regression tree using information gain/variance reduction, prunes it using Reduced-Error Pruning (with back fitting), and sorts numeric attributes only once. Simple Cart is [40] a method that builds classification trees by implementing minimal cost-complexity pruning.

BF Tree classifier [164; 77] builds a Best-First decision tree and uses binary splitting criterion. In [164] a comparison between the performance of BF Tree method and CART in terms of accuracy, training time and tree size on 38 data sets was conducted. Its results showed that both CART and BF Tree are comparable in terms of accuracy and training time but BF Tree outperforms CART in terms of having significantly lower tree sizes on most data sets.

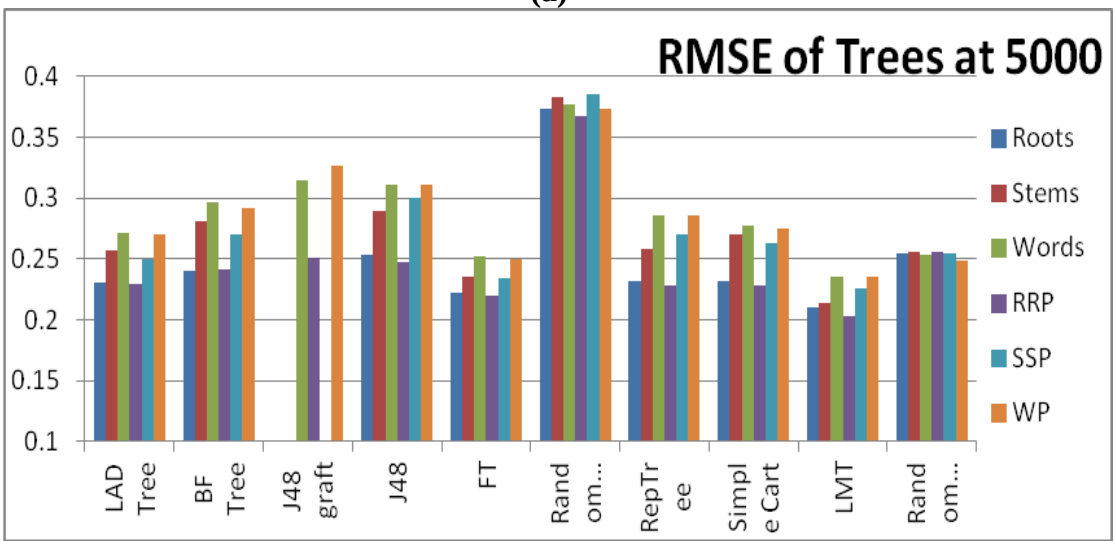
Random Forest classifier is [38] a method that combines tree classifiers such that each tree depends on the values of a random vector sampled independently and with same distribution for all trees in forest. In [38] it was found that Random Forests give results that are competitive to boosting and adaptive bagging classification algorithms in performance.

As far as known, this thesis is the first work that compares various tree learners for Arabic TC. Here, as was presented in Table 33, section 5.4, results of tree classifiers showed that, relative to C4.5, LMT and FT classifiers performances are significantly the best among others for word and WP representations. However, for the remaining representations, classifiers' performances are comparable. The maximum F1 values for root representation are: Simple Cart (86%), LMT (83%), LAD Tree (84%), FT (81%), BF Tree (88%), RepTree (88%) and C4.5 (86%). The maximum F1 values for word and WP representations are: Simple Cart (60%), LMT (64%), LAD Tree (52%), FT (64%), BF Tree (55%), RepTree (53%), Random Forest (61%), Random Tree (37%), C4.5 graft (50%) and C4.5 (47%). Also, the performance of Random Tree is lower than the remaining classifiers for all representations.

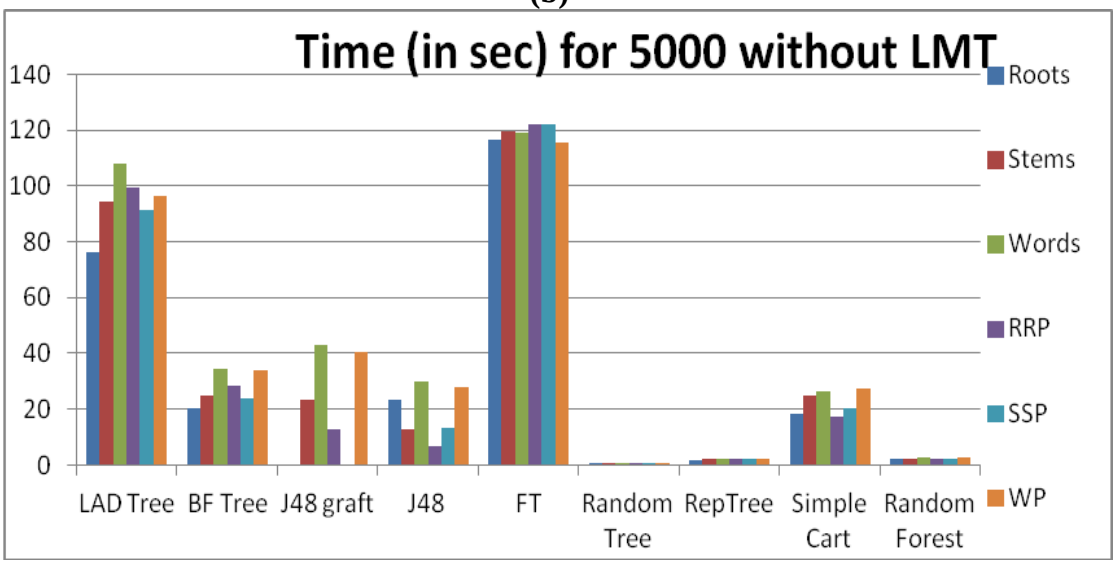
The above tree classifiers are further compared here in terms of their error, time and tree sizes as presented in Figure 38. From Figure 38a, for representations used, the least tree size was for FT method and the highest size was for Random Tree. Here, unlike results presented in [164], Simple Cart method provided less tree size than BF Tree method for all representations. Figure 38b showed that the method with least error is LMT followed by FT, Random Forest, Simple Cart, LAD Tree, Rep Tree, BF Tree, J48, and finally Random Tree. This is in general agreement with the finding of previous works mentioned above. Figure 38c illustrated that Random Tree method had the least training time among the tree classifiers followed by RepTree, Random Forest, J48, Simple Cart, BF Tree, LAD Tree, FT, and finally LMT (here time for LMT classifier is 10102.48 sec). Although, LMT provided best results in terms of significant F1 values, RMSE values but its time was the worst. So, in this thesis, FT classifier is better than other tree classifiers in terms of comparative F1 values, low RMSE and time values.



(a)



(b)



(c)

Figure 38: Comparison between decision trees classifiers' performance for all VSM representations according to (a) size of trees, (b) RMSE, (c) time

Results presented in Table 34, section 5.4 for comparing the performances of CART, C4.5, LMT, FT and BF Tree classifiers with classifiers as Simple Logistic or NB, are in general agreement with those in [164], [119], and [82] in terms of F1 values. Chakraborty et al, (2010) [43], although no significance testing was reported, showed that RepTree had the highest performance among implemented tree classifiers, followed by Simple Cart, J48, J48 graft, Random Forest, LMT, FT, BF Tree, Random Tree, and finally by LAD Tree classifier. Such results are different from those presented for the same classifiers in this thesis in terms of their order or values for word representation.

6.3.4 Rule Classifiers

Rule-based classifiers [47] produce rule sets which are relatively easy for people to understand and outperform decision tree classifiers [145]. Here, the five applied rule-based classifiers' results are further discussed.

NNge classifier is an algorithm [130] that generalizes exemplars without nesting or overlap and forms a generalization each time a new example is added by joining it to its nearest neighbor of same class and adopts a heuristic that performs modifying any generalizations in a uniform fashion, and is an extension of NGE [157] but doesn't allow hyper rectangles to nest or overlap. This method was tested [130] on 13 data sets (none of them are textual data sets) against CART, k-NN, Bayes, C4.5, and Composite Learner classifiers. In [157], NNge shows an improvement over standard NN classifier. Also, NNge tends to produce rules that test a large number of attributes. It also reduces the number of exemplars and improves classification accuracy while reducing classification time. In general this method was found to either outperform most above classifiers or is comparable to their performance.

Ridor classifier [80] is a technique that creates a two-way dependency relation between rules such that rule activation is investigated only in the context of other rule activation. Thus, it forms ripple down rules that form a binary decision tree where compound clauses are used to determine branching. Such clauses are not required to exhaustively cover all cases so that eventually this technique results in rule sets having minimal inter-rule interactions. This method [80] was tested on some large medical data and compared to ID3 and manual rules results. The comparison showed that Ridor is a simple, fast and effective method for rule induction.

OneR classifier is [102] a technique that outputs very simple rules on datasets and are called 1-rules since these rules classify an object on the basis of a single attribute. This method basically ranks attributes according to error rate (from training set results). Simple improvements presented in [102] to this method showed that it is similar (less than by about 3%) to the accuracy of C4 classifier when tested on 16 data sets (none is textual). However, its complexity is far less than C4.

Decision Table classifier is [114] a method that produce a decision table with a default rule mapping to the majority class. This representation has two components schema and body. Schema is a set of features that are included in the table and body consists of labeled instances from space defined by the features in the schema. To build this table, the algorithm decides which features to be included in schema using best-first search method to estimate the future prediction accuracy with k-fold cross validation. This algorithm [114] was tested on 16 datasets with discrete features and on 2 datasets with continuous features (non are textual). Also, this work compared the performance of this algorithm with C4.5 and majority classifiers. For discrete

features, this algorithm outperformed both C4.5 and majority classifiers for some datasets, whereas for continuous features this algorithm's performance is comparable to C4.5 in some datasets.

PART classifier [75] is a rule-induction method that avoids global optimization but produces accurate and compact rules. This method infers rules by repeatedly generating partial decision trees thus combines creating rules from decision trees and the separate-and-conquer rule-learning technique. The performance [75] of this classifier was tested against that of C4.5, C5.0 and RIPPER classifiers on 34 datasets. It was found that PART rule sets compare favorably to those of C4.5 and C5.0 and are more accurate, although larger, than those of RIPPER. Also, in a comparison between the performance of C4.5, PART and RIPPER [75] classifiers, it was found that PART has better performance than C4.5 and RIPPER in terms of CPU time, error rate and accuracy.

RIPPER classifier is [47] a propositional rule learning algorithm that performs efficiently on large noisy datasets (that extend naturally to first order representations) and are competitive in generalization performance with more mature symbolic learning methods as decision trees. It starts first with an initial model then secondly it iteratively improves it using heuristic techniques. This classifier was [47] tested on 22 datasets (few are textual) and compared with C4.5 and other previously rule-based algorithms in terms of generalization performance and efficiency. RIPPER was found to be extremely competitive with C4.5 rules and on most of datasets its error rate was lower than that for C4.5. Also, for noisy data sets, RIPPER is more efficient than C4.5 and scales nearly linearly with number of examples. In a comparison between the performance of C4.5 and RIPPER [47] classifiers on

different data sets, it was found that RIPPER has better performance than C4.5 in terms of error rate.

For Arabic TC, only the works [175]; [6] used RIPPER, PART and OneR methods among the rule-based algorithms in WEKA and compared their performances with C4.5. Thabtah et al, (2011) [175] applied these methods on Leeds Corpus of Contemporary Arabic with 427 texts among 15 categories, used Khoja's stemmer, and employed 10-fold cross validation. On average, the results of C4.5, RIPPER, PART, and OneR methods (using Roots for features) in terms of F1 are respectively 0.9, 0.887, 0.8877, and 0.1769. Also, it reported the methods' number of rules and error rates. It reported the number of rules to be lowest for OneR then in increasing order PART, RIPPER, and finally the highest for C4.5. However, the error rate was the highest for OneR, then in decreasing order RIPPER, PART and finally C4.5. Al-diabat (2012) [6] also applied these methods on 1526 texts with six categories, used Chi square for FSS and selected the top 30 features (Roots), and employed 10-fold cross validation. On average, the results of C4.5, RIPPER, PART, and OneR methods in terms of F1 are respectively 0.6085, 0.5788, 0.621, and 0.1331. The highest number of rules in [6] was for C4.5 then in decreasing order PART, RIPPER and finally OneR.

As was shown in section 5.4, the results of applying the five rule-based algorithms, namely NNge, OneR, Decision Table, Ridor, RIPPER, and PART, have shown that the best performing classifier for Roots or RRP representations among rule learners in terms of F1 values is the JRip classifier (87%) followed by Ridor (86%) and PART (83%), then by OneR (79%), then by Decision Table (54%) and finally by NNge (43%) classifiers. This is not in agreement with the results presented above for

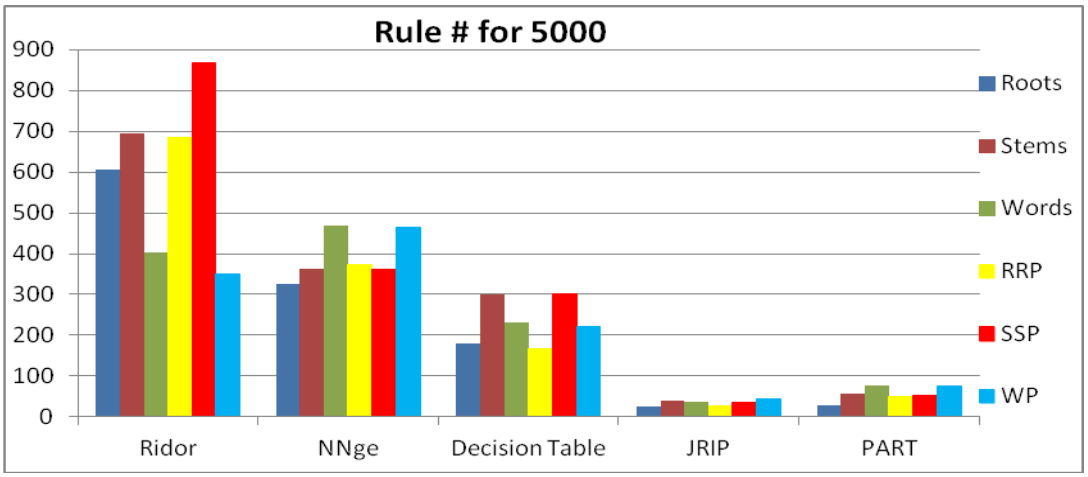
Arabic TC above in [175]; [6] in terms of values or order. However, unlike results in [175; 6], as shown in Table 34, section 5.4, RIPPER classifier is significantly better than C4.5 for 2 representations and comparable with it for the others. In [43], the highest best performing rule learner was the Decision Table followed by OneR, then JRip, then Ridor and PART and finally by NNge classifiers.

In this thesis, when using words representation the performance of these classifiers is different from those in [43] whether in order or values. In this chapter, these algorithms' performances are further compared in terms of number of rules, building time, and RMSE for 5000 top selected features only. This is presented in Figure 39.

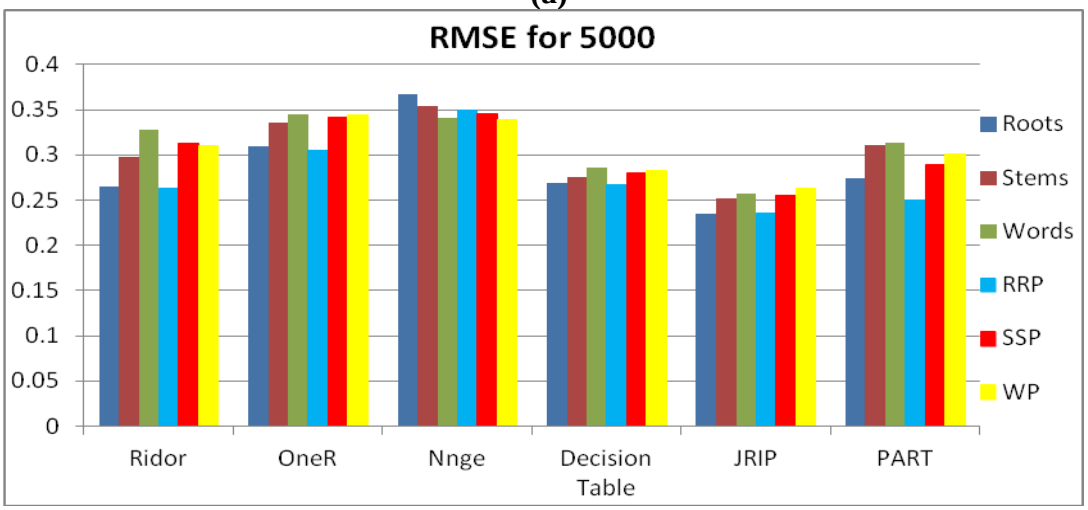
It can be concluded from Figure 39a that the least number of rules was in OneR (although not shown) followed, in increasing order, by RIPPER, PART, Decision Table, NNge, and finally Ridor for all representations. This is in agreement with results presented in [75] regarding the performance of RIPPER and PART in terms of number of rules and with the results of [175] for Arabic TC in that regard. From Figure 39b it can be seen that RMSE values are the least for RIPPER, then in increasing order Decision Table, PART, Ridor, OneR, and finally NNge. However, Figure 39c presents the amount of time taken for building these classifiers and the fastest classifier among these is OneR, and then less fast comes NNge, then RIPPER, PART, Ridor, and finally Decision Table.

TC performance in terms of accuracy for English was [79] reported to be about 78% (no FSS) whereas for Arabic as reported here for words representation is about 69%. Another difference that is noted here is that the number of rules and CPU time for RIPPER is less than for PART classifier whereas the reverse was concluded [75] for such classifiers. It is not clear here the cause of such difference and requires further

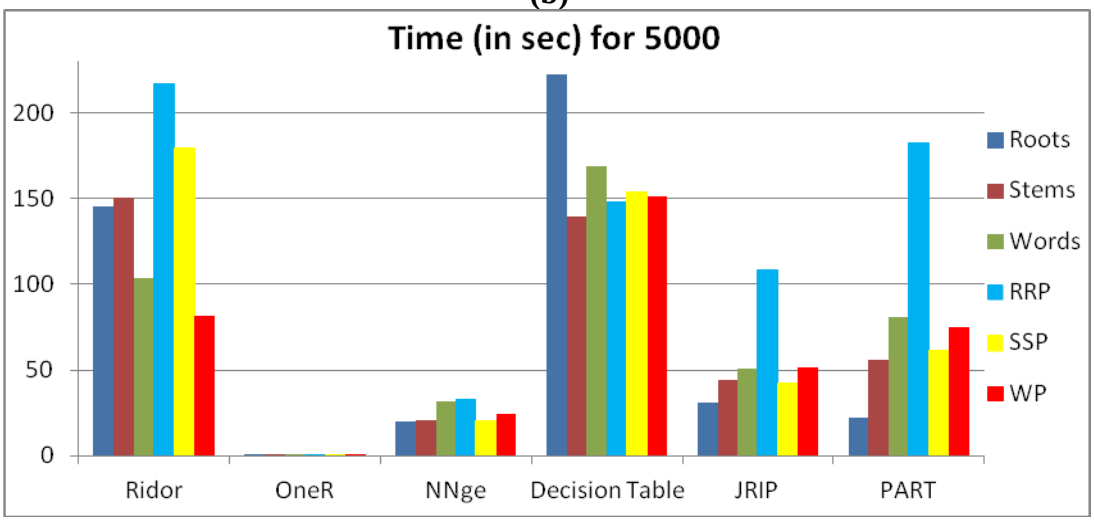
investigation whether on the same corpus but using other term weighting methods or on other corpora.



(a)



(b)



(c)

Figure 39: Comparison between rule-based classifiers' performance for all VSM representations according to (a) number of rules, (b) RMSE, (c) time

It is noteworthy that in [175; 6] the reported accuracy of OneR classifier (about 0.1) is much lower than its value here (accuracy values for WP, Words, SSP, Stems, RRP, and Roots representations are respectively 0.5833, 0.5249, 0.5323, 0.5485, 0.6294, and 0.6182). It is considered here that such large difference in accuracy values might be partially due to the use of the proposed *TFIDF* variant. However, this requires further investigation in future.

6.3.5 Miscellaneous Classifiers

The two miscellaneous classifiers applied here are VFI and HyperPipes. Both performances for all representations were rather low but HP method had higher F1 values than VFI as was presented in section 5.4. For Arabic TC these methods have not been implemented before.

In [65] HP classifier accuracy was compared with those of SVM, C4.5, NB and TWCNB (a modification of NB for TC found in [149]) classifiers in WEKA software and found that HP accuracy is comparable with TWCNB but higher than the others.

VFI was compared with HP as well as NB classifiers[49] on 35 data sets and it was found that VFI is faster than NB but slower than HP and less in accuracy than both NB and HP for most sets and the highest performance was for NB then HP then VFI.

In this thesis, the results of comparing VFI and HP classifiers performances in terms of F1 are in agreement with results shown above. Also, comparing the performance of HP, VFI with NB here shows that for only words and WP representations it is similar to those reported in [65].

[49] further info regarding results can be found at: http://bio.informatics.indiana.edu/ml_docs/weka/weka.classifiers.VFI.html

6.3.6 Meta Classifiers

Meta classifiers use either ensemble of base classifiers, boosting or bagging to improve the performance of usually a weak classifier. Here, only the best seven performing Meta classifiers (presented in Table 33, section 5.4) are further explained and compared.

Filtered classifier[50] runs an arbitrary classifier on data that has been passed through an arbitrary filter (here Discretize).

END classifier [74] is an Ensemble of Nested Dichotomies that chooses randomly the nested dichotomies and repeats this process for 10 iterations then takes the average. In [74] this method was tested on 21 data sets and its implemented base classifiers was either C4.5 or Logistic regression. Both were tested to see if this ensemble improvement/degradation of performance depends on the base classifier or not. It was compared to C4.5 and logistic regression, among others, and it was found that END produces more accurate classifications than when applying C4.5 and logistic regression. In this thesis, the base classifier for END is C4.5.

LogitBoost classifier [77] applies additive logistic regression with Decision Stump base classifier and best-first as the splitting strategy. For multi-class problem, direct generalizations based on multinomial likelihood were derived and their performance were found to be comparable to some boosting algorithms and far superior in some. This method also performed a slight modification to boosting that reduced computation through weight trimming. In [77] LogitBoost was tested on simulated and real data sets (non textual) and compared with AdaBoost (based on C4.5), CART, and Bootstrapping algorithms as well as C4.5 for two-class and multi-class

[50] info on Filtered clasifier was taken from: http://weka.sourceforge.net/doc.dev/weka/classifiers/meta/FilteredClassifier.html

problems. This classifier's performance is comparable to AdaBoost performance on real data sets. In [77], it was tested to investigate the effect of using C4.5 and Decision Stump (as base classifiers) on LogitBoost performance where both had the same effect. In this thesis, although not presented here, both base classifiers were used and their effect on Logit Boost performance is comparable which is in agreement with results obtained in [77].

Random SubSpace classifier (RSS) [99] consists of multiple trees constructed systematically by pseudo randomly selecting subsets of features. This method can take advantage of high dimensionality, unlike others that suffer from it, since it improves on generalization accuracy as it grows in complexity. In [99] this method was tested on 4 data sets and compared with C4.5 (with/out pruning), AdaBoost, and bootstrapping methods. Results of such tests showed that RSS is more accurate than C4.5 and higher than boosting and bootstrapping methods in accuracy.

Rotation Forest (RF) classifier [150] is an ensemble of base classifiers (in this thesis C4.5) where the feature set is split into subsets randomly (K is the number of subsets and a parameter of the algorithm) and Principle Component Analysis (PCA) is applied to each subset. K-axis rotations take place to form the new features for the base classifier, i.e. reassembling a new extracted feature set while keeping all components. In [150] this method was compared with Bagging, AdaBoost, and Random Forest algorithms and tested on 33 benchmark datasets. Results were significantly favorable to RF classifier in terms of accuracy over all others.

AdaBoost.M1 classifier [78] is a method that boosts the performance of a weak learner (in this thesis C4.5). This method was compared with C4.5 along with the Bagging classifier, using FindAttrTest, FindDecRule and C4.5 weak algorithms. In

[78] 27 data sets were used for comparing such methods' results show that when using C4.5 as weak learner, Boosting and Bagging seem more evenly matched even though Boosting has a slight advantage.

Bagging classifier [39] is a method for aggregating multiple versions by making bootstrap replicates of learning set of a predictor (in this thesis C4.5) and using these to get an aggregated predictor. This method was tested on different data sets (whether real or simulated) and compared it with CART's performance, subset selection in linear regression. In [39] Bagging accuracy results is substantially higher than others.

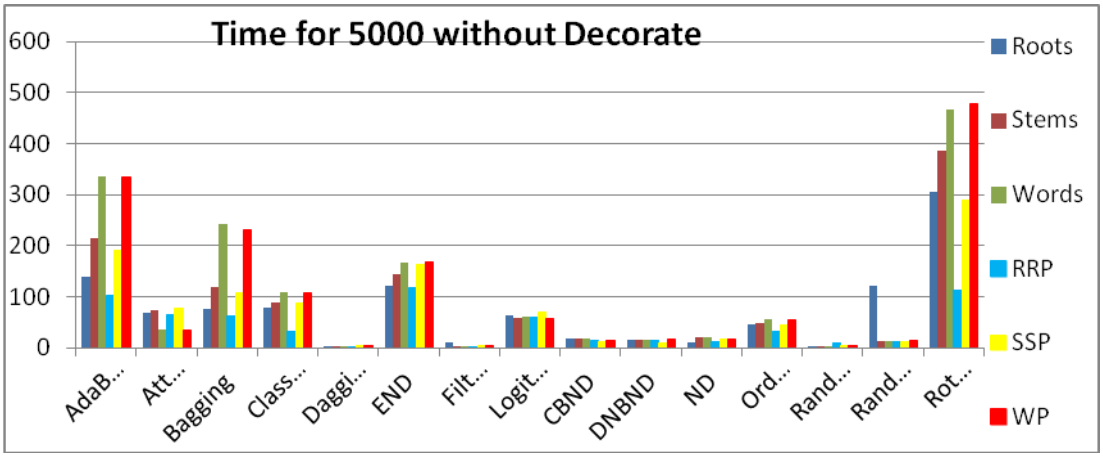
Classification Via Regression (CVR) classifier [76] is a model tree that takes the form of a decision tree with linear regression functions at its leaves (base classifier is M5P [144], [179]). In [76] this method was found to be significantly more accurate than C5.0' and linear regression when tested on 33 datasets.

As was shown in Table 33, section 5.4, the results of applying 17 meta algorithms showed that for Roots or RRP representations Logit Boost classifier's performance is the best among others (relative to AdaBoost.M1) (F1=94%) followed by RSS classifier (F1=93%), then by END classifier (92%), then by AdaBoost.M1 classifier (92%), then Bagging (91%), CVR (90%), RF (90%), Att. selected classifier (84%), then Dagging (83%) where the highest performances in terms of F1 of above classifiers are not significant. This is not in agreement with the results presented above in [43], whether in values or order, since the best performance, in decreasing order, is for END, Filtered classifier, CVR, Decorate, Bagging, Att. selected classifier, Logit boost, Dagging, and finally Multi Class Classifier.

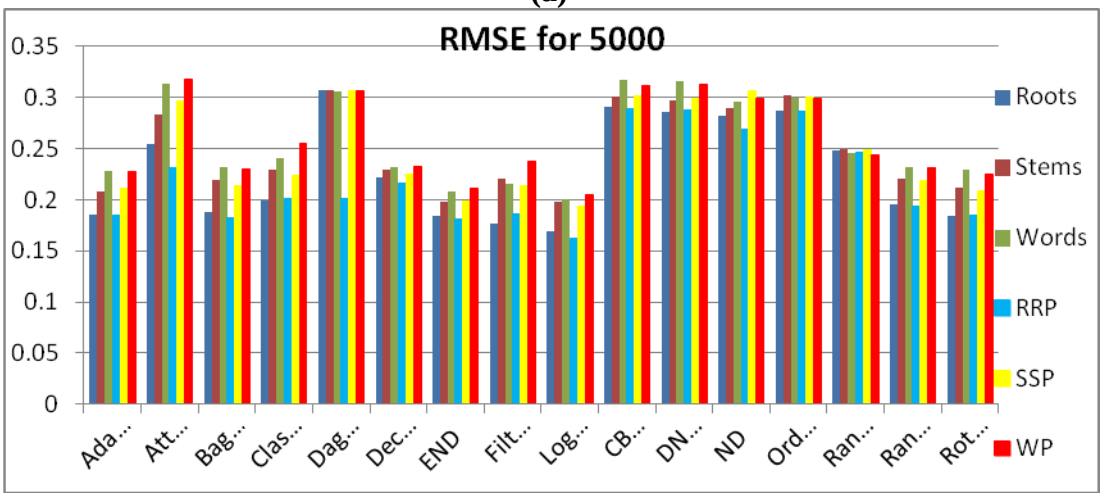
The performance of the AdaBoost.M1 here in terms of $F1^M$ varied among representations in boosting C4.5 performance from about 3-4% for root representations to about 10-12% for word representations for 5000 features. Also, as mentioned (in subsection 2.4.2) in the work of Raheel at el, [147] the maximum $F1^M$ was found to be about 88.5% for root representation. However, here for roots representation, the performance of this classifier is about 92%.

From results in Table 34, section 5.4, the comparison between the performances of best classifiers among Meta classifiers whether among these classifiers or with other types as C4.5 or Logistic is in agreement with reported results in [99], [74] and [77]. In this chapter, these algorithms' performances are further compared in terms of time and RMSE for top 5000 selected features only as is presented in Figure 40.

Figure 40a presents a comparison between these classifiers in this thesis in terms of building time. Some of these classifiers' time is small, although its macro F1 is low such as Dagging, yet for few others that have comparatively high macro F1; their time is still small such as RSS or FT classifiers. However, most Meta classifiers that have high macro F1 values have a rather high time. Figure 40b presents the best performing classifiers in terms of lowest RMSE values. Such values are in agreement with their macro F1 values (i.e. classifiers with lowest RMSE have highest F1 values).



(a)



(b)

Figure 40: Comparison between Meta classifiers' performance for all VSM representations according to (a) time, (b) RMSE

Next is a comparison between the best performing classifiers of each type is further analyzed.

6.3.7 Comparison between Classifiers:

In previous subsections, a comparison between the performances of classifiers of same type was conducted and analyzed. Here, the best performing classifiers among each type are compared further. Before doing that, a summary of what is reached so far in this thesis is presented.

1- For Bayes-based: the performance of Bayes-based classifiers was compared in terms of F1 measure where the BN classifier's performance (max. F1 is 99%) is significantly better than the remaining Bayes-based classifiers for all representations,

2- For Functions: Simple logistic performance is significantly higher than the remaining function classifiers for all representations, where for Roots representation, the F1 values are 83% for Simple Logistic and 68% for SMO,

3- For Trees: results of tree classifiers performances showed that (max. F1 values), relative to C4.5, LMT (83%), FT (F1=81%), BF Tree (88%), RepTree (87%), C4.5 (86%), and Simple Cart (86%) are comparable for most representations,

4- For Rules: results of applying the five rule-based algorithms have shown that the best performing classifier for Roots or RRP representations among rule learners in terms of F1 values is the JRip (87%) classifier followed by Ridor (86%) and PART (83%), then by OneR (79%), then by Decision Table (54%) and finally by NNge (43%) classifiers,

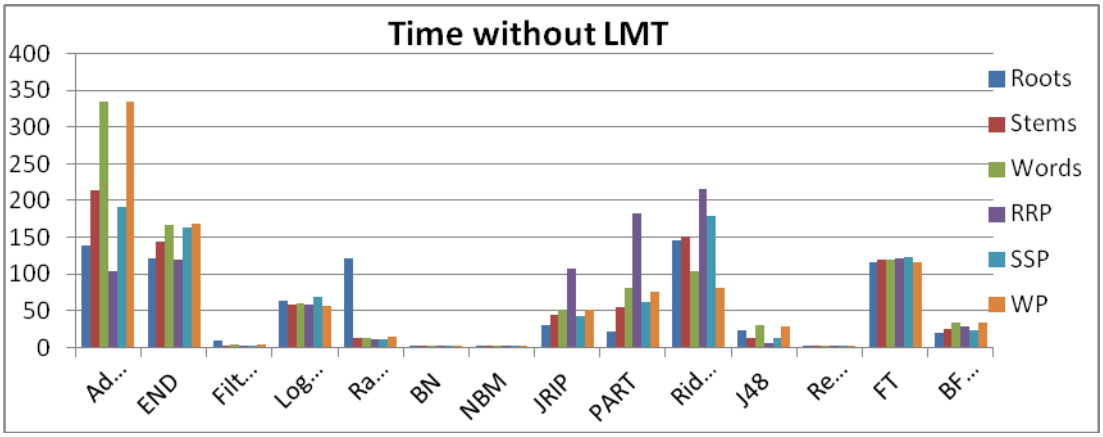
5- For Miscellaneous: results of comparing VFI and HP classifiers performances are in agreement with previous works results,

6- For Meta: results of applying 17 meta algorithms showed that for Roots or RRP representations Logit boost classifier's performance is the best among others (relative to AdaBoost.M1) (F1=94%) followed by RSS classifier (F1=93%), then by END classifier (92%), then by AdaBoost.M1 classifier (92%), then Bagging (91%), CVR (90%), RF (90%), Att. selected classifier (84%), Decorate (83%), Filtered classifier (83%), then Dagging (50%) where the highest performances in terms of F1 of above classifiers are not significant.

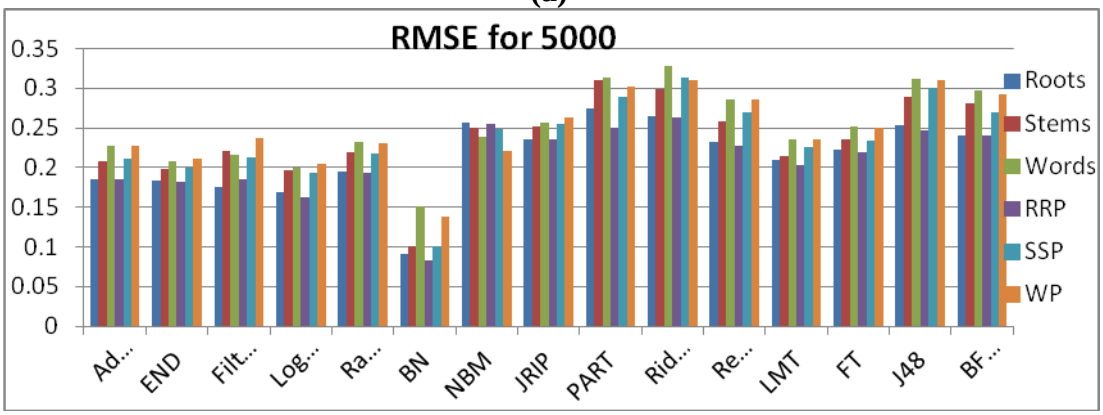
As is shown in Table 34, section 5.4, the BN classifier (99%) is significantly the best one among used classifiers in terms of F1 values for all representations. The second best performance classifier is Logit Boost (94%) followed by in decreasing order,

RSS (93%), AdaBoost and END (92%), BF Tree (88%), RIPPER and RepTree (87%), C4.5 and Ridor (86%), Simple logistic, PART, LMT, and FC (83%), FT tree (81%), NBM (78%), then SMO (68%).

In order to further compare these classifiers, their training time and RMSE values are compared. This is presented in Figure 41. From Figure 41a, in terms of least training time, the best classifier is NBM followed by BN, RepTree, and then Logit Boost. From Figure 41b, in terms of least RMSE values, the best classifier is BN, followed by Logit Boost, then END and FC classifiers.



(a)



(b)

Figure 41: Comparison between best classifiers' performance for all VSM representations from different types according to (a) time, (b) RMSE

Thus, it can be concluded that the best classifier among all used ones in this thesis in terms of F1, time, and RMSE values is BN, followed by Logit Boost for all representations.

6.4 Conclusions and Future Work

The results of implementing the proposed variant *TFIDF* and VSM representations for various classifiers have been presented and further analyzed and compared in this chapter and can be briefed as:

1- The comparison between the performance results of most classifiers showed that using Roots representation significantly improved their performance than when using Stems or Words representations. Also, a comparison between the performances of those classifiers showed that using RRP representation significantly improved most of their performances than when using SSP or WP representations. However, it is evident that although a slight improvement/degradation was obtained for some classifiers when the original VSM was extended, yet such results are not significant.

2- The classifiers' results showed that the best classifier was BN, followed by Logit Boost classifier when features' number is 1000 for all representations in terms of F1, training time, and RMSE values. The same can be concluded when features' number is 5000. However, the effect of the variation of the number of selected features on BN performance was the minimal among all classifiers. This is so since for 5000 features the improvements were slightly less, among the implemented classifiers, the BN classifier had the least percentage of required features to obtain the maximum performance: about 4.5% for Roots, about 1% for RRP, about 0.28% for Stems, about 0.23% for SSP, about 0.043% for Words, and about 0.08% for WP. This is different from previously reported studies on English TC.

3- The high improvement in classification performance when using roots representation compared to when using words representation suggest an increase in

knowledge obtained by using roots representation for most implemented classifiers and that using BN or Logit Boost classifiers for obtaining excellent Arabic TC results is favorable not just in terms of accuracy, F1, time values, but also in lower error values.

4- It is not clear at this stage whether using the proposed variant of *TFIDF* does indeed improve the performance of classifiers or not since its effect in this thesis was not compared with that of other term weighting methods. However, results in this work indicate that this variant improves or has no effect on TC performance. This is so although k-NN classifier's performance in this thesis was poor.

5- The comparison between different rule learners in this work is similar to previously reported works on other data sets but rather different than those for Arabic TC in terms of F1, time, RMSE values and number of rules. The comparison between different tree learners in this work is rather similar to those of previously reported works on other languages in terms of F1, time, RMSE values and size of trees. The same applies to the comparison between the results of Meta learners here in terms of F1, time, and RMSE values. However, the comparison between different Bayes-based learners in this work is different than previously reported works on other languages in terms of F1, time, and RMSE values.

6- It is noteworthy that, for 1000 features, the amount of improvement in classification performance that AdaBoost.M1 provided for C4.5 classifier varied among VSM representations from about 6% for roots or RRP to about 24-28% for words or WP. Also, for 1000 features the amount of improvement in classification performance that RSS provided for RepTree classifier varied among VSM representations from about 6-7% for roots or RRP to about 18-20% for words or WP.

For 1000 features, the amount of improvement in classification performance that END provided for C4.5 classifier varied among VSM representations from about 5-6% for roots or RRP to about 23-28% for words or WP. For Logit Boost, the effect of the base classifier choice on its performance is tested among representations and although not presented but was found to be negligible. In general, the small differences in the performance of Meta classifiers are not significant among representations.

7- The high improvement in classification performance when using roots representation compared to when using words representation suggest an increase in knowledge obtained by using roots representation for most implemented classifiers.

In future, further work is expected to compare the performance of such classifiers on other Arabic corpora (once acquired) and to investigate the effect of the proposed variant of TFIDF method by comparing its effect on TC performance with other weighting methods on different Arabic corpora. Also, further work is expected to investigate the effect of other different VSM representations for Arabic on TC performance and include further feature choices as word senses.

Chapter 7: Conclusions and Recommendations

This thesis explores and improves different preprocessing methods and investigates their effect on TC performance for Arabic text. The preprocessing methods for TC that are investigated are concentrated in two areas: morphological analysis and document representations. More specifically, this thesis improves two existing root extraction techniques by proposing and implementing an algorithm that handles irregular words and compares between these techniques in terms of their accuracy and execution time [13], [14]. It also proposes and implements an adjustment and two expanded weight-based techniques and compares their performance with the original ones [14]. Throughout the process of analyzing the results of root extraction techniques, further modifications are presented such as handling foreign Arabized words to establish a root extraction system.

This thesis investigates the effect of using the outputs of the best reported accuracy root extraction technique among those implemented here (i.e. roots or stems as alternatives of their respective words in document representation) on single-labeled TC performance as well as including other feature choices such as phrases in document representation. It also proposes and implements a variant *TFIDF* for weighting features in VSM representation. Finally, in this thesis, such six VSM representations and the proposed weighting method are used in the implementation of various single-label TC techniques and a comparison and analysis of their performances is conducted.

7.1 Research Contributions

This section presents the contributions for each research question introduced in Chapter 1.

- ***Research contributions regarding the first research question***

  "What are the steps to develop an Arabic corpus from two different small collections to be manually classified as single-labeled corpus among eight classes?"

- This thesis contributes to the literature of Arabic corpora by reviewing briefly available corpora and emphasizes the necessity for developing an Arabic corpus that eventually will be available online (Chapter 3).

- This thesis contributes to the literature of corpus development in developing an Arabic corpus. This corpus was developed after collecting texts from various online newspapers and downloading the LACC corpus that were all single-labeled. Then, a re-categorization process of the LACC corpus among eight pre-defined classes was performed in order to unify the labeling of the developed corpus. This resulted in a final corpus of 804 documents and about a million words. The generality differs among categories for this corpus (Chapter 3).

- ***Research contributions regarding the second research question***

  "What are the available root extraction methods to be implemented in this research? What are their disadvantages and how to improve and compare their performance in order to obtain the most correctly outputted stems and/or roots for respective words using the developed Arabic corpus in order to finally propose to develop a root extraction system?"

- This thesis contributes to the literature of available root extraction techniques applied to Arabic by providing a brief review of these techniques and compares them in terms of their performance, availability and advantages and/or disadvantages (Chapter 2).
- This thesis identified specific disadvantages of two available different root extraction techniques, namely their lack of extracting roots for irregular words, contributed by improving the performance of these two techniques through the development and implementation of an algorithm that handles such irregular cases (Chapter 4). The implementation of this algorithm reported improvement of the performance of original two root extraction techniques or their proposed Expanded methods was in the range of 7% - 14%, and its efficiency was also presented in terms of its execution time where in general, its space and time complexity is linear as long as the number of words in documents was less than 8,000. This algorithm can easily be included in any root extraction technique that does not handle such cases. Results of such improvements were published in [13] and [14]. Also, after critically analyzing the results of these techniques, handling the effect of foreign Arabized words was performed by developing a list that includes 7,227 foreign words and names of places, countries, and cities. Also, a proposed root extraction system is developed that gathers advantages of some of the implemented algorithms as well as handling the foreign words.
- ***Research contributions regarding the third research question***

  "How varying feature choices in Vector Space Model representation of corpus will affect the performance of various text classification methods as well as proposing and implementing a variant of TFIDF term weighting? If

there is an improvement in text classification performance, would it be statistically significant?"

- This thesis identified a problem in the frequently implemented term weighting method *TFIDF* and proposed a variant *TFIDF* method that takes into consideration the effect of the presence of a term among different classes. It represented the developed corpus using three different representation schemes, *words*, *stems* and *roots*. It also proposed three different VSM representations by extending the original representations through including their respective *phrases*. The comparison between the performance results of forty seven different classifiers showed that using *Roots* representation significantly improved their performance than when using *Stems* or *Words* representations for most classifiers. Also, comparing the performance results of those classifiers showed that using *RRP* representation significantly improved classifiers performance than when using *SSP* or *WP* representations for most classifiers. However, although a slight improvement/degradation was obtained for most classifiers when the original VSM was extended, yet such results are not significant (Chapter 5).
- ***Research contributions regarding the fourth research question***

  "Which classifiers applied to various representations of Arabic corpus have the best performance? Are the results obtained for such classifiers in agreement with previously reported studies?"

- This thesis contributes to the literature of available classification techniques applied to Arabic by providing a brief review of these techniques and compares them in terms of their performance, availability, and advantages

and disadvantages (Chapter 2). Also, it contributes to the literature of classification techniques by providing a brief review of those techniques that are used for Arabic TC in this thesis and compares their performances in terms of their F-measure, training time, and root mean square error values (Chapter 6).

- This thesis identified the lack of using all well-performing classifiers for Arabic TC. Thus, it used 47 classifiers that are available in WEKA software and tested these classifiers on all VSM representations of the developed Arabic corpus. It compared between the performances of these classifiers in terms of their F1 values and tested for significance (Chapter 5). This thesis further analyzed the results of such classifiers by comparing these results with those of previous works and comparing between these classifiers' performances by their training time and root mean square error values (Chapter 6). From these classifiers' performances, it is concluded here that the best one among them is the Bayes Net classifier (with $F1^M$ = 99%, training time about 2.5 seconds, RMSE value about 0.1 when using the *Roots* representation).

This research provided unique contributions in that it developed the first Arabic corpus that can be used in both single-labeled and multi-labeled TC although here it was used only for single-label TC. It also proposed a simple and efficient algorithm that handles the less visited irregular words in Arabic. Furthermore, it handles foreign Arabized words. It then explored extending the usual VSM representation of documents by including phrases, stem phrases, or root phrases in such representations and compared such extension with the usual choice of features in terms of *words*, *stems*, and *roots*, as well as proposed a variant of the well-known

TFIDF weighting method and implemented and compared various classifiers to investigate the effect of such preprocessing methods on TC performance.

7.2 Research Limitations

Although the results can be considered promising and positive, the research has some limitations that should be highlighted.

- The first limitation of this study is in the small length of corpus developed.
- The second limitation of this study is the lack of available efficient online processing tools for Arabic.
- The third limitation of this study includes the length of time for the study where a short time frame especially for the last portion of thesis resulted in the most of the limitations.

7.3 Recommendations for Further Work

This research essentially covered two main areas of research: the development, improvement, and comparison between two techniques for root extraction to finally propose a root extraction system, and the investigation of the effect of preprocessing methods on improving TC performance for Arabic.

Throughout the development of Arabic corpus and the re-categorization of its texts among eight classes, as briefly mentioned in Chapter 3, some of such texts were hard to categorize to only one class. This led to distributing a questionnaire on native Arabic speakers in Petra university and other working environments in Amman, Jordan took place throughout the period 4/11/2010 – 1/2/2011. The objective of the questionnaire was to obtain a text collection with multi-labels (at least for some of its texts when applicable). It was requested that each text be classified by at least two

participants. The total number of texts that were classified was 1,985. Also, Hooper's measure of consistency [100] was used for some texts to find the consistency among participants' choices. Preliminary results of applying Hooper equation support the results of our procedure. Thus, 36.82% of the files in the combined text collection are multi-labeled. Also, the final corpus's LC and LD are found to be 1.4 and 0.175 respectively. Such values are relatively low compared to other multi-labeled data collections as those shown in [178]. Examples of other factors [125] that were reported to affect the results of indexing (here labeling) are indexer's education, experience in indexing, and document length. Their effects, except the "experience in indexing" factor, were investigated.

Further to the work reported in this thesis, it is suggested that there could be advances for further research and development:

1. Further research regarding the developed Arabic corpus is to increase the size of this corpus so that the generality among classes becomes similar in values. Also, although Arabic TC was conducted and tested in this thesis on the developed Arabic corpus with the eight chosen classes, yet further research regarding performing the same classifications on the same corpus but with the eight classes chosen by LACC is required in order to compare the effect of such choices of classes on TC performance. Further research regarding the developed Arabic corpus is to increase its size and investigate the presence of texts with multi labels so that eventually the label cardinality as well as the number of texts and words will be higher in it. This would be carried out through collecting much more texts in MSA from various Arabic websites and performing active learning techniques to further classify these texts by one or more labels among the eight classes.

2. Further research regarding the investigated root extraction techniques, discussed in Chapter 4, is through: a- improvement of the rule-based technique by adding more rules, b- improvement of the proposed and implemented algorithm for correcting irregular words by adding more special cases, c- investigation of improving the weight-based technique through weighting the letters in *sOltmwnyhA* by using fuzzy sets to handle their grouping, and d- further testing the efficiency of proposed root extraction system.

3. Further research to that discussed in Chapters 5 and 6 regarding the investigation of the effect of other VSM representations on TC performance is through investigating other possible combinations of features (i.e. combining for example words with roots, .. etc) on single-label TC performance as well as comparing the effect of variant TFIDF with other term weighting methods on TC performance.

References

1. H.K Al-Ameed, *A Proposed New Model using a light stemmer for increasing the success of search in Arabic terms*, PhD Thesis, 2006. Bradford: University of Bradford, UK.
2. R. Abbès, J. Dichy and M. Hassoun, "Morph-lexical ambiguities in the recognition of written Arabic word-forms, evidence from the DIINAR.1 lexical resource" in proceedings of *The second International Conference on Machine Intelligence ACIDCA-ICMi'05*, 5-7 Nov, Tozeur, Tunisia, 2005.
3. H. Abu-Salem, M. Al-Omari, and M. W. Evens, "Stemming methodologies over individual query words for an Arabic IR system" *Journal of the American Society for Information Science and Technology JASIST*, 50(6): 524-529, 1999. ASIS&T.
4. D. Aha, D. Kibler, and M. Albert, "Instance-Based learning algorithms" in *Machine Learning*, 6: 37-66, 1991. Kluwer Acadmic Publishers. Boston.
5. Z. Ahmed, "Arabic weak verb formulation and computation" in *7th Annual Computational Linguistics in UK CLUK research colloquium*, 6-7 Jan, Birmingham, UK, 2004.
6. M. Al-diabat, "Arabic text categorization using classification rule mining" *Applied Mathematical Sciences*, 6(81): 4033-4046, 2012.
7. N. Al Fe'ar, E. Al Turki, A. Al Zaid, M. Al Duwais, M. Al Sheddi, N. Al Khamees, and N. Al Drees, "E-Classifier: A bilingual email classification system" in proceedings of *International Symposium on IT, ITSim'08*, 26-28 Aug, Kuala Lumpur, Malasyia, 3: 1–4, 2008. IEEE Xplore.
8. S Al-Harbi, A. Almuhareb, A. Al-Thubaity, M. S. Khorsheed and A. Al-Rajeh, "Automatic Arabic TC" *9es Journees internationals d'Analyse statistique des Donnees Textuelles JADT'08*, pp. 77-83, 12-14 March, Lyon, France, 2008. Presses Universitaires de Lyon.
9. M. Aljlayl and O. Frieder, "On Arabic search: Improving the retrieval effectiveness via light stemming approach" in proceedings of *the eleventh ACM International Conference on Information and Knowledge Management CIKM'02*, pp. 340–347, 4-9 November, Illinois Institute of Technology, Mclean, Virginia, USA, 2002. ACM.
10. M. N. Al-Kabi and S. I. Al-Sinjilawi, "A comparative study of the efficiency of different measures to classify Arabic Text" *University. of Sharjah Journal of pure and applied sciences*, 4(2): 13-25, June, 2007. University of Sharjah: Sharjah, UAE.
11. I. A. Al-Kharashi and Martha W. Evens, "Comparing words, stems, and roots as index terms in an Arabic IR system" *Journal of the American Society for Information Science and Technology JASIST*, 45(8): 548-560, 1994. ASIS&T.
12. A. Al-Marghilani, H. Zedan and A. Ayesh, "TM based on Self-Organizing Map method for Arabic-English documents" in proceedings of *the 19th Midwest Artificial Intelligence and Cognitive Science Conference MAICS'08*, pp. 174–181, 12-13 April, Cincinnati, USA, 2008. AAAI.
13. M. Y. Al-Nashashibi, A. Yaghi, and D. Neagu, "An improved root extraction technique for Arabic words" in proceedings of *2nd International Conference on Computer Technology and Development ICCTD 2010*, S. Mahmoud and Z. Lian (Eds.), pp. 264-269, 2-4 Nov, Cairo, Egypt, 2010a. IEEE Xplore.
14. M. Y. Al-Nashashibi, A. Yaghi, and D. Neagu, "Stemming Techniques for Arabic Words: A Comparative Study" in proceedings of *2nd International Conference on Computer Technology and Development ICCTD 2010*, S. Mahmoud and Z. Lian

(Eds.), pp. 270-276, 2-4 Nov, Cairo, Egypt, 2010b. IEEE Xplore.

15. 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*, 5-7 Nov, Ain Shams University, Cairo, Egypt, pp. 436–440, 2006. IEEE.
16. R. Al-Shalabi and Martha Evens, "A Computational Morphology System for Arabic" in proceedings of *The workshop on computational approaches to Semitic languages*, Montreal, Quebec, Canada, pp. 66–72, 1998. ACL.
17. R. Al-Shalabi, G. Kanaan and H. Al-Serhan, "New approach for extracting Arabic roots" in proceedings of *2003 International Arab conference on Information Technology (ACIT'2003)*, pp. 42-59, 20-23 Dec, Alexandria, Egypt, 2003. Arab Academy for Science and Technology and Maritime Transport, Egypt.
18. R. Al-Shalabi, "Pattern-based stemmer for finding Arabic roots" *Information Technology Journal*, 4(1), pp. 38-43, 2005.
19. R. Al-Shalabi, G. Kanaan and M. H. Gharaibeh, "Arabic TC using kNN algorithm" in proceedings of *The 4th Int Multiconference on Computer Science and Information Technology CSIT'06*, pp. 1–9, April, Amman, Jordan, 2006. Applied Science Private University, Amman, Jordan.
20. R. Al-Shalabi and R. Obeidat, "Improving KNN Arabic TC with n-grams based document indexing" in proceedings of *the 6th International Conference on Informatics and Systems INFOS'08*, pp. 108-112, 27-29 March, Cairo, Egypt, 2008. Faculty of Computers and Information-Cairo University: Cairo, Egypt.
21. E. Al-Shammari and J. Lin, "A novel Arabic lemmatization algorithm" in proceedings of *2nd Workshop on Analytics for Noisy unstructured text Data AND'08*, pp. 113-118, 24 July, Singapore, 2008a. ACM.
22. E. Al-Shammari and J. Lin, "Towards an Error-Free Arabic Stemming" in proceedings of *2nd ACM International Workshop on Improving Non English Web Searching iNEWS'08*, pp. 9-15, 30 Oct, Napa Valley, California, USA, 2008b. ACM.
23. E. Al-Shawakfa, A. Al-Badarneh, S. Shatnawi, K. Al-Raba'ah, and B. Bani-Ismail, "A comparison study of some Arabic root finding algorithms" *Journal of the American Society for Information Science and Technology JASIST*, 61(5): 1015-1024, 2010. ASIS&T.
24. I.A. Al-Sughaiyer and I.A Al-Kharashi, "Arabic Morphology Analysis Techniques: A Comprehensive Survey" *Journal of the American Society for Information Science and Technology JASIST*, 55(3): pp. 189–213, Feb. 2004. ASIS&T.
25 L. Al-Sulaiti Home page, URL: http://www.comp.leeds.ac.uk/eric/latifa/research.htm, 2009. University of Leeds, UK.
26. L. Al-Sulaiti and E. Atwell, "The design of a corpus of Contemporary Arabic" *International Journal of Corpus Linguistics*, 11(1): 1–36, 2006. John Benjamins Publishing Company.
27. A. Al-Zoghby, A. S. Eldin, N. A. Ismail, and T. Hamza, "Mining Arabic text using soft-matching association rules" in proceedings of *International Conference on Computer Engineering and systems, ICCES '07*, pp. 421–426, 27-29 Nov, Cairo, Egypt, 2007. IEEE Xplore.
28. M. Attia, *A large-scale computational processor of the Arabic morphology, and applications*, M.Sc. Thesis, January 2000. Faculty of Engineering- Cairo University: Giza, Egypt.
29. M. Attia, "Developing robust Arabic morphological transducer using Finite state

technology", in *8th Annual Computational Linguistics in UK CLUK Research Colloquium*, Manchester, UK, 2005.

30. M. Attia, "Arabic tokenization system" in proceedings of the *workshop on Computational Approaches to Semitic Languages: Common Issues and Resources (Semitic'07)*, Prague, Czech Republic, pp. 65-72, June, 2007. ACL: Stroudburg, PA, USA.
31. Imam Mohammed Ibn Abi Baker Ar-Rhazi, *Mukhtar us-Sihah*. 1986. Beirut: Librairie du Liban Publishers. (in Arabic).
32. C. Apté, F.J. Damerau, and S.M. Weiss, "Automated learning of decision rules for text categorization" *ACM Transactions on Information Systems*, 12(3): 233-251, July, 1994. ACM.
33. H. Bayyomee, Kh. Kolfat, and A. Al-Shafe'e, *Lexicon for Arabic Verbs morphology*. 1989. Cairo: Dar Ilias Modern Publishing Comp. (in Arabic).
34. M. J. Bawaneh, M. S. Alkoffash and A. I. Al Rabea, "Arabic TC using k-NN and NB" *Journal of Computer Science*, 4(7): 600–605, 2008. Science Publications.
35. K.R. Beesley, "Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001" in proceedings of *ARABIC NLP Workshop: Status and Prospects ACL-EACL2001*, pp. 1–8, 6 July, Toulouse, France, 2001. ACL.
36. A. Boudlal, R. Belahbib, A. Lakhouaja, A. Mazroui, A. Meziane, and M. Ould Abdallahi Ould Bebah, "A markovian approach for Arabic root extraction" in proceedings of *The international Arab conference on information technology ACIT 2008*, University of Sfax, Tunisia, Dec. 16-18, 2008.
37. T. Brants, F. Chen, and A. Farahat, "Arabic document analysis" in proceedings of *workshop Arabic Language Resources and Evaluation Conference LREC'02*, Las Palmas, Spain, 2002.
38. L. Breiman, "Random Forests" *Machine Learning*, 45(1): 5-32, Oct 2001. Kluwer Academic Publishers, The Netherlands.
39. L. Breiman, "Bagging predictors" *Machine Learning*, 24(2): 123-140, 1996. Kluwer Academic Publishers, The Netherlands.
40. L. Breiman, J.H. Freidman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, 1984. Wadsworth Int Group, Belmont, California.
41. T. Buckwalter, "Buckwalter Arabic morphological analyzer Version 1.0", *Linguistic Data Consortium*, Philadelphia, 2002, available online at URL: http://www.qamus.org/. [Accessed 22/12/2010].
42. L. Cahill, "A syllable approach to verbal morphology in Arabic" in proceedings of *Language Resources and Evaluation LREC workshop on Semitic languages*, pp. 19-26, 17th May, Malta, 2010.
43. V. Chakraborty, M. Vasarhelyi, and V. Chiu, "Automatic classification of accounting literature" in proceedings of *19th Annual Strategic and Emerging Technologies Research workshop*, San Francisco, CA, USA, 31 July, 2010.
44. A. Chen and F. Gey, "Building an Arabic Stemmer for Information Retrieval" in proceedings of *NIST Special Publication: The Eleventh Text REtrieval Conference TREC'02*, Voorhees, E.M. & Harman, D.K. (Eds.), 19-22 Nov, Gaithersburg, Maryland, 2002. NIST: Gaithersburg, Maryland.
45. Long-Sheng Chen and Chai-Wei Chang, "A new term weighting method by introducing class information for sentiment classification of textual data" in proceedings of *International MultiConference of Engineers and Computer Scientists Vol 1 IMECS'11*, 16-18 March, Hong Kong, 2011.
46. Ding-An Chiang, Huan-Chao Keh, Hui-Hua Huang, and D. Chyr, "The Chinese text

categorization system with association rule and category priority" *Expert Systems with Applications*, 35: 102-110, 2008. ElSEVIER.

47. William W. Cohen, "Fast effective rule induction" in proceedings of *the twelfth international conference on Machine Learning ML95*, pp. 115-123, 9-12 July, Tahoe City, California, USA, 1995. Morgan Kaufmann Publishers, San Francisco, CA, USA.
48. K. Darwish, D. Doermann, R. Jones, D.W. Oard, and M. Rautiainen, "TREC-10 Experiments at University of Maryland CLIR and Video" in proceedings of *NIST Special Publication: The Tenth Text REtrieval Conference TREC'01*, pp. 549–562, USA. 2001. NIST: Gaithersburg, Maryland.
49. K. Darwish, "Building a Shallow Arabic Morphological Analyzer in one day" in proceedings of *workshop on computational approaches to Semitic languages ACL'02*, Philadelphia, PA, USA, 2002a. ACL.
50. K. Darwish, "Al-stem: A Light Arabic Stemmer", As part of Dissertation Work *Probabilistic Methods for Searching OCR-Degraded Arabic Text*, University of Maryland, College Park, 2002b.
51. K. Darwish, H. Hassan, and O. Emam, "Examining the effect of improved context sensitive morphology information retrieval" in proceedings of *of the ACL workshop on Computational approaches to Semitic languages*, pp. 25-30, June, Ann Arbor, MI, USA, 2005. ACL.
52. E. Daya, D. Roth and S. Wintner, "Identifying Semitic Roots: Machine Learning with Linguistic Constraints" *Computational Linguistics*, 34(3): 429-448, 2008. ACL.
53. C. Deisy, M. Gowri, S. Baskar, S.M.A. Kalairasi and N. Ramraj, "A Novel term weighting scheme MIDF for Text Categorization" *Journal of Engineering Science and Technology*, 5(1): 94–107, March, 2010. School of Engineering, Taylor's University College.
54. A. N. DE Roeck and W. Al-Fares, "A Morphologically sensitive clustering algorithm for identifying Arabic roots" in proceedings of *38th Annual meeting of ACL*, Hong Kong, pp. 199–206, 2000. ACL.
55. M. Diab, K. Hacioglu and D. Jurafsky, "Automatic tagging of Arabic text: from raw text to base phrase chunks" in proceedings of *5th meeting of the North American chapter of the Association for Computational Linguistics/Human Language Technologies conference (HLT-NAACL 04)*, pp. 149–156, Boston, MA, USA, 2-7 May, 2004. ACL.
56. J. Dichy, A. Braham, S. Ghazali, M. Hassoun, "La Base de connaissances linguistiques DIINAR.1 (DIctionnaire INformatise de l'Arabe, version 1)"., in *Colloque international sur le traitement automatique de l'arabe – proceedings of the International Symposium on The Processing of Arabic*, A. Braham (Ed.), Université de la Manouba, Tunis (en Arabe, Français et Anglais) 18-20 April, 2002.
57. J. Dichy, and A. Farghaly, "Roots and patterns vs. stems plus grammar-lexis specifications: on what basis should a multilingual lexical database centred on Arabic be built?" in proceedings of *workshop on Machine Translation for Semitic languages MT-summit IX*, 23 Sept, New Orleans, USA, 2003.
58. Chuong B. Do and Andrew Y. Ng, "Transfer learning for text classification" in proceedings of *Advances in Neural Information Processing Systems NIPS*, 2006.
59. L. Dong, E. Frank, and S. Kramer, "Ensembles of balanced nested dichotomies for multiclass problems" in proceedings of *9th European conference on Principles and Practice of Knowledge Discovery in Databases: PKDD'05*, LNCS -3721, pp 84-95,

Porto, Portugal, October 3-7, 2005. Springer.

60. R.M. Duwairi, "Machine Learning for Arabic Text Categorization" *Journal of the American Society for Information Science and Technology JASIST*, 57(8): 1005–1010, April, 2006. ASIS&T.
61. R. Duwairi, "Arabic Text Categorization" in *The International Journal of Information Technology*, 4(2): 125–131, March, 2007. ASIS&T.
62. R. M. Duwairi, M. N. Al-Refai and N. Khasawneh, "Stemming versus light stemming as feature selection techniques for Arabic text" in proceedings of *4th International Conference on Innovations in Information Technology, 2007. IIT '07*, pp. 446–450, 18-20 Nov, Dubia, UAE, 2008. IEEE Xplore.
63. R. M. Duwairi, M. N. Al-Refai and N. Khasawneh, "Feature Reduction Techniques for Arabic TC" *Journal of the American Society for Information Science and Technology*, 60(11): 2347–2352, Nov, 2009. ASIS&T.
64. E. Eibeed, "Suggestions for the best way to classify articles in Knol-Google" (*in Arabic*), URL: http://knol.google.com/k/ [last accessed 16/2/2011].
65. J. Eisenstein and R. Davis, "Visual and linguistic information in gesture classification" in proceedings of *the 6th International Conference on Multimodal Interfaces ICMI'04*, pp. 113-120, 13-15 Oct, Pennsylvania, USA, 2004. ACM.
66. S. El-Beltagy and A. Rafea, "KP-Miner: A Key phrase Extraction System for English and Arabic documents" in *Information Systems Journal*, 34(1): 132-144, 2009. Elsevier.
67. A. El-Dahdah, *A Dictionary of Arabic Grammar in Charts and Tables*, 2008. Beirut: Librairie du Liban Publishers. Revised by: Dr. GM Abdul-Massih (in Arabic).
68. M. El-Haj, U. Kruschwitz, and C. Fox, "Experimenting with automatic text summarization for Arabic" in proceedings of *Human Language Technology LTC-09*, 2009. Revised selected papers, volume 6562 of Lecture Notes in Computer Science, pp. 490-499. Springer: Berlin, Heidelberg, 2011.
69. M. El-Haj, U. Kruschwitz, and C. Fox, "Using mechanical turk to create a corpus of Arabic summaries", in proceedings of the *Language Resources (LRs) and Human Language Technologies (HLT) for Semitic languages workshop held in conjunction with the 7th International Language Resources and Evaluation Conference (LREC'10)*, pp. 36-39, Valletta, Malta, 2010.
70. A. M. El-Halees, "Arabic TC using maximum entropy" *The Islamic University Journal (Series of Natural studies and engineering)*, 15(1): 157–167, 2007. The Islamic University, Gaza, Palestine. URL: http://www.iugaza.edu.ps/ar/periodical/articles/natural15(1)2007pp157-167.pdf [last accessed 22/12/2010].
71. M. El-Kourdi, A. Bensaid, and T. Rachidi, "Automatic Arabic documents categorization based on the Naive Bayes algorithm" in proceedings of *the workshop on Computational Approaches to Arabic Script-Based Languages, The 20th International Conference on Computational Linguistics COLING'04*, pp. 51-58, 23-27 August, University of Geneva, Switzerland, 2004. ACL. URL: http://acl.ldc.upenn.edu/coling2004/W5/index.html [last accessed 22/12/2010].
72. T. A. El-Sadany and M. A. Hashish, "An Arabic morphological system" *IBM Systems Journal*, 28(4): 600-612, 1989. IBM, IEEE Xplore.
73. Eibe Frank and M. Hall, "A simple approach to ordinal classification" in proceedings of *12th European Conference on Machine Learning ECML'01*, pp. 145-156, Freiburg, Germany, September 5-7, 2001. Berlin: Springer-Verlag.
74. Eibe Frank and S. Kramer, "Ensembles of nested dichotomies for multiclass problems" in proceedings of *21st International Conference on Machine Learning*

ICML'04, pp. 39, Banff, Alberta, Canada, 2004.

75. Eibe Frank and I. H. Witten, "Generating accurate rule sets without global optimization" in proceedings of *fifteenth International conference on Machine Learning*, Shalvik, J. (Ed.), pp. 152-160, 24-27, Madison, Wisconsin, USA, 1998. Morgan Kaufmann Publishers, San Francisco, CA, USA.
76. Eibe Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten, "Using model trees for classification" *Machine Learning*, 32: 63-76, 1998. Kluwer Academic Publishers, The Netherlands.
77. J. Friedman, T. Hastie, R. Tibshirani, "Additive logistic regression: A statistical view of boosting" *Annals of Statistics*, 28(2): 337-407, 2000.
78. Y. Freund and R.E. Schapire, "Experiments with a new Boosting algorithm" in proceedings of *13th International Conference on Machine Learning (ICML-96)*, p. 148-156, Bari, Italy, 3-6 July, 1996.
79. J. Fürnkranz, T. Mitchell, and E. Riloff, "A case study in using linguistic phrases for text categorization on the WWW", *AAAI technical report W5-98-05*, 1998. AAAI.
80. B. Gaines and P. Compton, "Induction of Ripple-DOwn Rules applied to modeling large databases" *Journal of Intelligent Information Systems* November, 5(3): 211-228, 1995. Springer.
81. L. Galavotti, F. Sebastiani and M. Simi, "Experiments on the use of feature selection and negative evidence in automated TC" in proceedings of *4th European conference on research and advanced technology for Digital Libraries ECDL-00*, pp. 59-68, Sept, Lisbon, Portugal, 2000. Springer-Verlag: Berlin, Heidelberg.
82. J. Gama, "Functional Trees" *Machine Learning*, 55(3): 219-250, 2004. Kluwer Academic Publishers, The Netherlands.
83. A. Gelbukh and O. Kolesnikova, "Supervised learning algorithms evaluation on recognizing semantic types of Spanish verb-noun collocations" *Computación y Sistemas*, 16(3): 297-308, 2012.
84. S. Ghwanmeh, R. Al-Shalabi, G. Kanaan, K. Khanfar, and S. Rabab'ah, "An algorithm for extracting the root for the Arabic language" in proceedings of *5th International Business Information Management Association Conference IBIMA on the Internet and Information Technology in Modern Organizations*, 13-15 Dec, Cairo, Egypt, 2005.
85. S.H. Ghwanmeh, "Applying Clustering of Hierarchical K-means-like Algorithm on Arabic Language" *Inernational. Journal of Information Technology*, 3(3): 168–172, July, 2006. Singapore Computer Society.
86. S. Ghwanmeh, G. Kanaan, R. Al-Shalabi and A. Ababneh, "Enhanced Arabic IR system based on Arabic TC" in proceedings of *4th International conference on Innovations in Information Technology, 2007, IIT'07*, pp. 461–465, 18-20 Nov, Dubai, UAE, 2008. IEEE Xplore.
87. M. Grobelnik and D. Mladenic "Tutorial on Text Mining" 2004. Available online at URL: http://eprints.pascal-network.org/archive/00000017/01/Tutorial_Marko.pdf. [Accessed 27/12/2010].
88. N. Habash "Large Scale Lexeme Based Arabic Morphological Generation", *in* proceedings of *Session Tritement Automatique de l'Arabe, Les conférences JEP ("Journées d'Etude sur la Parole") et TALN ("Traitement Automatique des Langues Naturelles") JEP-TALN*, Fez, pp.1–6, April, 20, 2004.
89. W. Hadi, M. Salam and J. Al-Widian, "Performance of NB and SVM classifiers in Islamic Arabic data" in proceedings of *The International Conference on Intelligent Semantic Web – Services and Applications ISWSA'10*, 14-16 June, Amman, Jordan, 2010. ACM.

90. W. Hadi, F. Thabtah and G. Kanaan, "NB and K-nearest neighbor to categorize Arabic text data" in proceedings of *The European multidisciplinary society for modeling and simulation technology ESM'08*, pp. 196-200, 27 – 29 Oct, Universite du Havre, Le Havre, France, 2008. EUROSIS-ETI.
91. Mark Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, "The WEKA data mining software: an update" *Special Interest Group Knowledge Discovery and Data Mining SIGKDD explorations*, 11(1), 2009.
92. J. Han and M. Kamber, *Data Mining Concepts and Techniques*, 2nd edition. 2006. San Francisco: Morgan Kaufmann Publishers (an imprint of ELSEVIER), CA, USA.
93. F. Harrag, A.M. Al-Salman, and M. Ben Mohammed, "A comparative study of neural networks architectures on Arabic text categorization using feature extraction" in proceedings of *International Conference on Machine and Web Intelligence ICMWI'10*, 3-5 Oct, Algiers, Algeria, pp. 102-107, 2010. IEEE.
94. F. Harrag, E. El-Qawasmeh and P. Pichappan, "Improving Arabic TC using decision trees" in proceedings of *First International conference on Networked Digital Technologies NDT'09*, pp. 110–115, 28 – 31 July, Ostrava, The Czech Republic, 2009a. IEEE Xplore.
95. F. Harrag and E. El-Qawasmeh, "Neural Network for Arabic TC" in proceedings of *2nd International conference on Application of Digital information and Web technologies, ICADIWT'09*, pp. 778–783, 4-6 August, London, UK. 2009b. IEEE Xplore.
96. J.A. Haywood and H.M. Nahmad, *A New Arabic Grammar of Written Language*. 1998. London: Lund Humphries Publishers, UK.
97. I. Hmeidi, G. Kanaan, and M. Evens, "Design and Implementation for Information Retrieval with Arabic Documents" *Journal of the American Society for Information Science and Technology JASIST*, 48(10): 867–881, 1997. ASIS&T.
98. I. Hmeidi, B. Hawasashin and E. El-Qawasmeh, "Performance of KNN and SVM classifiers on full word Arabic articles" *Advanced Engineering Informatics* 22: 106-111, August, 2008. ELSEVIER.
99. T.K. Ho, "The random subspace method for constructing decision forests" *IEEE transactions on pattern analysis and machine intelligence*, 20(8): 832-844, 1998. IEEE.
100. R.S. Hooper, "Indexer consistency tests-Origin" *Measurements, results and utilization*, 1965. IBM, Bethesda.
101. G. Holmes, B. Pfahringer, R. Kirkby, E. Frank and M. Hall, "Multiclass alternating decision trees" in proceedings of *13th European Conference on Machine Learning ECML'02*, pp. 161-172, 19-23 Aug, Hilsinki, Finland, 2002. LNCS 2430, Springer.
102. R.C. Holte, "Very simple classification rules perform well on most commonly used datasets" in *Machine Learning*, 11, p. 63-91, 1993.
103. A. Hotho, A. Nürnberger and G. Paaβ, "A brief survey of Text Mining" in *LDV_Forum*, 20(1): 19-62, May, 2005. Text MiningImpressum. Found at URL: http://www.jlcl.org/2005_Heft1/19-62_HothoNuernbergerPaass.pdf . [last accessed 27/12/2010].
104. M. Hussien, F. Olayah, M. Al-dwan, and A. Shamsan, "Arabic text classification using SMO, Naive Bayesian, J48 algorithms" *International Journal of Research and Reviews in Applied Sciences IJRRAS*, 9(2): 306-316, Nov 2011.
105. T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features" in proceedings of *10th European Conference on Machine Learning ECML'98*, pp. 137-142, 21-23 April, Chemnitz, Germany, LNCS, vol

1398, 1998. Springer.
106. G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers" in proceedings of *eleventh conference on uncertainty in artificial intelligence*, pp. 338-345, 18-20 August, Montreal, Quebec, Canada, 1995. Morgan Kaufmann Publishers, San Mateo.
107. G. Kanaan, R. Al-Shalabi, and A. Al-Akhras, "kNN Arabic TC using IG Feature selection" in proceedings of *the 4th international multi-conference on Computer Science and Information Technology CSIT'06*, pp. 1–9, April, Amman, Jordan, 2006. Applied Science Private University, Amman, Jordan.
108. G. Kanaan, R. Al-Shalabi, S. Ghwanmeh and H. Al-Ma'adeed, "A comparison of TC techniques applied to Arabic text" *Journal of the American Society for Information Science and Technology*, 60(9): 1836–1844, July, 2009a. ASIS&T.
109. G. Kanaan, M. Yaseen, R. Al-Shalabi, B. Al-Sarayreh and A. B. Mustafa, "Using EM for Text Classification on Arabic Documents" in proceedings of *the Second International Conference on Arabic Language Resources and Tools*, pp. 9-11, 22-23 April, Cairo, Egypt, 2009b. The MEDAR Consortium.
110. S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design" *Technical report CD-99-14, control division, Dept. of Mechanical and Production Engineering*, National University of Singapore, 1999. Also published in Neural Computation, 13(3): 637-649, 2001.
111. S. Khoja, "Stemming Arabic Text" 1999. Available online at URL: http://zeus.cs.pacificu.edu/shereen/research.htm#stemming [accessed 27/12/2010].
112. L. Khreisat, "A machine learning approach for Arabic text classification using N-gram frequency statistics" *Journal of Informatics* 3: 72-77, Nov, 2009. ELSEVIER.
113. H. Kim and W-Y. Loh, "Classification trees with unbiased multi way splits" *Journal of the American Statistical Association*, 96: 589-604, 2001.
114. R. Kohavi, "The power of decision tables" in proceedings of *European conference on machine learning ECML'95*, pp. 174-189, Crete, Greece, April 25-27, 1995.
115. Ron Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid" in proceedings of *2nd International Conference on Knowledge Discovery and Data Mining KDDM'96*, pp. 202-207, Portland, OR, August 2-4, 1996.
116. Q. Kuang and X. Xu, "Improvement and application of TF*IDF method based on text classification" in proceedings of *2010 International conference on Internet Technology and Applications ITAP'10*, 21-23 Aug, Wuhan, China, 2010. IEEE Xplore.
117. A. Kyriakopoulou, "Text classification aided by clustering: a literature review" *Chapter 14 in Tools in Artificial Intelligence*, Paula Fritzsche (Ed.), August, pp. 233-252, 2008. InTech. URL: http://www.intechopen.com/articles/show/title/text_classification_aided_by_clustering__a_literature_review [accessed 27/12/2010].
118. N. Landwehr, *Logistic model trees*, Diploma thesis, University of Freiburg, Freiburg, Germany, 2003
119. N. Landwehr, M. Hall, and E. Frank, "Logistic Model Trees" *Machine Learning*, 95(1-2):161-205, 2005.
120. L.S. Larkey and Margaret E. Connell, "Arabic Information Retrieval: at UMass in TREC-10" in proceedings of *Text REtrieval Conference TREC'10, NIST*, Nov, pp. 562–570, 2001. NIST: Gaithersburg, Maryland.
121. L.S. Larkey, L. Ballesteros, M.E. Connell, "Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis" in proceedings of *Special Interest Group on Information Retrevial SIGIR'02*, pp. 275–282, August,

Tampere, Finland, 2002. ACM.
122. L.S. Larkey, Lisa Ballesteros, and Margaret E. Connell, "Light stemming for Arabic Information retrieval", *Arabic computational morphology knowledge-based and empirical methods, text, speech and language technology series*, Abdelhadi Soudi, Antal van den Bosch and Gunter Neumann (Eds.), Vol. 38, Part IV, pp. 221–243, 2007. The Netherlands: Springer.
123. S. le Cessie, J.C. van Houwelingen, "Ridge Estimators in Logistic Regression" *Applied Statistics*, 41(1): 191-201, 1992.
124. Young-Suk Lee, K. Papineini, S. Roukos, O. Emam, and H. Hassan, "Language model based Arabic word segmentation", in proceedings of *the 41st annual meeting of the Association for Computational Linguistics*, Vol. 1, pp. 399-406, July, Sapporo, Japan, 2003. ACL.
125. L.E. Leonard, "Inter-indexer consistency studies, 1954-1975: a review of the literature and summary of study results" in *The Library of University of Illinois at Urbana-Champaign (occasional papers)*, Dec, no. 131, 1977.
126. D. Lewis, Reuters-21578 corpus, 2004. Available at: http://www.daviddlewis.com/resources/testcollections/reuters21578/ [last accessed 1/5/2011].
127. D. Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A new benchmark collection for Text Categorization research" *Journal of Machine Learning Research*, 5: 361-397, 2004.
128. Ying Liu, H.T. Loh, and A. Sun, "Imbalanced text classification: a term weighting approach" *Expert Systems with Applications*, 36: 690-701, 2009. ElSEVIER.
129. C.D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. 1999. Massachusetts: MIT press, USA.
130. Brent Martin, "Instance-Based learning: Nearest Neighbour with Generalization" M.Sc. Thesis, 1995. Hamilton, New Zealand: University of Waikato.
131. E. Marsi, A. van den Bosch and A. Soudi "Memory-based morphological analysis generation and part-of-speech tagging of Arabic" in proceedings of *ACL workshop on computational approaches to Semitic languages*, Ann Arbor, Michigan, pp. 1-8, 2005. ACL.
132. A. McCallum and K. Nigam, "A comparison of event models for Naive Bayes text classification" in proceedings of *AAAI-98 workshop on learning for text categorization*, pp. 41-48, 26-27 July, Madison, Wisconsin, USA, 1998
133. P. Melville and R.J. Mooney, "Constructing diverse classifier ensembles using artificial training examples" in proceedings of *18th International Joint Conference on Artificial Intelligence IJCAI '03*, pp. 505-510, Acapulco, Mexico, 9-15 Aug, 2003.
134. P. Melville and R.J. Mooney, "Creating diversity in ensembles using artificial data" *Information Fusion: Special issue on diversity in multi classifier systems*, 6(1): 99-111, 2005. ELSEVIER.
135. A. M. Mesleh, "Chi Square Feature Extraction based SVMs Arabic language text categorization system" *Journal Computer Science*, 3(6): 430-435, 2007. Science Publications.
136. A. M. Mesleh, "SVM based Arabic language TC system: feature selection comparative study" *International Joint Conferences on Computer, Information and Systems Sciences and Engineering CISSE'07*, pp. 11–16, 3 – 12 Dec, University of Bridgeport, USA. Advances in Computer and Information Sciences and Engineering, 2008a. Springer Science + Business Media B.V.2008.
137. A. M. Mesleh and G. Kanaan, "SVM TC system: using Ant colony optimization based feature subset selection" in proceedings of *International conference on computer*

engineering and systems ICCES'08, pp. 143–148, 25-27 Nov, Cairo, Egypt, 2008b. IEEE Xplore.

138. R. Milhalcea and P. Tarau, "A language independent algorithm for single and multiple document summarization" in proceedings of *second international joint conference on natural language processing IJCNLP'05*, 11 – 13 Oct, Jejo Island, Korea, 2005.
139. A. Moschitti and R. Basili, "Complex linguistic features for text classification: a comprehensive study" *Advances in Information Retreival, LNCS*, 2997:181-196, 2004. Springer.
140. H. Ng, W.B. Goh, and K.L. Low, "Feature term selection, perceptron learning, and a usability case study for TC" in proceedings of *Special Interest Group on Information Retrevial SIGIR'97, 20th ACM International conference on Research and Development in IF*, pp. 67-73, December, Philadelphia, PA, 1997. ACM.
141. H. M. Noaman, S. Elmougy, A. Ghoneim, and T. Hamza, "NB classifier based Arabic TC" in proceedings of *The 7th International conference on Informatics and Systems, INFOS '10*, pp. 1–5, 28-30 March, Cairo, Egypt, 2010. IEEE Xplore.
142. J. Platt, "Fast training of SVM using Sequential Minimal Optimization" *Advances in kernel methods - support vector learning*, 1998. MIT press.
143. J.R. Quinlan, "Improved use of continuous attributes in C4.5" *Journal of Artificial Intelligence Research*, 4:77-90, 1996. Journal of Artificial Intelligence Research and Morgan Kaufmann Publishers.
144. J.R. Quinlan, "Learning with continuous classes" in proceedings of *Australian Joint Conference on Artificial Intelligence AJCAI'92*, pp. 343-348, 1992. World Scientific, Singapore.
145. J.R. Quinlan, "Simplifying decision trees" *International Journal of Man-Machine Studies*, 27: 221-234, 1987.
146. S. Raheel and J. Dichy, "Reducing data sparsity in a language dependent automatic classification of Arabic documents" in proceedings of *7th Conference of the French chapter of ISKO'09*, 24-26 June, Lyon, France, 2009a.
147. S. Raheel, J. Dichy, and M. Hassoun, "The Automatic Categorization of Arabic Documents by Boosting Decision Trees" in proceedings of *5th International Conference on Signal-Image Technology and Internet-based Systems SITIS'09*, pp. 294-301, 29 Nov – 4 Dec, Marrakech, Morocco, 2009b. IEEE Xplore. Found from URL: http://www.raheels.net/includes/RAHEEL_SAEED_SITIS2009.pdf. [Accessed 27/12/2010].
148. S. Raheel and J. Dichy, "An empirical study on the feature's type effect on the automatic classification of Arabic documents" in proceedings of *Conference on Intelligent Text Processing and Computational Linguistics CICLing'10, LNCS 6008*, pp. 675–686, 2010. Springer- Verlag: Berlin, Heidelberg.
149. J.D.M. Rennie, L. Shih, J. Teevan, and D. Karger, "Tackling the poor assumptions of Naive Bayes text classifiers" in proceedings of *20th International Conference on Machine Learning ICML'03*, pp. 616-623, 21 -24 Aug, Washington DC, 2003. AAAI press.
150. J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, "Rotation forest: A new classifier ensemble method" *IEEE transactions on pattern analysis and machine intelligence*, 28(10): 1619-1630, 2006. IEEE.
151. M. Rogati, S. McCarley and Y. Yang, "Unsupervised learning of Arabic stemming using parallel corpus" in proceedings of *41st Annual meeting of ACL*, Sapporo, Japan, pp. 391–398, 2003. ACL: Morristown, NJ, USA.
152. T. G. Rose, M. Stevenson and M. Whitehead, "The Reuters corpus volume I – from yesterday's news to tomorrow's language resources" in proceedings of *Third*

International Conference on Language Resources and Evaluation LREC'02, pp.827-833, 29-31 May, 2002. http://about.reuters.com/researchandstandards/corpus/LREC_camera_ready.pdf [last accessed 22/2/2011].

153. Sakhr software company. 2004. URL: http://www.textmining.sakhr.com/ [accessed 27/12/2010].
154. D. Said, N. M. Wanas, N. Darwish, and N. Hegazy, "A study of text preprocessing tools for Arabic Text Categorization" in proceedings of *the 2nd International Conference on Arabic Language Resources and Tools MEDAR'09*, pp. 230-236, 22-23 April, Cairo, Egypt, 2009.
155. G. Salton, A. Wong, and C. Yang, "A Vector Space Model for automatic indexing" *Communication ACM*, 18, 11, pp. 613-620, Nov, 1975. ACM. Also, reprinted in Spark Jones and Willett [1997], pp. 273-280.
156. G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval" *Information Processing and Management*, 24(5): 513-523, January, 1988. Pergamon press, UK.
157. S. Salzberg, "A nearest hyper rectangle learning method" *Machine Learning*, 6: 277-309, 1991.
158. M. Sanderson, "Word Sense Disambiguation and Information Retrieval" in proceedings *of 17thAnual International ACM SIGIR conference*, pp. 142-151, Dublin, Ireland, 1994. Springer-Verlag: New York, USA.
159. H. Sawaf, J. Zaplo and H. Ney, "Statistical classification methods for Arabic news articles" in proceedings of *workshop on Arabic natural language processing, ACL'01*, 6 July, Toulouse, France, 2001. ACL: Morriston, NJ, USA. Online: http://www.elsnet.org/acl2001-arabic.html [last accessed 27/12/2010].
160. M. Sawalha and E. Atwell, "Comparative Evaluation of Arabic Language Morphological Analyzers and Stemmers" in proceedings of *International Conference on Computational Linguistics COLING'08: Companion volume – Posters and Demonstrations*, pp. 107–110, 18-22 August, Manchester, UK, 2008. ACL: Morriston, NJ, USA.
161. F. Sebastiani, "Machine Learning in Automated Text Categorization" *ACM Computing Survey*, 34(1): 1–47, March, 2002. ACM.
162. A. Selamat and Ng C. Ching, "Arabic script documents language identifications using Fuzzy ART" in proceedings of *Second Asia International Conference on Modeling and Simulation*, pp. 528-533, Kuala Lumpur, Malaysia, 13-15 May, 2008. IEEE Computer society.
163. S. Sharoff, "Towards basic categories for describing properties of texts in a corpus", in proceedings of *Language Resources and Evaluation Conference LREC04* (volume V), M. T. Lino, M. F. Xavier, F. Ferreira, R. Costa, R. Silva, C. Pereira, F. Cervalho, M. Lopes, M. Catarino & S. Barros (Eds.), pp. 1743-1746, Lisbon, Portugal, 2004.
164. H. Shi, *Best-first decision tree learning*, Master thesis, University of Waikato, Hamilton, NZ. 2007.
165. J. Sinclair, "Preliminary recommendations on text typology" *Eagles document EAG-TCWG-TTYP/P*, URL: http://ilc.cnr.it/EAGLES96/texttyp/texttyp.html, 1996, [last accessed 16/2/2011].
166. A. Singhal, C. Buckley and M. Mitra, "Pivoted document length normalization" in proceedings of *19th annual International ACM SIGIR conference on Research and Development in Information Retrieval SIGIR 96*, pp. 21-29, 18-22 August, Zurich, Switzerland, 1996a. ACM.

167. A. Singhal, G. Salton, M. Mitra, and C. Buckley, "Document length normalization" *Information Processing and Management*, 32(5): 619-633, Sept, 1996b. ELSEVIER.
168. N. Snider and M. Diab, "Unsupervised induction of Modern Standard Arabic verb classes using syntactic frames and LSA" in proceedings of *International Conference on Computational Linguistics - Association of Computational Linguistics COLING/ACL'06 main conference poster sessions*, pp.795–802, Sydney, July, 2006. ACL.
169. R. Sonbol, N. Ghneim, and M.S. Desouki, "Arabic morphological analysis: a new approach" in proceedings of *3rd International Conference on Information and Communication Technologies: from Theory to Applications ICTTA'08*, pp. 1-6, 7-11 April, Damascus, Syria, 2008. NJ:IEEE.
170. Marc Sumner, Eibe Frank, and Mark Hall, "Speeding up Logistic Model Tree Induction" in proceedings of *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp 675-683, Porto, Portugal, October 3-7, 2005.
171. M. M., Syiam, Z. T. Fayed and M. B. Habib, "An intelligent system for Arabic TC" *International Journal of Intelligent Computing and Information Sciences IJICIS*, 6(1): 1–19, 2006. World Sci. Publ. Co.
172. K. Taghva, R. Elkhouri, and J. Coombs, "Arabic stemming without a root dictionary" in proceedings of *the International Conference on Information Technology: Coding and Computing ITCC'05*, Vol. 1, pp. 152-157, 4-6 April, Las Vegas, NV, USA, 2005. NJ:IEEE.
173. F. Thabtah, W. M. Hadi and G. Al-Shammare, "VSMs with k-nearest neighbor to categorize Arabic text data" in proceedings of *the World Congress on Engineering and Computer Science WCECS'08*, pp. 778–781, 22 – 24 Oct, San Francisco, USA, 2008. IAENG International Association of Engineers, Hong Kong.
174. F. Thabtah, M. Eljinini, M. Zamzeer and W. M. Hadi, "Naïve Bayesian based on Chi square to categorize Arabic data" *Communications of the IBIMA*, 10:158-163, 2009.
175. F. Thabtah, Omar Gharaibeh and H. Abdeljaber, "Comparison of Rule-based classification techniques for the Arabic textual data" in proceedings of *4th International Symposium on Innovation in Information and Communication Technology ISIICT'11*, 29 Nov-1 Dec, Philadelphia Univ, Amman, Jordan, pp.105-111, 2011. IEEE.
176. T. Theeramunkong and V. Lertnattee, "Improving centroid-based text classification using term-distribution-based weighting system and clustering" in proceedings of *2nd International Symposium on Communication and Information Technology ISCIT'01*, pp. 1167-1182, Nov, Cheingmai, Thailand, 2001.
177. K.M. Ting and I.H. Witten, "Stacking bagged and dagged models" in proceedings of *14th International Conference on Machine Learning ICML'97*, pp.367-375, Nashville, Tennessee, USA, 8-12 July, 1997. Morgan Kaufmann, San Francisco, CA.
178. G. Tsoumakes and I. Katakis, "Multi-Label Classification: An Overview" *International Journal of Data Warehousing and Mining*, 3(3): 1-13, 2007. IGI Global.
179. Y. Wang and I.H. Witten, "Induction of model trees for predicting continuous classes" in proceedings of *the poster papers of the 9th European Conference on Machine Learning ECML'97*, 23-25 April, Prague, Czech Republic, 1997. Springer.
180. G. Webb, "Decision Tree grafting from the all-tests-but-one-partition" in proceedings of *16th International Joint Conference on AI*, pp. 702-707, 31 July - 6 Aug,

Stockholm, Sweden, 1999. Morgan Kaufmann, San Francisco, CA.
181. WEKA manual, version 3.6.6, "Bayesian Network classifiers", pp. 115-157, 1998. Found at: http://etudiant.istic.univ-rennes1.fr/current/esir2/aci/weka-3-6-6/WekaManual.pdf [last accessed 2/7/2012].
182. Jinxi Xu, Alexander Fraser and Ralph Weischedel, "TREC 2001 Cross-lingual Retrieval at BBN" in proceedings of *Text REtrieval Conference TREC'01*, Nov, 2001. NIST: Gaithersburg, Maryland.
183. Y. Yang and J.O. Pedersen, "A comparative study on feature selection in text categorization" in proceedings of *14th International Conference on Machine Learning ICML'97*, pp. 412-420, 8-12 July, Nashville, TN, USA, 1997. Morgan Kaufmann.
184. B. M. Zahran and G. Kanaan, "Text Feature Selection using Particle Swarm Optimization Algorithm" *World Applied Sciences Journal*, 7 *(special issue of Computer and IT)*, pp. 69-74, 2009. IDOSI Publications.

Appendix I: Relevant Detailed Background Information, Equations, and Comparisons in Literature Review, and Relevant Tables for Developed Corpus

For Chapter 2:

Number of Web sites, Internet users in Arab Countries per World

Country	# Web sites[51]	% Total websites in Arabic Countries	# Internet Users (CIA's World Fact book)[52]	Population (CIA's World Fact book)[5]	Internet users to Population (%)	Websites to internet users (%)
Jordan	2,582	3.92	1.13 million	6.34 million	17.82	0.23
Emirates	7,435	11.28	2.3 million	4.8 million	47.92	0.32
Bahrain	1,753	2.66	250,000	727,785	34.35	0.7
Algeria	4,372	6.63	3.5 million	34.18 million	10.24	0.125
Saudi Arabia	9,575	14.53	6.2 million	28.69 million	21.61	0.154
Sudan	1,472	2.23	1.5 million	41.09 million	3.65	0.98
Somalia	146	0.22	98,000	9.83 million	0.997	0.15
Iraq	1,872	2.84	54,000	28.95 million	0.187	3.47
Kuwait	2,354	3.57	900,000	2.69 million	33.46	0.26
Morocco	4,024	6.11	7.3 million	34.86 million	20.94	0.06
Yemen	999	1.52	320,000	22.82 million	1.402	0.312
Tunisia	2,672	4.05	1.72 million	10.49 million	16.397	0.155
Comoros	33	0.05	21,000	731,438	2.87	0.157
Djibouti	36	0.05	11,000	516,055	2.13	0.33
Syria	3,882	5.89	3.47 million	20.18 million	17.195	0.112
Oman	1,098	1.67	340,000	3.42 million	9.94	0.323
Palestine (West Bank)			355,500 (2009[53])	2.461 million (2009[6])		
Palestine (Gaza Strip)			-	1.55 million (2009[6])		
Qatar	1,006	1.53	351,000	833,285	42.12	0.287
Lebanon	5,725	8.69	950,000	4.02 million	23.63	0.603
Libya	1,567	2.38	260,000	6.31 million	4.12	0.603
Egypt	12,656	19.2	8.62 million	83.08 million	10.38	0.147
Mauritania	-	-	30,000	3.13 million	0.96	-
Total	**65,917**		**39.325 million (1.15% only of**	**347.377 million (18.98% of**	**Average is 14.65**	**Average is 0.168**

Indication of Availability of Infrastructure needed for Internet Usage in Arab World[54]

No	Service	Statistics	Country
1.	# of PC (per 100 of population) (2008)	7.5	Jordan
		4	DJibouti
		4	Egypt
		75	Bahrain
		6	Morocco
		70	Saudi Arabia

[51] Collected from: http://www.arabo.com. Last accessed 8/6/2010.
[52] Collected from: http://www.clickz.com/tats/web_worldwide. last accessed 8/6/2010.
[53] From http://www.internetworldstats.com/stats5.htm. Last accessed 8/6/2010
[54] Collected from UNDP http://www.arabstats.org/indicator.asp?ind=249&gid=4&sgid=35. [last accessed 8/6/2010] (in Arabic).

		10	Tunisia
2.	Expenditure from total local income per person on IT and communications (2007) (%)	8.9	Jordan
		2.5	Algeria
		5.8	Egypt
		4.5	Kuwait
		8.3	Morocco
		4.7	Saudi Arabia
		6	Tunisia
3.	Expenditure per person on IT and communications in US$ (2007) (%)	258.7	Jordan
		100.1	Algeria
		95.2	Egypt
		1906.9	Kuwait
		202.3	Morocco
		743.9	Saudi Arabia
		206.3	Tunisia
		2236.4	Emirates
4.	Expenses of using internet (monthly in US$) (2006)	10.9	Jordan
		9.31	Algeria
		4.98	Egypt
		13.72	Kuwait
		5.33	Saudi Arabia
		3.1	Tunisia
		5.44	Emirates
		9.22	Syria
		5.18	Oman
		52.48	Sudan
		5.47	Qatar
		10	Lebanon
		22.05	Libya
		7.9	Bahrain
		6	Yemen
		12.7	Comoros
		41.01	Djibouti
		16.02	Mauritania
5.	Percent of internet users to population (2006) (%)	11.71	Jordan
		35.09	Emirates
		20.67	Bahrain
		5.7	Algeria
		10.55	Saudi Arabia
		7.8	Sudan
		5.64	Somalia
		0.13	Iraq
		25.64	Kuwait
		15.1	Morocco
		1.03	Yemen
		9.2	Tunisia
		2.9	Comoros
		1.1	Djibouti
		5.64	Syria
		9.99	Oman
		7.9	Palestine
		26.57	Qatar
		15.36	Lebanon
		3.3	Libya
		6.9	Egypt
		0.5	Mauritania
6.	# of safe servers per million person (2008)	8.8	Jordan
		125.8	Emirates
		78.2	Bahrain
		0.5	Algeria
		8.3	Saudi Arabia

7.	Internet users per 100,000 of population (2002)	0	Sudan
		0.1	Somalia
		64.9	Kuwait
		1.4	Morocco
		0.2	Yemen
		10.7	Tunisia
		1.6	Comoros
		1.2	Djibouti
		0.1	Syria
		11.5	Oman
		1.3	Palestine
		50.7	Qatar
		13	Lebanon
		0.5	Libya
		1	Egypt
		1.6	Mauritania
		7.73	Jordan
		139.4	Emirates
		19.93	Bahrain
		0.26	Algeria
		6.73	Saudi Arabia
		13.79	Kuwait
		0.9	Morocco
		0.06	Yemen
		0.35	Tunisia
		0.16	Comoros
		7.59	DJibouti
		0.01	Syria
		2.66	Oman
		2.55	Qatar
		21.08	Lebanon
		0.15	Libya
		0.45	Egypt
		0.29	Mauritania
8.	Approximate # of internet users (2006)	797,000	Jordan
		1,708,000	Emirates
		210,000	Bahrain
		2,460,000	Algeria
		4,700,000	Saudi Arabia
		3,500,000	Sudan
		94,000	Somalia
		817,000	Kuwait
		6,100,000	Morocco
		270,000	Yemen
		1,295,000	Tunisia
		21,000	Comoros
		11,000	DJibouti
		1,500,000	Syria
		319,000	Oman
		266,000	Palestine
		290,000	Qatar
		950,000	Lebanon
		6,000,000	Egypt
		30,000	Mauritania
9.	Indicator to internet readiness (2008)	4.18551	Jordan
		4.76199	Emirates
		4.37571	Bahrain
		3.14429	Algeria
		4.28148	Saudi Arabia
		3.28168	Libya
		3.12229	Mauritania

		3.75747	Egypt
		3.97924	Kuwait
		3.59132	Morocco
	(defined as the degree of a	4.34095	Tunisia
	country or local community	3.40690	Syria
	to participate or to benefit	4.08312	Oman
		4.68134	Qatar
		2.2	Jordan
		11.8	Emirates
		12.1	Bahrain
	# of prescribers in broadband	4.2	Saudi Arabia
	service per 100 person	0.2	Mauritania
10.	(2008)	0.9	Egypt
		1.5	Morocco
		2.2	Tunisia
		0.1	Syria
		1.1	Oman
		8.1	Qatar

ML based Works Used for Morphological Analysis

No	Reference #	ML technique used	Training/text collections	Performance Results
1-	[54]	a clustering technique by which they used 2-grams, unique 2-grams and a modified version of it	used five small data sets to extract roots for some irregular cases as *weak* and *hamzated* words	accurate clustering up to 94.06%.
2-	[151]	unsupervised ML based on statistical MT, and an English stemmer	a small parallel corpus as its whole training resources then a monolingual un-annotated text was used to further improve the stemmer.	unsupervised stemmer performance was compared with a GOLD one and was found to have 87.5% agreement
3-	[131]	Buckwalter's analyzer and tables that outputted only stems then such stems are used as inputs for training k-NN for morphological analysis then using two filters	LDC collection	when using also the two filters F-measure values increased to about 57.5%.
4-	[160]	used three existing stemming methods: a) Khoja's stemmer (for root extraction), b) Buckwalter's Morphological Analyzer, c) Al-Shalabi et, al (2003) root extraction algorithm. Then, it compared between their accuracy and looked into improving it using majority voting technique (if no agreement on a specific root by aforementioned methods).	Collected news texts	The Khoja stemmer achieved the highest accuracy among used stemmers. The voting algorithm achieved about 70% accuracy for newspaper texts but slightly less than the Khoja stemmer.
5-	[52]	SNoW package was used to tune state-of-the-art versions of three linear classifiers. The purpose of using such classifiers was to identify only triliteral roots in Arabic. the following were investigated: the features number, linguistic constrains, variable size feature representation, and handling only two types of irregular forms: 1- *weak* (including *eliminated-long-vowels* cases) and 2- *geminated* roots.	This work used the following resources: 1- a list of roots, 2- lists of common prefixes and suffixes, 3- corpora annotated with roots using Buckwalter's morphological analyzer, 4- knowledge of word-formation processes and in particular the behavior of *weak* roots in certain paradigms.	When the classifiers were combined using linguistic knowledge pertaining to word formation processes in Arabic by implementing a scoring function that approximates the likelihood of a given candidate to the root, F value became 80.44%.
6-	[15]	Used Back-Propagation Neural Network (BPNN) for extracting Arabic triliteral roots only. Inputted word size was limited to a maximum of five letters and each letter was encoded to three binary digits where letters in *sOltmwnyhA* are provided specific encoding of 1, 2, or 3 whereas other letters are encoded to zero value.	Implementing this approach required to train it first on a set of 500 5-letter words with roots attached then testing it on other 200 5-letter words.	Accuracy rate of 94%.
7-	[124] work was the first one that	an unsupervised algorithm to build the Arabic word segmenter from a large un-segmented Arabic corpus where this	It used training set as: 1- a small manually segmented Arabic corpus of about	This method achieved around 97% exact match accuracy on test set when

	analyzed Arabic words within their content	work performed the following steps: 1- the algorithm uses a Trigram Language Model (3-gram LM) to determine the most probable morpheme sequence for a given input by calculating the probabilities of morphemes (here finding the stem of the word not its root), 2- the task of a decoder used was to find the morpheme sequence which maximizes the trigram probability of the input sentence (i.e. morphological analysis of word within its context), finally 3- the unsupervised acquisition of new stems from an automatically segmented new corpus is done through three steps: a) select new stem candidates on the basis of a frequency threshold, b) filter out new stem candidates containing a sub-string with a high likelihood of being a prefix, a suffix or prefix-suffix (PS), c) further filter out new stem candidates on the basis of contextual information.	110,000 words, 2- a large un-segmented Arabic corpus of about 155 million words, whereas it's testing set: was about 28,449 word tokens.	including 3-gram LM, PS filter and/or new stems acquisition.
8-	[36] work was the second one that analyzed Arabic words within their context.	It used Hidden Markov Model (HMM) approach for choosing the proper root for each word in text among possible roots. This step was performed after extracting possible roots for such word out-of-context using a rule-based method.	This was performed by training this classifier using an annotated corpus from NEMLAR.	Results show that more than 98% of roots were correctly chosen by system in training set while 94% of roots were correctly chosen in test set.

Other Used Term Weighting Methods

$$Widf(t_k, d_j) = \frac{tf(t_k, d_j)}{\sum_{i \in D} tf(t_k, i)}$$

$$Midf(d_j, c_i) = [1 + \log(tf(d_j, c_i))] \left(\frac{DFR(d_j)}{\sum_{i \in D} tf(t_k, i)} \right)$$

Where $DFR(d_j)$ stand for the number of non-zero values of document d_j, $tf(d_j,c_i)$ is the frequency of term t_k in document d_j which belong to category c_i

Evaluation Metrics for TC

The contingency matrix for category c_i			
Category c_i		Expert Judgments	
		Yes	No
Classifier Judgments	Yes	TP_i	FNi
	No	FPi	TN_i

$$P_i = \frac{TP_i}{TP_i + FP_i} \qquad R_i = \frac{TP_i}{TP_i + FN_i}$$

$$A_i = \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \qquad E_i = 1 - A_i$$

Micro-averaging $$P^{\mu} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)}$$

$$R^{\mu} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}$$

Macro-averaging

$$P^{M} = \frac{\sum_{i=1}^{|C|} P_i}{|C|}$$

$$R^{M} = \frac{\sum_{i=1}^{|C|} R_i}{|C|}$$

F_β function

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \qquad \beta \in [0,\infty[$$

$$F_1 = \frac{2PR}{P + R} \qquad \text{When } \beta = 1$$

Naïve Bayes Classifiers Equations

$$p(c_i / d_j) = \frac{p(c_i)\,p(d_j / c_i)}{p(d_j)}$$

Where: $p(c_i / d_j)$: Probability that a given document d_j belongs to a given class c_i, $p(d_j)$: Probability of document d_j, this probability is a constant, thus can be ignored especially if not possible to calculate, $p(c_i)$: Probability of class c_i, it is computed usually by the percentage of documents in c_i to documents number in all categories, $p(d_j / c_i)$: Probability of document d_j given class c_i, and since documents are modeled as sets of words.

According to Bayes theorem, such words are assumed independent, thus $p(d_j / c_i)$ can be written as:

$$p(d_j / c_i) = \prod_k p(w_k / c_i)$$

So:

$$p(c_i / d_j) = p(c_i) \prod_k p(w_k / c_i)$$

Where: $p(w_k / c_i)$ is Probability that k[th] word of document d_j occurs from class c_i, and this can be computed using info taken from training set (Thabtah, et al 2009) as follows:

$$p(w_k / c_i) = \frac{T_{ct} + \lambda}{N_{ci} + \lambda V}$$

Where T_{ct}: Number of times the word occurs in class c_i, N_{ci}: Number of words in class c_i, V: Size of the vocabulary table, λ: Positive constant, usually 1, or 0.5 to avoid zero probability.

TC Methods that used Stemming Techniques for DR on Arabic

No	TC Method	Stemming method	Compared
1.	SVM, k-NN, NB [135]	Light stemmer Larkey et al [121] work	**Did not give detailed results but reported that it degraded SVM performance**
2.	k-NN [63]	a- Al-Shalabi root extractor [17] b- Aljlayl light stemmer [9] c- Word clustering method	**All methods improved effectiveness of classifier compared to word, the best improvement was for light stemming, then word clustering, then root extraction methods.**
3.	k-NN, Rocchio, NB [108]	Both Aljlayl and Frieder [9] and Larkey, et al [121] light stemmers (no details)	**with stemming, for both k-NN and Rocchio: for k-NN with *tf* or *tfidf*, performance improved but others it degraded, No**

			stemming for NB
4.	k-NN, Rocchio [171]	a- root-based (simple explanation but no ref) b- light stemmer (no info) c- statistical (n-gram, 2 or 3) d- hybrid (statistical + light stemmer)	**(all methods gave better results than words only but the hybrid method gave the best improvement in classifiers performance.**
5.	AdaBoost.M1 with C4.5, [147]	DIINAR.1 lexicon [56]	**This part was performed after experimenting on 1250 doc among 5 classes using only Boosted C4.5 classifier and 3 feature selection methods χ^2, IG, Gain Ratio (GR) and compared with no selection for original words, lemmas and roots separately. Macro average F1 results showed that for roots best value for F1 = 88.46% using 161 terms and IG.**
6-	SVM, NBN classifiers [146]	DIINAR.1 lexicon [56]	**Both SVM and NBN were used on vector representation of 7034 doc among 7 classes representing words, lemmas, roots separately using *tfidf* and studied the effect of those VSM on classification and used for FS separately IG and χ^2. Evaluation was by F1, R, P, accuracy and results show that using roots outperformed others in terms of F1 and accuracy values.**
7-	SVM, NBN classifiers [148]	a- DIINAR.1 lexicon [56] b- Statistical 3-gram and 4-gram	**Both SVM and NBN were used on vector representation of 7034 doc among 7 classes representing words, lemmas, roots word 3-gram, word 4-gram separately using *tfidf* and studied the effect of those VSM on classification and used for FS both separately IG and χ^2. Evaluation was by macro-averaging F1, R, P and Accuracy and results show that using word 3-gram outperformed others in terms of F1 (92.4%) and accuracy (92.3%) values.**
8-	SVM light, [154]	For stemming and root extraction used 2 different systems: a) Al-Stem for finding stems and Sebawai for roots [49], [108] b) both RDIMORPHO3 stemmer and root extractor [28].	**Results show: 1- using Al-Stem + MI or IG enhances the performance for small sized dataset, 2- using the words leads to worst performance in small datasets while in large datasets its performance was the among the best, 3- Al-Stem performed better than RDI stemmer while RDI root extractor performed better than Sebawai one. However, no significance tests were provided.**

FSS Main Functions *(derived from (Sebastiani, 2002) [161], d: constant damping factor)*

Function	Denoted by	Mathematical form
DIA association factor	$z(t_k, c_i)$	$P(c_i \mid t_k)$
Information Gain	$IG(t_k, c_i)$	$\sum_{c\in\{c_i,\bar{c_i}\}}\sum_{t\in\{t_i,\bar{t_i}\}} P(t,c).\log\frac{P(t,c)}{P(t).P(c)}$
Mutual Information	$MI(t_k, c_i)$	$\log\frac{P(t_k,c_i)}{P(t_k).P(c_i)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\lvert Tr\rvert.[P(t_k,c_i).P(\bar{t}_k,\bar{c}_i)-P(t_k,\bar{c}_i).P(\bar{t}_k,c_i)]^2}{P(t_k).P(\bar{t}_k).P(c_i).P(\bar{c}_i)}$
NGL coefficient	$NGL(t_k, c_i)$	$\frac{\sqrt{\lvert Tr\rvert}.[P(t_k,c_i).P(\bar{t}_k,\bar{c}_i)-P(t_k,\bar{c}_i).P(\bar{t}_k,c_i)]}{\sqrt{P(t_k).P(\bar{t}_k).P(c_i).P(\bar{c}_i)}}$

Relevancy Score	$RS(t_k, c_i)$	$\log \frac{P(t_k \mid c_i) + d}{P(\bar{t}_k \mid \bar{c}_i) + d}$
Odds Ratio	$OR(t_k, c_i)$	$\frac{P(t_k \mid c_i).(1 - P(t_k \mid \bar{c}_i))}{(1 - P(t_k \mid c_i)).P(t_k \mid \bar{c}_i)}$
GSS coefficient	$GSS(t_k, c_i)$	$P(t_k, c_i).P(\bar{t}_k, \bar{c}_i) - P(t_k, \bar{c}_i).P(\bar{t}_k, c_i)$

TC Methods that used FSS methods for DR on Arabic

No	Reference #	FSS methods	Classifiers used	Results
1-	[171]	used for global selection the methods DF, IG, χ^2, NGL, OR and GSS	k-NN and Rocchio classifiers	Using DF thresholding and a hybrid of DF and IG gave the best results when using k-NN or Rocchio classifiers
2-	[136]	used MI, χ^2, NGL, OR, and GSS	SVM classifier	Using χ^2, NGL or GSS gave better results. This work showed that when using MI for 160 features provided better results compared to when using OR.
3-	[147]	IG and χ^2	AdaBoost.M1 to boost a weak classifier (here a decision trees one C4.5) and compared its performance with other four classifiers (C4.5 alone, SVM, NB, and NB Multinomial (NBM).	Results show that when using IG, both SVM and NBM classifiers outperformed NB and C4.5 but slightly higher than AdaBoost.M1 (but still comparable). It was also found, when using χ^2, that both SVM and NBM outperformed NB and C4.5 but slightly higher than AdaBoost.M1.
4-	[137]	used Ant Colony Optimization (ACO) based on χ^2 and compared its effect with χ^2, NGL, GSS, OR, IG, and MI	SVM classifier	Using ACO based on χ^2 for FSS outperformed others when using SVM classifier.
5-	[184]	used Particle Swarm Optimization (PSO) and compared its effect with χ^2, DF, *tfidf* as well as no selection	Radial Basis Function (RBF) Neural Networks (NN) classifier	Using PSO for FSS outperforms the rest FSS methods used for this text classifier.

Specific Classifiers Implemented for Arabic

No	Reference #	Classifier Type	Classifier	FSS methods, Training/testing texts	Results
1-	[94]	Decision trees	ID3	using IG on 2 small data sets with different classes, *tf* for FSS, light stemming, with 2/3-1/3 training-testing ratios	average F1 about **0.70**
2-	[147]	Decision trees	C4.5, AdaBoost.M1 to boost C4.5		average F1 about **0.80,** and for boosting F1 is about **0.84**.
3-	[112]	Statistical	n-gram	Used embeddings to map each document into R representing the tri-gram freq. statistics profiles for that document. Also, it used both the Dice measure and Manhattan distance to compute the distance between the text to be classified and training texts	using tri-gram with Dice outperforms that with Manhattan distance
4-	[70]	Statistical	Maximum entropy	text set with 6 classes, used stemming	F1 = 0.8041
5-	[159]	statistical	Maximum entropy	LDC Arabic newswire (7M words, 1994 part) and used n-gram (either on word level or on character level) as a step towards stemming, used 80%-20% for training-testing ratio	F1 has a max value of **0.627** after number of iterations (5 – 250)
6-	[95] used	ANN	ANN back-	453 documents with 14 classes,	SVD increased F1 to

	ANN back-propagation classifier		propagation classifier	used Al-Stem for stemming, used SVD for FSS (i.e. reducing features to 200), limited number of unique words to 739, used 2/3 -1/3 for training-testing ratio	**0.88** compared to 0.85 without it
7-	[60]	distance-based	Dice similarity	1000 documents with 10 classes, used Al-Shalabi, et al [17] algorithm for stemming, used 50%-50% for training-testing ratio	micro ***R* = 0.628, *P* = 0.74**
8-	[171]	profile-based	Rocchio classifier	$\beta = 1.6$, $\gamma = 0.4$, used 1,132 documents with 39,468 words with leaving-one-out method, applied different types of stemming techniques and FSS functions	macro-average F1 has a max value of about **0.94** when applying a hybrid method for stemming and another hybrid method for FSS by using DF thresholding and IG
9-	[27]	association rule mining	Apriori and CHARM algorithms to find Frequent Closed Item sets and Frequent Item sets, CHARM for soft-matching Association rules	Frequent Closed Item sets where used, used the RDI morphological analyzer for stemming, used a min. threshold for Support and Confidence of 15% and 70% respectively, proposed a semantic similarity function, tested on an Arabic textual database of 5,524 records	It induced accurate predictive rules of implemented system despite the variation of automatically extracted textual databases. It also illustrate the excellence of soft-matching over hard exact-matching

NB Classifier Implemented for Arabic

No	Reference #	Weighing, FSS	Prob. eq	Corpus + class	Training - testing	Results
1.	[90]	- , -	Mod	600 + 6	70-30%	**F1=93.69%**
2.	[89]	- , -	-	2244 + 5	10-fold cross validation	**Average F1 = 0.884**
3.	[7]	- , -	-	Leed's collection+3	-	**Accuracy 60%**
4.	[108]	Boolean, -	-	1445+9	k-fold cross validation	**Micro average F1=84%**
5.	[10]	*tfidf*, DF threshold	mod	12+12	12+1	**F1=85%**
6.	**[135]	*tfidf*, local χ^2, # terms = 162 gave best results	mod	1445+9	2/3-1/3	**Macro F1=84.54%** **
7.	**[174]	local χ2, # terms = 800 gave best results	mod	1562+6	70-30%	**Macro F1 = 72.8%** **
8.	** [147]	*tfidf*, roots, χ^2, IG	Used NB and NBM (using Weka)	6825+7	Stratified 10-fold cross validation	**when using 1239 features, 1- for NBM** classifier: a) using IG max F1 is about 88% , b) using χ^2 max F1 = 87.5%. **2- for NB** classifier: a) using IG F1 is 75%. b) using χ^2 F1 is 81%. **
9.	[146]	*tfidf*, roots, stems, # selected features varied from 400 - 2000, χ^2, IG	Used NBM (using Weka)	7034+7	Stratified 10-fold cross validation	**For features no = 2000, when using Lemma (highest value but still comparable to using root): for accuracy = 87.79% when using** χ^2 and 87.63% when using IG **(about the same), whereas for F1= 0.878 when using** χ^2 and 0.876 when using IG.
10.	[148]	*tfidf*, roots, stems, 3-gram, 4-gram # selected features varied from 400 –	Used NB network (using Weka)	7034+7	Stratified 10-fold cross validation	**For features no = 2000, when using 3-gram (highest value than using Lemma or root): for accuracy** = 89.49% when using χ^2 **and 89.62% when using IG, whereas for F1=** 0.894 when

		2000, χ^2, IG				using χ^2 **and 0.896 when using IG. Comparing this with those for Lemma there is an improvement of about 2%**
11.	[71]	*tfidf*, rooted features selected from 50 -> 2000 based on highest *tfidf* values	mod	1550+5	1/3-2/3 1/2-1/2 2/3-1/3 Leave-one-out	**Accuracy = 68.78% for trained, 62% for tested**
12.	[141]	-, roots	mod	300 + 10	-	**Accuracy about 62.2%**
13.	[34]	Norm *tfidf*, light stemming	-	242+6	k-fold cross validation	**73.6% (not stated if provided value is for F1, P, R or other)**
14.	** [61]	Feature vector composed of words, their *tf* and *idf*, stemming	mod	1000+10	50-50%	**For NB on average P=R about 80%.** **

** *: this mark means that values shown were calculated by me from figures shown in paper so an estimate.*

k-NN Classifier Implemented for Arabic

No	Paper	K value	Similarity used	FSS	Corpus size + classes	training - testing	Results
1.	** [135]	-	-	here k-NN results shown for comparison	1445 + 9	2/3-1/3	**F1 = 72.72%** **
2.	[63]	10	-	Stemming, weighting *tf*	15,000 + 3	(60-40)%	**P = 92%, R = 91% for light stemming (highest among other stemming methods)**
3.	** [62]	-	-	Stemming	15,000 + 3	60-40%	**Results of micro P, R shown (I cal. Micro average F1 = 91.5% for light stemming &** 88% for stemming**)** **
4.	** [98]	29	Cosine	Local χ^2	2206 + 2	(99 – 1)%	**Max F1 = 93.6% at 250 terms** **
5.	[20]	-	Cosine	Based on DF > 3	1445+ 4	(60-40)%	**F1 = 73.57% for using n-gram** & 66.88% for single terms
6.	[90]	-	Cosine	-	600 + 6	(70-30)%	**F1 = 90.93%**
7.	[173]	11	Cosine, Dice, Jaccard	-	Small + 6	(70-30)%	**F1 = 94.91% for both Dice and Jaccard when using *tfidf***
8.	[19]	Varied, 18, effectiveness started to decline at k>24	Cosine	Based on DF + light stemming	621 + 6	(90-10)%	**Both Micro** recall & precision = 95% at k=18**, but 96% at k = 21,**
9.	[109]	13	Jaccard	Light stemming	1445 + 9	k-fold cross validation k=4	**using *tfidf* F=78%** & *tf* F=69% had improved with stemming**, while using *Widf* was 80%** and lowered to 73% with stemming
10.	** [171]	1 - 19	Euclidean distance	Hybrid stemming using tri-gram + light stemming, hybrid of DF threshold + IG	1132 + 6	Leave-one-out	**Macro F1=52% for k=1,** lower for higher k as no of features about 5000 **
11.	[34]	1 - 20 but as	Euclidean	Light	600 + 6	k-fold cross	**84.2% (not mentioned**

		k > 15 effectiveness of classifier decreased	distance	stemming		validation	**which F, P, R or else)**
12.	** [107]	19	Jaccard	Light stemming + IG	600 + 6	Varied (shown for 60-40%)	**Macro max F1 = 75.8% at 360 training doc** **
13.	** [61]	Change k: 10, 20, 50, 100	Dice	Stemming	1000 + 10	50-50%	**On average P=R=66%.** ** (results shown here for k=50)

: this mark means that values shown were calculated by me from figures shown in paper so an estimate.

SVM Classifier Implemented on Arabic Texts

No	Ref. #	SVM type	weighting	FSS	Corpus size + classes	training - testing	Results
1.	** [137]	TinySVM	*tfidf*	Used ACO based on local χ^2 algorithm, compared with other FSS methods (NGL, GSS, OR, IG, MI. Max. performance at 160 terms.	1445+9	966 – 479 (2/3-1/3)	Macro F1 for: no FSS is 74.04, for χ^2 *is 87.54*, for NGL is 86.5, for GSS is 86.5, for OR is 78.75, for MI is 78.53, for IG is 78.81, for **ACO is 89.61%.** **
2.	** [135]	TinySVM	*tfidf*	Local χ^2 Max. performance at 162 terms.	1445+9	2/3-1/3	Macro average **F1 = 88.11%** **
3.	** [136]	TinySVM	*tfidf*	χ^2, NGL, GSS, OR, MI.	1445+9	2/3-1/3	Macro average **F1 about 87.5% for χ^2 at 160 features.** **
4.	[8]	RapidMiner	Boolean	Local χ^2 applied on DF. Top 30 terms of each class	7 data sets (17,658 documents) each has its different no of classes	70-30	**Average Accuracy 68.65%**
5.	** [98]	Gist SVM and kernel principal components analysis software toolkit	*tfidf*	Local χ^2 varied 50 -> 500 features	2235-2	2206-29	**Micro F1, max at 450 terms is 98.2%** but all its value at all terms selected is higher than k-NN ones. **
6.	** [147]	SMO (from Weka)	*tfidf*	χ^2, IG	6825+7	Stratified 10-fold cross validation	**Results show that when using 1239 features: using IG for SVM max F1 is about 88%** and when using χ^2 it is about 88%. **
7.	[146]	SMO (from Weka)	*tfidf*	χ^2, IG	7034+7	Stratified 10-fold cross validation	**For features no = 2000, when using root (highest value but slightly higher to using Lemma): for accuracy = 87.97% when using** χ^2 and 87.80% when using IG**, whereas for F1= 0.880 when using** χ^2 and 0.878 when using IG**.**
8.	[148]	SMO (from Weka)	*tfidf*	χ^2, IG	7034+7	Stratified 10-fold cross validation	**For features no = 2000, when using 3-gram (highest value than using Lemma or root): for accuracy = 92.41% when using** χ^2 and 92.28% when using IG**, whereas for F1= 0.924 when using** χ^2 and 0.923 when using IG**. Compared to when using root there is an improvement of about 4%**

9-	[89]	From Weka	-	-	2244 + 5	10-fold cross validation	**Average F1 = 0.954**

***: this mark means that values shown were calculated by me from figures shown in paper so an estimate.*

For Chapter 3:

Samples of Function Words

Samples of Function words = {"إلى"، "الى"،"إليها" ،"اليكن"، "وإليها"، "فإليها"،"إليهما"، "إليهم"،"اليه"، "إليه"، "اليهما"، "اليهم"،"إليكن"،"إليكم"، "إليكما"، "إليك"، "اليك"، "اليكما"، "اليكم" ، "اليها"، "فاليها"، "وإلى"، "فإلى"، "فالى"، "وإليهما"، "فإليهما"،"فاليهما"، "فاليهم"، "فإليهم"، "فإليه"، "وإليكم"، "فإليكم"، "وإليك"، "فإليك"، "فاليك"، "فإليكما"، "فاليكما"، "فاليكن"، "وإليكن"، "فإليكن"، "على" ،"عليه"، "عليهما"،"عليهم"، "عليك"،"عليكما"، "عليكم"، "عليكن"، "عليها"، "وعلى"، "وعليه"، "فعليه"، "وعليهما"، "فعليهما"، "وعليهم"، "فعليهم "،"فإن"، "بأن"،"انما"،"أنما"، "إنما"، "أنها"، "انه" ، "كأن" ،"فأن"،"فإنه"، "فأنه" ، "فإنها"،"فانما"،"فأنما"، "فإنما"،"فأنها" ، "فانه"، "فكأن "،"وأنك"، "وأنكما"، "وأنكي"، "وأنكم"، "وإنكم"، "وإنكن"، "وإنكي"، "وإنك"، "أنك"، "أنكما"، "أنكي"، "أنكم"، "إنكم"، "إنكن"، "إنكي"، "إنك"، "انها"،"فأني" ، "أنى" ،"بانه"، "بأنه"، "بأنها"،"بأنهم"، "بأنهن"، "لأنه" ، "له"، "لك"، "لها"، "لكما"،"لهما"، "لكم"، "لهم"، "لكن"، "لهن" ،"ولهم"، "ولها"، "وله"، "ولهن"، "فله"، "لانه"، "بانها"، "انكي"، "انكم"، "انكن" ،"ولك"، "فلها"، "ولكما"،"ولهما"، "ولكم"، "فلهم"، "ولكن"، "فلهن"، "لات"، "إلا"، "الا"، "ألا"، "والا"، "وإلا"، "وألا"، "فالا " ، "وأضحت"، "واضحت" ، "وأضحتا"، "واضحتا"، "فأضحت" ، "فاضحت"، "فأضحتا"، "فاضحتا" ، "أضحوا"، "اضحوا"، "وأضحوا" ، "واضحوا"، "فأضحوا"، "فاضحوا"، "يضحي"، "ويضحي"، "فيضحي"، "تضحي"، "مايفتؤا"، "ومايفتؤا"، "فمايفتؤا"، "مايفتئان"، "ومايفتئان"، "فمايفتئان"، "مايفتئن"، "ومايفتئن"، "فمايفتئن"، "ماتفتئن"، "وماتفتئن"، "فماتفتئن ، "ماانفك"، "ومماانفك"، "فماانفك"، "ماانفكا"، "ومماانفكا"، "فماانفكا"، "مـاانفكوا"، "ومـاانفكوا"، "فمـاانفكوا"، "مـاانفككن"، "ومـاانفككن"، "فمـاانفككن"، "مـاينفـك"، "ومـاينفـك"، "فمـاينفـك"، "ماتنفـك"، "وماتنفـك، "فماتنفك"، "ماينفكا"، "وماينفكا"، "فماينفكا"، "ماتنفكا"، "وماتنفكا"، "فماتنفكا"، "ماينفكوا"، "ومـاينفكوا " ،"كادي"، "كدتما"، "كدتم"، "كدتن" "كرب"، "وكرب"، "فكرب"، "كربا"، "وكربا"، "فكربا"، "كربوا"، "وكربوا"، "فكربوا"، "كربت"، "وكربت"، "فكربت"، "كربتا"، "وكربتا"، "أوشك"، "وأوشك"، "فأوشك"، "أوشكا"، "وأوشكا"، "فأوشكا"، ، "فكربتا"، "كربنا"، "كربتما"، "كربتم"، "كربن"، "وكربن"، "فكربن" "ابتدأن"، "وابتدأن"، "فابتدأن ، "انبرى"، "وانبرى"، "فانبرى"، "انبرت"، "وانبرت"، "فانبرت"، "انبريا"، "وانبريا"، "فانبريا"، "انبرتا"، "وانبرتا"، "فانبرتا"، "انبروا"، "وانبروا"، "فانبروا"، "انبرين"، "وانبرين"، "فانبرين"**}**

Appendix II: Additional Detailed Informations, Tables and Figures of Chapter 4

TIM BUCKWALTER'S ARABIC TRANSLITERATION

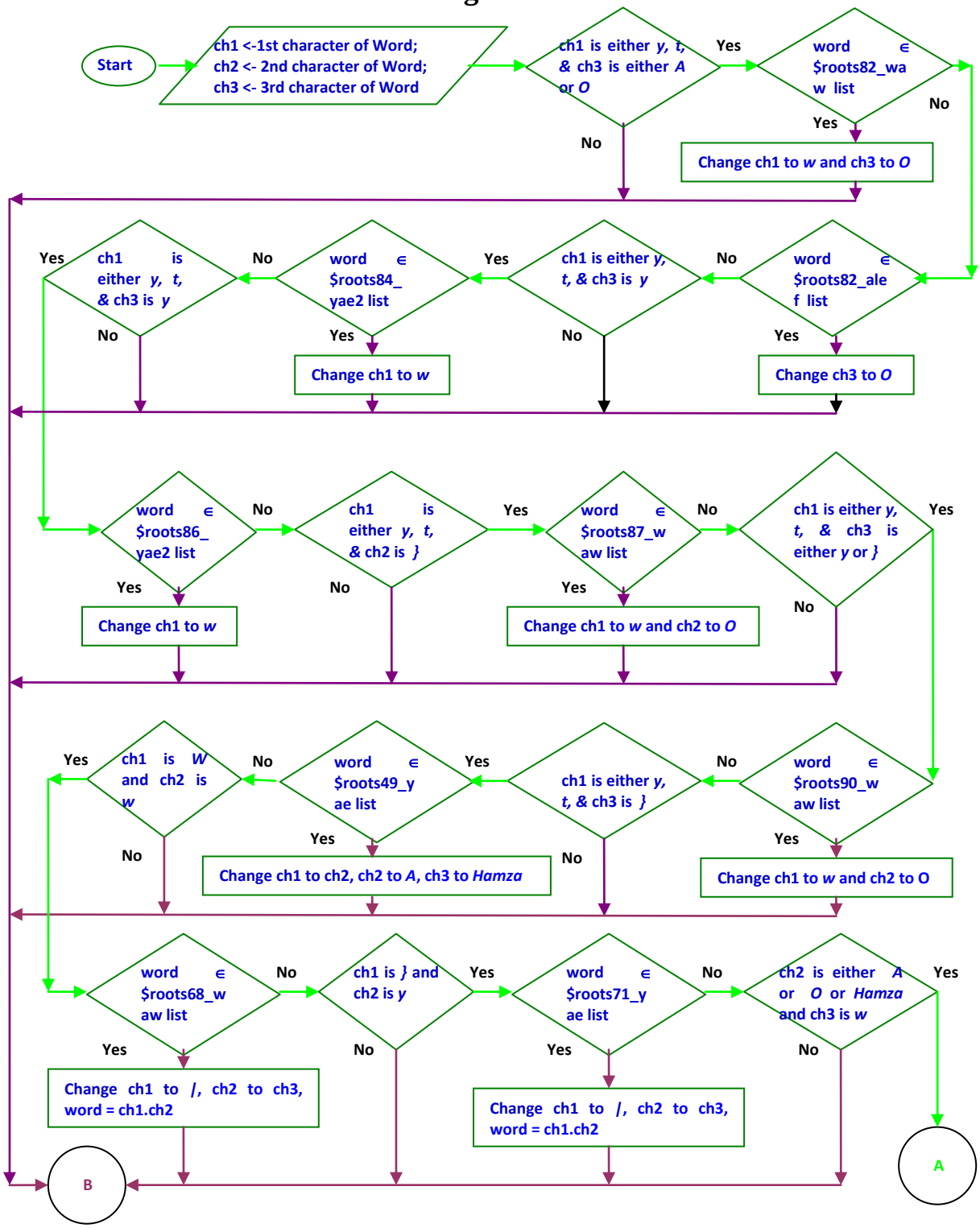
I developed my transliteration system before XML days. To make it XML-friendly I would:
replace < with I (for *hamza*-under-*alif*)
replace > with O (for *hamza*-over-*alif*—the A is already used for bare *alif*)
replace & with W (for *hamza*-on-*waw*)

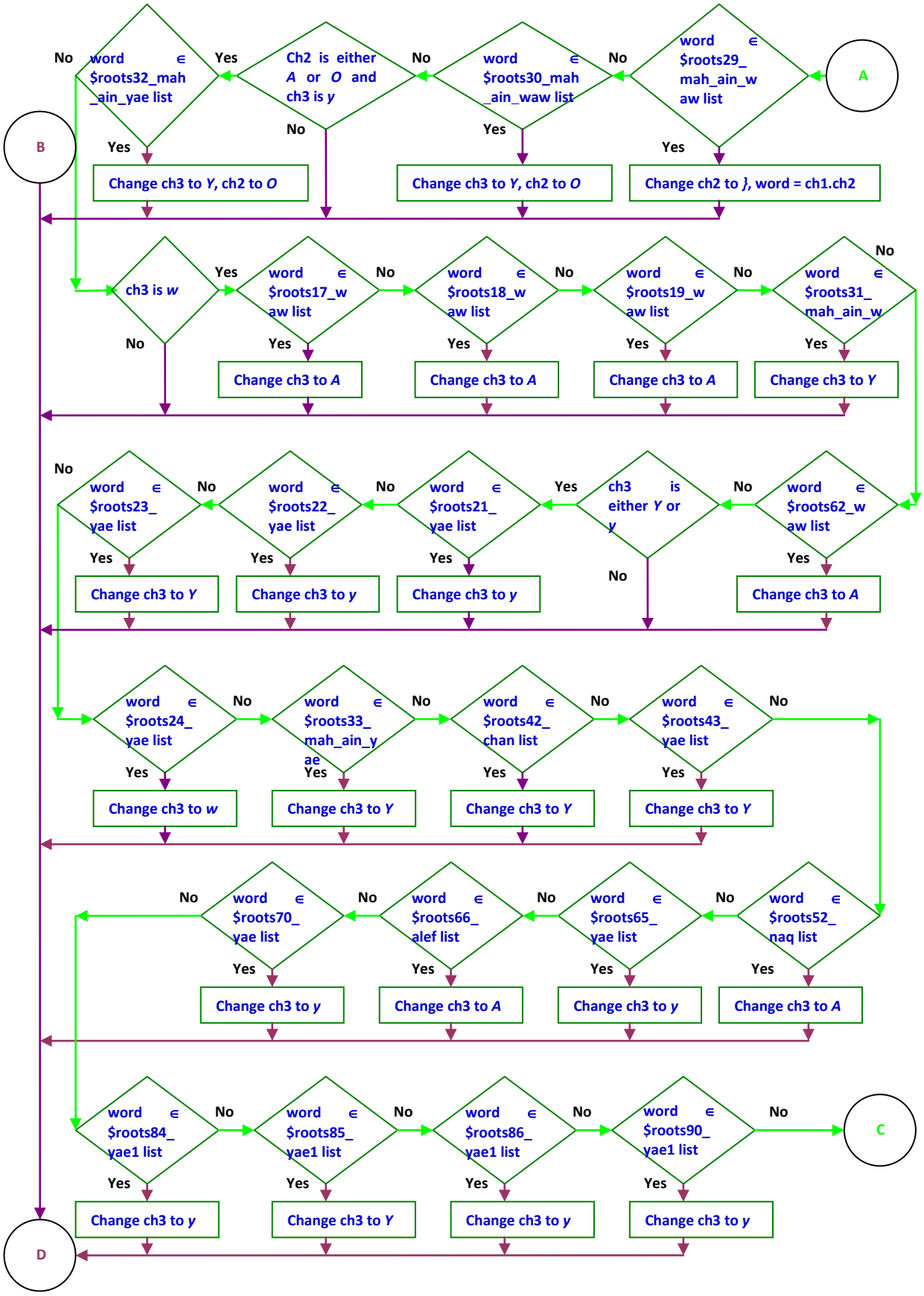
ء	'	ذ	*	ل	l
آ	|	ر	r	م	m
أ	>	ز	z	ن	n
ؤ	&	س	s	ه	h
إ	<	ش	S	و	w
ئ	}	ص	s	ى	Y
ا	A	ض	D	ي	y
ب	b	ط	T	ً	F
ة	p	ظ	z	ٌ	N
ت	t	ع	E	ٍ	K
ث	v	غ	g	َ	a
ج	j	ـ	_	ُ	u
ح	H	ف	f	ِ	i
خ	x	ق	q	ّ	~
د	d	ك	k	ْ	o

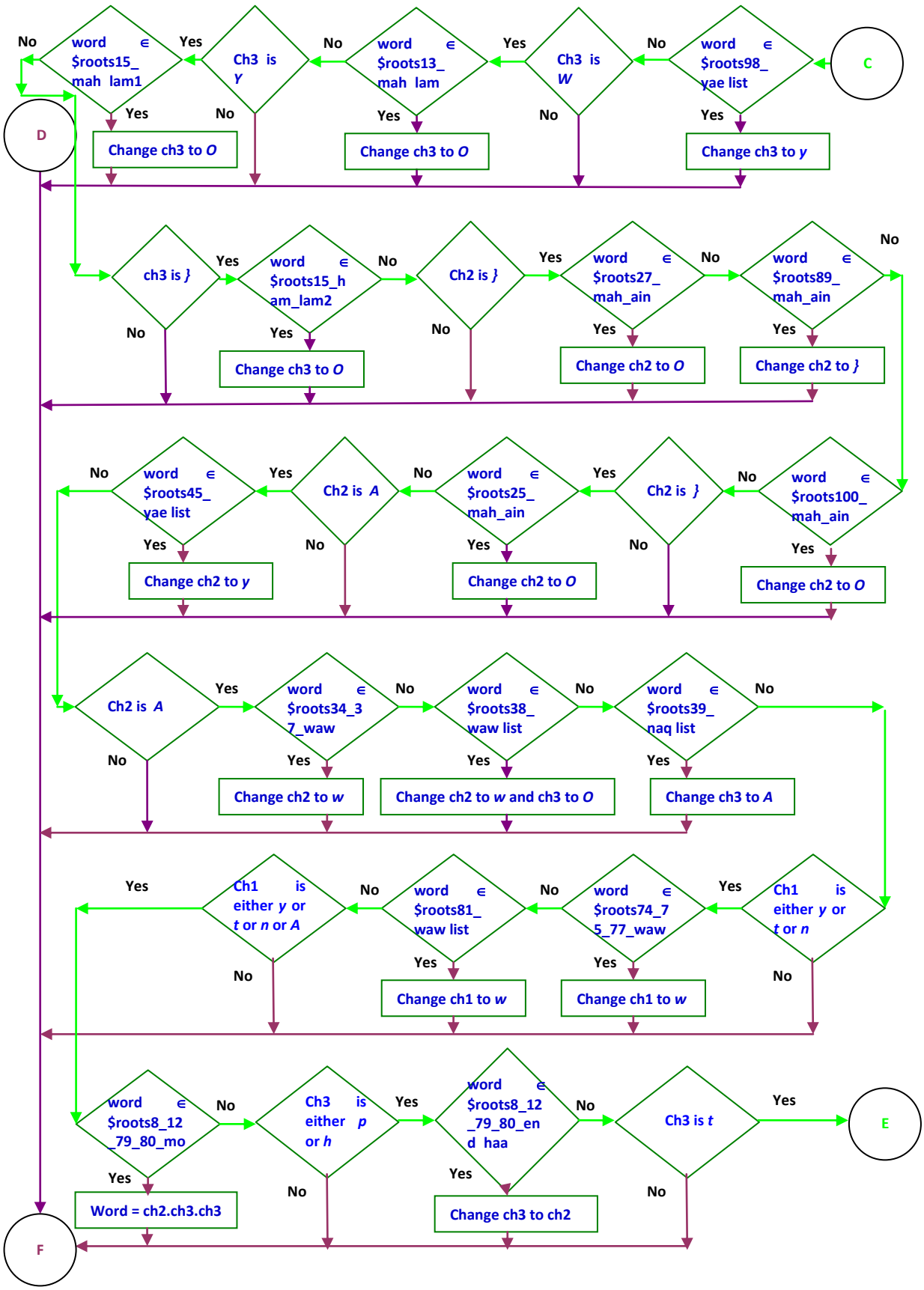
Examples of Two Letter Geminated Words

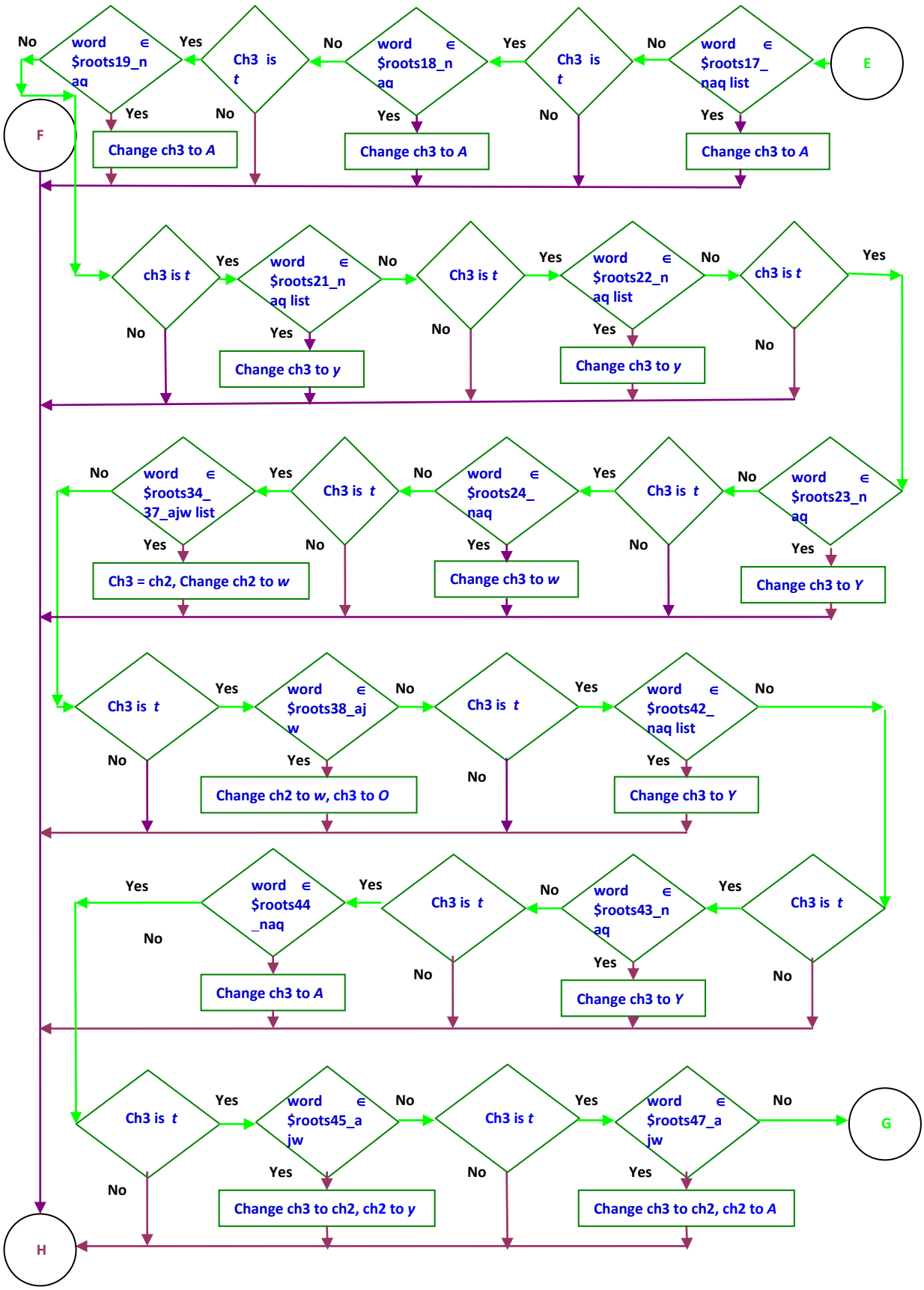
Two letter geminated words list = {"مد" ،"شد"، "تم" ، "ظل" ، "قل"، "ضم"، "ضل"، "ظن"، "حل"، "حن"، "جد" ، "ود"، "طن"، "خف"، "مر"، "حق"، "شك" ، "رد"، "كل"، "بح"، "بت"، "بص"، "بش"، "بر"، "ثج"، "ثط"، "ثع"، "جن"، "حب"، "سد"، "سف بث"، "بج"، "بد"، "بذ"، "بز"، "بس"، "بض"، "بط"، "بظ"، "بع"، "بغ"، "بك"، "تف"، "تل"، "ثر"، "ثل"، "ثم"، "جب"، "جث"، "جح"، " جز"، "جس"، "جش"، "جص"، "جض"، " ,""جخ"، "جذ"، "جر"، "بق"، "بل"، "بن"، "تب"، "تخ"، "حد"، "حر"، "حش"، "حظ"، "حف "جع"، "جف"، "جل"، "جم"، "حت"، "حث"، "حج"، "حذ"، "حز"، "حس"، "حص"، "حض"، "حط"، "حك"، "حم"، "خب"، "تر"، "تك"، خج"، "خد"، "خذ"، "خر"، "خز"، "خس"، "خش"، "خص"، "خط"، "خق"، "خل"، "خم"، " ,""ثب"، "ذم"، "ذن"، "رت"، "رح"، "رخ "دث"، "دج"، "دح"، "در"، "دس"، "دش"، "دع"، "دك"، "دل"، "دم"، "ذح"، "خـت"، "خـن"، "دب"، "دخ"، "دف"، "رف"، "رك"، "رم"، ذر"، "رب"، "رث"، "رج"، "رذ"، "رز"، "رس"، "رش"، "رص"، "رض"، "رع"، "رق"، "زب"، "زت"، "زج"، "زح"، " ,""رن "سب"، "سج"، "سر"، "سك"، "سل"، "سم"، "سن"، "دق"، "دن"، "ذب"، "ذف"، "ذل"، "زخ"، "زر"، "زن"، "شم"، "صد"، "صف"، "سك"، "صم"، "ضد"، "ان"، "أر"، "ار"، "أز"، "از"، "أط"، "اط"، "أض"، "اض"، "وص"، "وج"، "وط""**}**

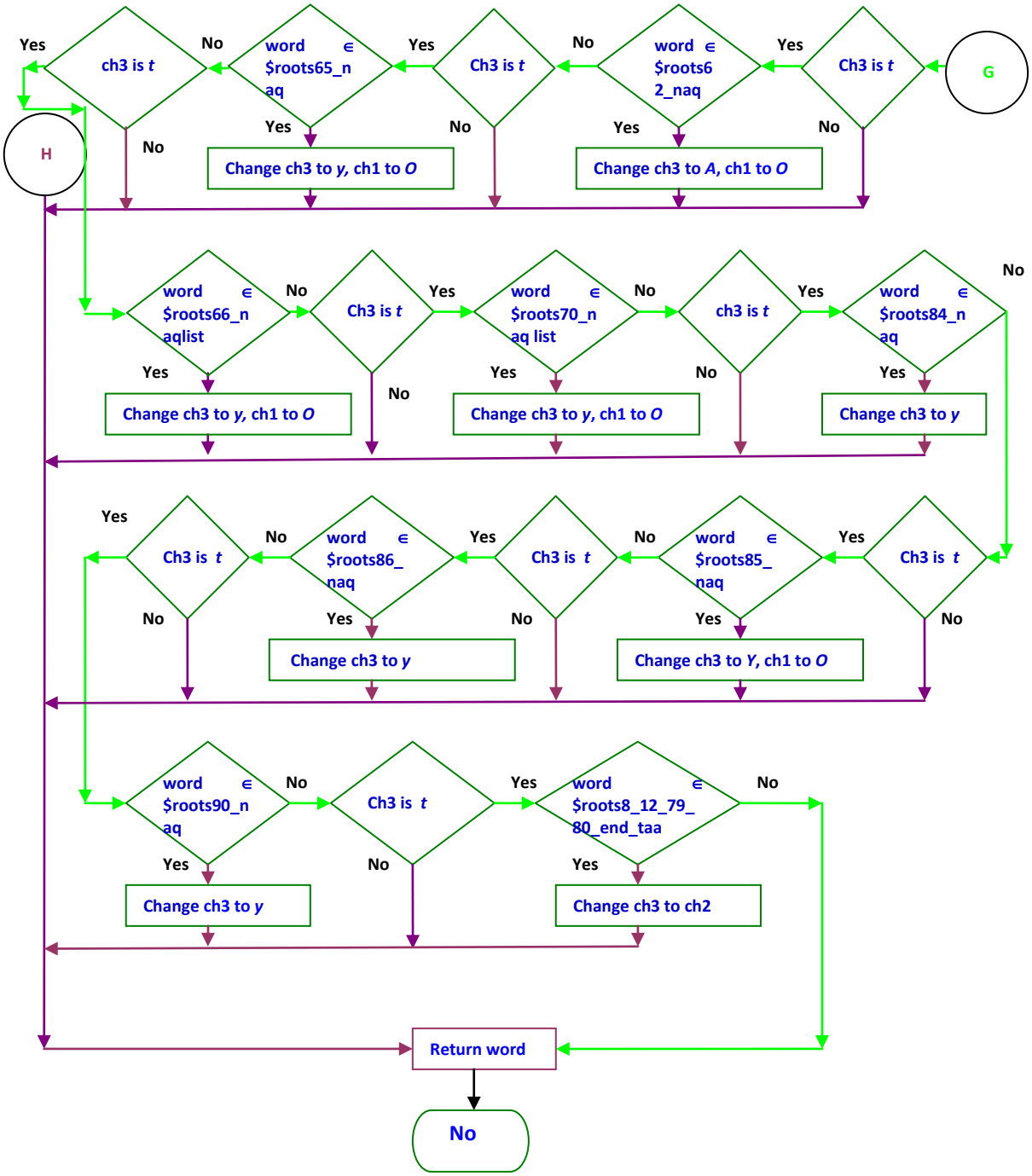
Detailed *Correction* Algorithm Flowchart



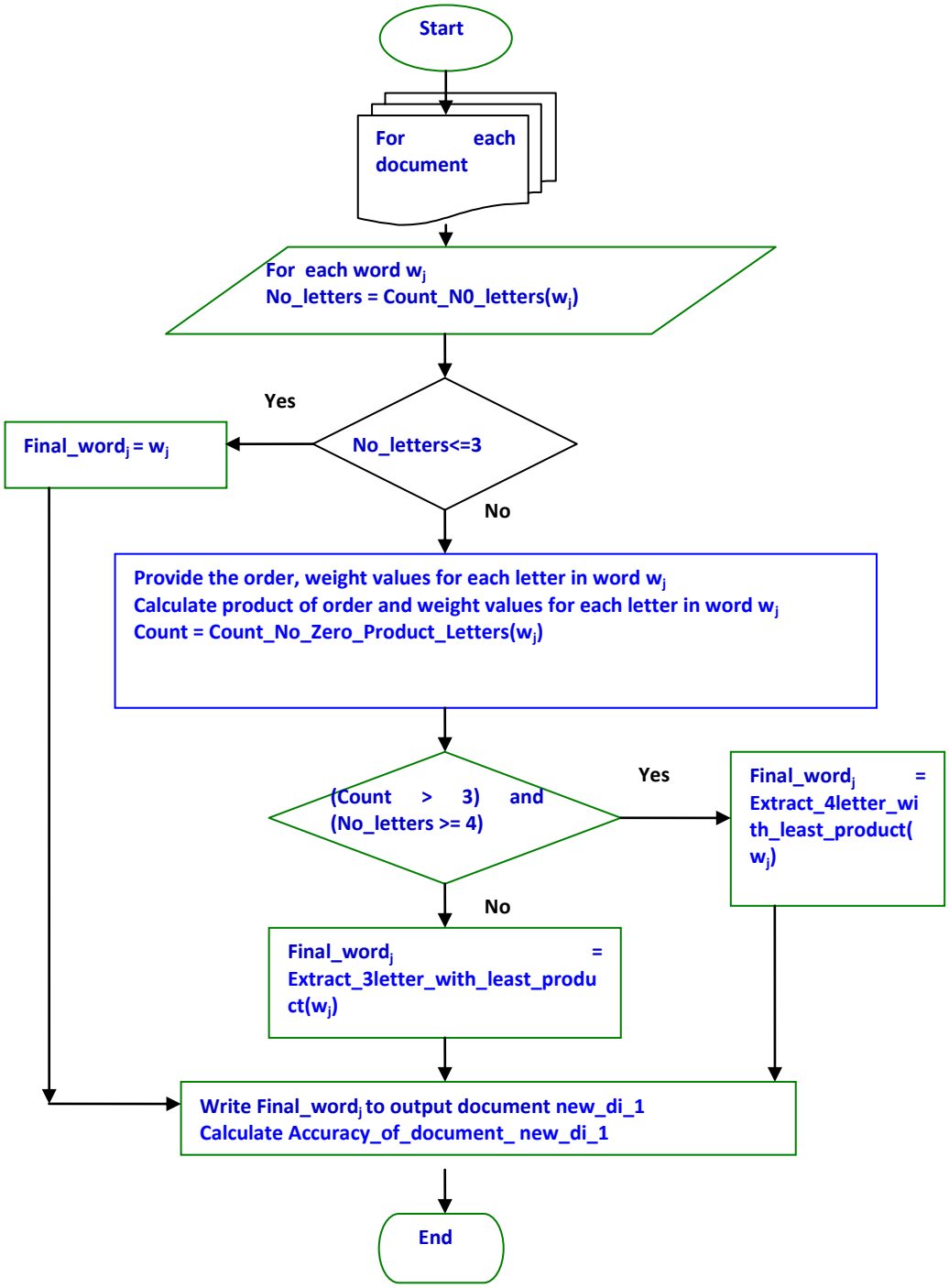

word ∈ $roots32_mah_ain_yae list
Ch2 is either A or O and ch3 is y
word ∈ $roots30_mah_ain_waw list
word ∈ $roots29_mah_ain_waw list
A
B
Change ch3 to Y, ch2 to O
Change ch3 to Y, ch2 to O
Change ch2 to ʃ, word = ch1.ch2
ch3 is w
word ∈ $roots17_waw list
word ∈ $roots18_waw list
word ∈ $roots19_waw list
word ∈ $roots31_mah_ain_w
Change ch3 to A
Change ch3 to A
Change ch3 to A
Change ch3 to Y
word ∈ $roots23_yae list
word ∈ $roots22_yae list
word ∈ $roots21_yae list
ch3 is either Y or y
word ∈ $roots62_waw list
Change ch3 to Y
Change ch3 to y
Change ch3 to y
Change ch3 to A
word ∈ $roots24_yae list
word ∈ $roots33_mah_ain_yae
word ∈ $roots42_chan list
word ∈ $roots43_yae list
Change ch3 to w
Change ch3 to Y
Change ch3 to Y
Change ch3 to Y
word ∈ $roots70_yae list
word ∈ $roots66_alef list
word ∈ $roots65_yae list
word ∈ $roots52_naq list
Change ch3 to y
Change ch3 to A
Change ch3 to y
Change ch3 to A
word ∈ $roots84_yae1 list
word ∈ $roots85_yae1 list
word ∈ $roots86_yae1 list
word ∈ $roots90_yae1 list
C
Change ch3 to y
Change ch3 to Y
Change ch3 to y
Change ch3 to y
D
Yes
No


C
word ∈ $roots98_ yae list
Ch3 is W
word ∈ $roots13_ mah lam
Ch3 is y
word ∈ $roots15_ mah lam1
D
Change ch3 to y
Change ch3 to O
Change ch3 to O
Yes
No
ch3 is }
word ∈ $roots15_h am_lam2
Ch2 is }
word ∈ $roots27_ mah_ain
word ∈ $roots89_ mah_ain
Change ch3 to O
Change ch2 to O
Change ch2 to }
word ∈ $roots45_ yae list
Ch2 is A
word ∈ $roots25_ mah_ain
Ch2 is }
word ∈ $roots100_ mah_ain
Change ch2 to y
Change ch2 to O
Change ch2 to O
Ch2 is A
word ∈ $roots34_3 7_waw
word ∈ $roots38_ waw list
word ∈ $roots39_ naq list
Change ch2 to w
Change ch2 to w and ch3 to O
Change ch3 to A
Ch1 is either y or t or n or A
word ∈ $roots81_ waw list
word ∈ $roots74_7 5_77_waw
Ch1 is either y or t or n
Change ch1 to w
Change ch1 to w
word ∈ $roots8_12 _79_80_mo
Ch3 is either p or h
word ∈ $roots8_12 _79_80_en d haa
Ch3 is t
E
Word = ch2.ch3.ch3
Change ch3 to ch2
F

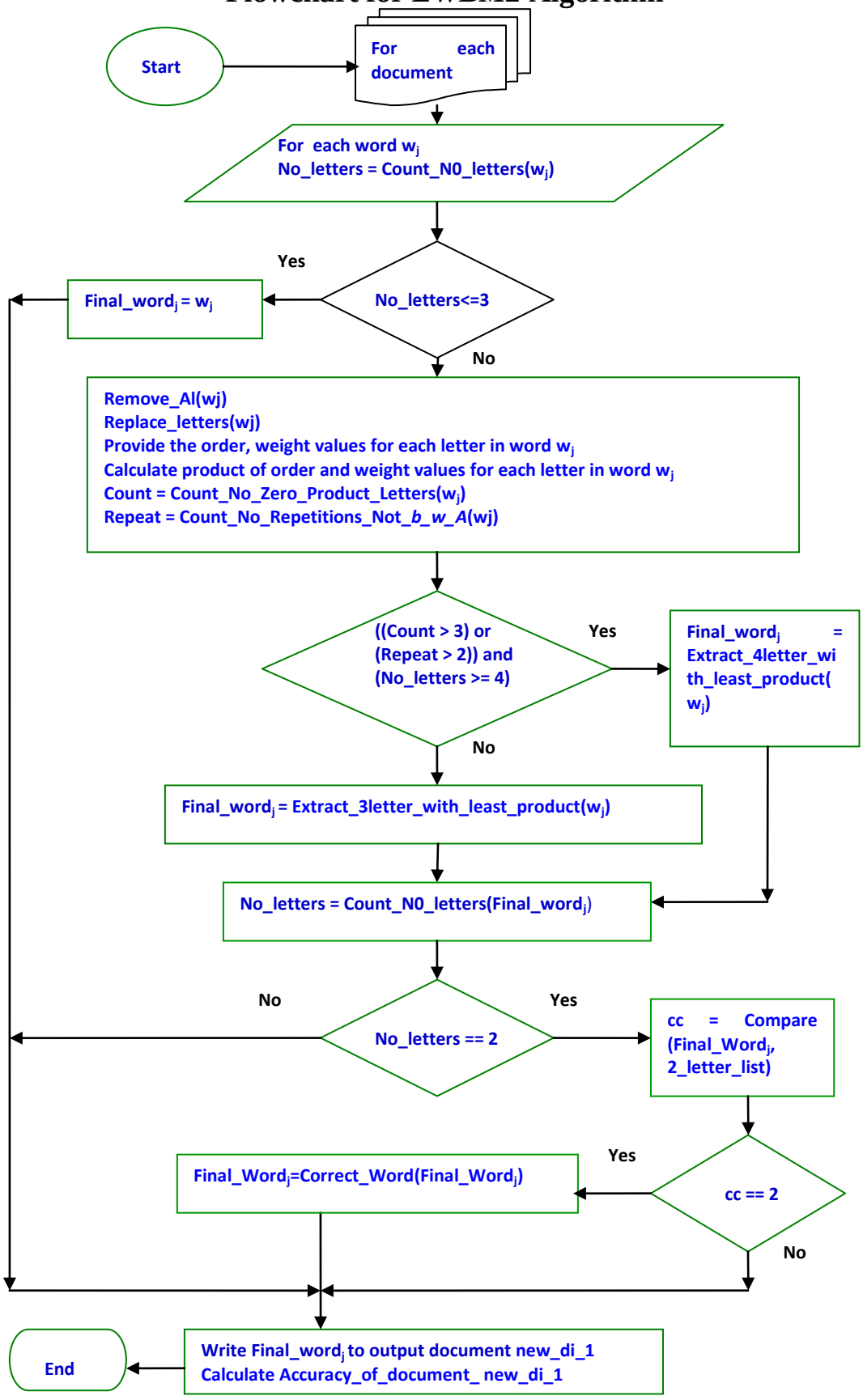

E
word ∈ $roots17_naq list
Ch3 is t
word ∈ $roots18_naq
Ch3 is t
word ∈ $roots19_naq
Ch3 is t
Change ch3 to A
F
ch3 is t
word ∈ $roots21_naq list
Ch3 is t
word ∈ $roots22_naq list
ch3 is t
Change ch3 to y
word ∈ $roots23_naq
Change ch3 to Y
word ∈ $roots24_naq
Change ch3 to w
word ∈ $roots34_37_ajw list
Ch3 = ch2, Change ch2 to w
Ch3 is t
word ∈ $roots38_ajw
Change ch2 to w, ch3 to O
word ∈ $roots42_naq list
Change ch3 to Y
word ∈ $roots43_naq
Ch3 is t
word ∈ $roots44_naq
Change ch3 to A
Change ch3 to Y
Ch3 is t
word ∈ $roots45_ajw
Change ch3 to ch2, ch2 to y
word ∈ $roots47_ajw
Change ch3 to ch2, ch2 to A
G
H
Yes
No


G
ch3 is t
Yes
No
word ∈ $roots65_naq
Ch3 is t
word ∈ $roots62_naq
Ch3 is t
H
Change ch3 to y, ch1 to O
Change ch3 to A, ch1 to O
word ∈ $roots66_naqlist
Ch3 is t
word ∈ $roots70_naq list
ch3 is t
word ∈ $roots84_naq
Change ch3 to y, ch1 to O
Change ch3 to y
Ch3 is t
word ∈ $roots86_naq
Ch3 is t
word ∈ $roots85_naq
Ch3 is t
Change ch3 to y
Change ch3 to Y, ch1 to O
word ∈ $roots90_naq
Ch3 is t
word ∈ $roots8_12_79_80_end_taa
Change ch3 to y
Change ch3 to ch2
Return word
No

Flowchart for *EWBM1* Algorithm



Flowchart for *EWBM2* Algorithm



Samples of Lists used in *Correction* Algorithm

No	Comments on List	List

1.	// from 8 - 12 3-letter root MODDA'AF (root 8 eg رَدَّ - يَرُدُّ root 9 eg ضَلَّ - يَضِلُّ root 10 eg مَلَّ - يَمَلُّ root 11 eg حَبَّ - يَحُبُّ root 12 eg عَضَّ - يَعِضُّ) remove first then check 2 letter roots
	// from 79 - 80 3-letter root METHAL WAWEE MODDA'AF
	// root 79 eg وَصَّ - يَوُصُّ
	// root 80 eg وَدَّ - يَوَدُّ
		$roots8_12_79_80_mod = { "تمد"، "تشد"، "تتم" ، "تظل" ، "تقل"، "تضم"، "تضل"، "تظن"، "تحل"، "تحن"، "تجد" ، "تود"، "تطن"، "تخف"، "تمر"، "تحق"، "تشك" ، "ترد"، "تكل"، "تبح"، "تبت"، "تبص"، "تبش "، "نوج"، "نوص"، "نوط"، "نود"}
		$roots8_12_79_80_end_taa = {"مدت"، "شدت"، "تمت" ، "ظلت" ، "قلت"، "ضمت"، "ضلت"، "ظنت"، "حلت"، "حنت"، "جدت" ، "ودت"، "طنت"، "خفت"، "مرت"، "حقت"، "شكت" ، "ردت"، "كلت"، "بحت "، "أرت"، "ارت"، "أزت"، "ازت"، "أطت"، "اطت"، "أضت"، "اضت" ، "ودت" ، "وصت"، "وجت"، "وطت"}
		$roots8_12_79_80_end_haa = {"مده"، "شده"، "تمه" ، "ظله" ، "قله"، "ضمه"، "ضله"، "ظنه"، "حله"، "حنه"، "جده" ، "وده"، "طنه"، "خفه"، "مره"، "حقه"، "شكه" ، "رده"، "كله"، "بحه"، "بته"، "بصه"، "بشه"، "بره " ، "ودة"، "وصة"، "وجة"، "وطة"}

2.	// root 13 eg نَجَأَ - يَنْجُؤُ return to أ	$roots13_mah_lam = {"نجؤ"}
3.	// root 15 eg بَرِئَ - يَبْرَأُ return end in 1 or in 2 to ئ	$roots15_ham_lam1 = {"برى"، "بذى"، "بسى"، "بهى"، "تفى"، "جمى"، "جنى"، "حدى"، "حلى"، "خجى"، "خذى"، "خرى"، "خطى"، "دفى"، "دنى "، "هزى"، "هنى"} $roots15_ham_lam2 = {"برئ"، "بذئ"، "بسئ"، "بهئ"، "تفئ"، "جمئ"، "جنئ"، "حدئ"، "حلئ"، "خجئ"، "خذئ"، "خرئ"، "خطئ"، "دفئ "، "هذئ"، "هزئ"، "هنئ"}
4.	// root 17 eg دَعَا - يدْعُو return end to ا	$roots17_waw = {"دعو"، "بخو"، "بدو"، "بذو"، "برو"، "بزو"، "بصو"، "بظو"، "بعو"، "بلو"، "بهو"، "تلو"، "ثدو"، "ثرو"، "ثطو"، "ثغو"، "جبو "، "حصو"، "هطو"، "هفو"} $roots17_naq = {"دعت"، "بخت"، "بدت"، "بذت"، "برت"، "بزت"، "بصت"، "بظت"، "بعت"، "بلت"، "بهت"، "تلت"، "ثدت"، "ثرت"، "ثطت "، "هجت"، "هذت"، "هرت"، "هزت"، "هصت"، "هطت"، "هفت"}
5.	// root 18 eg زَهَا - يَزْهَى return end to ا	$roots18_waw = {"زهو"} $roots18_naq = {"زهت"}
6.	// root 19 eg شَقِيَ - يَشْقَى return end to ا	$roots19_waw = {"شقو"، "بزو"، "بهو"، "جسو"، "جهو"، "حظو"، "حفو"، "حقو"، "حلو"، "خفو"، "ربو"، "رجو"، "رخو"، "رذو"، "رضو "، "قعو"، "كدو"، "لهو"، "ندو"، "نسو"، "نشو"، "نقو"} $roots19_naq = {"شقت"، "بزت"، "بهت"، "جست"، "جهت"، "حظت"، "حفت"، "حقت"، "حلت"، "خفت"، "ربت"، "رجت"، "رخت"، "رذت"، "غرت"، "غشت"، "فجت"، "قصت"، "قعت"، "كدت"، "لهت"، "ندت"، "نست"، "نشت"، "نقت"}

Samples of Root Lists

No	Comments on List	List
1.	// from 1 - 7 3-letter root SALEM // root 1 eg نَصَرَ - يَنْصُرُ	$roots1 = {"نصر"، "بتر"، "بثق"، "بجد"، "بجس"، "بجل"، "بدر"، "بدن"، "بذخ"، "بذر"، "بذل"، "بزر"، "بزق"، "بزل"، ","بذم"، "برج"، "برح"، "برد"، "برز"، "برض"، "برق"، "برك"، "برم"، "بزج"، "بزم"، "بسر"، "بسط"، "بسق"، "بسل"، "بشر"، "بصق"، "بطر"، "بطل"، "بطن"، "بغض"، "بقر"، "بكل"، "بلج"، "بلد"، "بلص"، "بلط"، "بلغ"، "بلق"، " ,"بقط"، "بقل"، "بكت"، "بكر"، "بكس"، "بكش"، "بنق"، "تبر "، "خطب"، "خفت"، "خفد"، "خفش "، "همل"، "هند"}
2.	// root 2 eg ضَرَبَ - يَضْرِبُ	$roots2 = {"ضرب"، "بتك"، "بتل"، "بجم"، "بسم"، "بشق"، "بشك"، "بصم"، "بضع"، "بطش"، "تبن"، ","ترز"، "تسع"، "تفن"، "ثبق"، "ثبن"، "ثرب"، "ثرم"، "ثلب"، "ثلط"، "ثلم"، "ثمد"، "جبذ"، "جدب"، "جدف"، "جذب"، "جذف"، "جذم"، "جرس"، "جرش"، "جرم"، "جزف"، "جزل"، "جزم"، "جفل"، "جلز"، ","جلسز"، "همس"، "همش"، "همط"}
3.	// root 3 eg بَعَثَ - يَبْعَثُ	$roots3 = {"بعث"، "بتع"، "بحث"، "بحر"، "بخر"، "بخس"، "بخص"، "بخع"، "بخق"، "بدح"، "بدع"، "بعط"، "بعق"، " ,"بده"، "بذح"، "برع"، "بزخ"، "بصع"، "بطح"، "بعج"، "بعر"، "بعص"، "بعض"، "بعك"، "بعل" ، "بغت"، "بغر"، "بغز"، "بغش"، "بغم"، "بكع"، "بلت"، "بلح"، "بلع"، "بهت"، "بهث"، "نهق"، "نهك"، "هبغ"، "هجع"، "هدغ"، "هرق"، " ,"بهج "، "نهس"، "نهش"، "نهض"، "نهط"، "نهف"، "هزع"، "هطع"، "هقع"، "هكع"، "هلك"، "همع"، "همغ"، "هنع"، "هنغ"}
4.	// root 4 eg فَرِحَ - يَفْرَحُ	$roots4 = {"فرح"، "بثر"، "بثع"، "بجح"، "بجر"، "بخل"، "بدل"، "برش"، "برص"، "برغ"، "بره"، "بهق"، " ,"بشع"، "بشم"، "بصر"، "بظر"، "بعد"، "بغث"، "بقع"، "بكم"، "بلخ"، "بلم"، "بله"، "بهص"، "تبع"، "تخذ"، "ترح"، "ترف"، "تره"، "تعب"، "تغب" "نهل"، "هبل"، "هرص"، "هرع"، "هزج"، "هزق"، "هقف"، "هقم"، "هكر"، "هلع"، "هلم"، "هنب"، "هنق"}
5.	// root 9 eg ضَلَّ - يَضِلُّ	$roots9 = {"ضلل"، "بتت"، "برر"، "بصص"، "بقق"، "بلل"، "بنن"، "تبب"، "تخخ"، "ترر"، "تكك"، "تمم"، "ثبب"، "ثجج"، "ثطط"، "ثعع"، "جدد"، "جنن"، "حذذ"، "حرر "، "ذمم"، "رفف "، "نطط"، "نعع"، "نقق"، "هتت"، "هجج"}

Appendix III: Additional Detailed Informations, Tables and Figures of Implemented Root Extraction Techniques in Chapter 4

Samples of Accuracy Results for All Ten Algorithms using AT8 corpus

Category: Politics

Name of Text *.txt	S1 (%)	S1_corr (%)	S2 (%)	S2_corr (%)	S3 (%)	S3_corr (%)	S4 (%)	S4_corr (%)	RB (%)	Enh_RB (%)
alqabas1	53.90	60.86	58.54	67.66	57.05	63.02	57.05	66.83	57.38	70.98
alqabas2	60.87	65.22	66.96	73.48	61.30	67.39	61.52	68.48	66.30	73.04
ahram1	60.30	66.25	65.51	77.67	64.52	73.20	62.78	74.69	60.30	76.92
addustour1	57.98	63.81	62.65	69.26	59.53	64.59	57.98	68.48	61.87	71.21
alwafd2	60.42	63.54	69.79	76.04	60.42	64.58	61.46	73.96	57.29	69.79
maktoob1	53.37	60.80	58.87	69.60	56.53	65.06	55.98	64.92	53.65	70.15

Category: Economics

Name of Text *.txt	S1 (%)	S1_corr (%)	S2 (%)	S2_corr (%)	S3 (%)	S3_corr (%)	S4 (%)	S4_corr (%)	RB (%)	Enh_RB (%)
ahram2	59.07	66.51	67.91	77.21	63.26	72.56	61.86	72.09	63.72	77.21
addustour2	52.16	60.30	59.80	69.98	57.76	67.68	55.98	71.25	54.45	70.74
alwatan5	55.66	60.70	65.18	73.55	63.76	69.88	63.46	71.71	63.76	73.85
cnn9	56.63	63.25	58.43	67.47	56.33	60.84	58.13	64.76	56.93	70.18
okaz4	44.04	51.38	49.54	63.30	49.54	61.47	42.20	58.72	44.95	60.55

Category: Religious issues

Name of Text *.txt	S1 (%)	S1_corr r (%)	S2 (%)	S2_corr r (%)	S3 (%)	S3_corr r (%)	S4 (%)	S4_corr (%)	RB (%)	Enh_RB (%)
addustour3	53.49	63.00	63.85	75.48	62.16	71.04	61.10	71.67	64.48	79.49
al-madina9	53.13	60	62.02	73.13	60.40	67.07	62.83	71.31	61.82	74.95
alqabas11	60.61	64.99	66.74	73.30	61.27	67.40	61.71	68.71	66.30	73.09
bbc16	55.68	63.74	65.93	76.56	64.84	70.70	60.07	69.23	55.31	70.33
alquds-alarabi4	59.23	65.24	63.95	72.53	62.66	68.24	55.79	68.67	58.16	72.75

Category: Educational And Health

Name of Text *.txt	S1 (%)	S1_corr r (%)	S2 (%)	S2_corr r (%)	S3 (%)	S3_corr r (%)	S4 (%)	S4_corr r (%)	RB (%)	Enh_RB (%)
ahram5	52.75	64.90	60	75.88	55.69	66.08	54.71	66.67	53.33	71.18
ahram6	57.83	66.86	63.86	74.70	66.27	72.89	56.02	66.27	69.88	82.53
alquds10	60.42	65.28	64.58	72.22	65.28	73.61	59.72	70.83	62.5	75
alquds11	55.25	62.98	62.98	73.48	65.75	70.72	54.14	68.51	59.67	76.80
alquds12	60	62.73	72.73	79.09	72.73	75.45	62.27	80.91	65.45	79.09

Category: Social

Name of Text *.txt	S1 (%)	S1_corr r (%)	S2 (%)	S2_corr r (%)	S3 (%)	S3_corr r (%)	S4 (%)	S4_corr r (%)	RB (%)	Enh_RB (%)
addustour5	56.39	67.12	57.31	71.01	54.57	64.61	58.68	68.49	56.39	71.01
al-fadjr6	51.28	58.69	60.11	70.94	58.41	64.94	57.27	65.53	58.69	72.08
al-fadjr7	61.78	68.15	63.38	72.93	62.10	68.47	61.15	73.25	66.24	77.07
alkhabar12	55.84	64.04	60.88	71.29	58.36	65.30	57.41	66.88	62.46	75.39
cnn10	50	60.09	58.26	67.89	49.54	58.26	54.59	68.81	59.63	75.69
alqabas12	57.27	64.55	61.82	74.55	60.91	69.09	55.45	70.91	58.18	80

Category: Music

Name of Text *.txt	S1 (%)	S1_corr (%)	S2 (%)	S2_corr r (%)	S3 (%)	S3_corr (%)	S4 (%)	S4_corr (%)	RB (%)	Enh_RB (%)
addustour12	48.86	62.60	58.02	72.52	56.49	67.94	59.54	70.99	51.91	73.28
al-fadjr10	54.07	61.05	59.30	68.61	55.81	64.53	65.12	71.51	57.56	78.49
el-massa2	54.52	61.56	62.81	71.11	59.80	68.82	60.30	68.34	58.54	77.39
alyaum2	62.10	70.16	70.16	81.45	66.94	75	75.81	82.26	62.10	74.19
alqabas10	56.30	66.87	60.34	72.16	60.34	67.34	58.63	67.03	62.99	73.87

Category: Sports

Name of Text *.txt	S1 (%)	S1_corr r (%)	S2 (%)	S2_corr r (%)	S3 (%)	S3_corr r (%)	S4 (%)	S4_corr r (%)	RB (%)	Enh_ RB

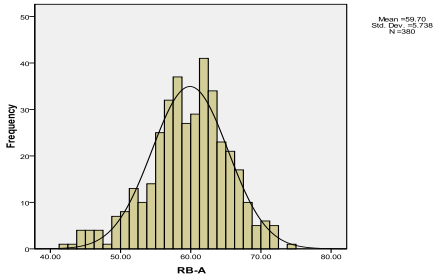
ahram3	**58.96**	**66.23**	**67.53**	**75.84**	**58.18**	**66.49**	**58.96**	**67.01**	**60.78**	**68.83**
ahram10	**57.58**	**65.15**	**65.15**	**74.62**	**58.71**	**64.52**	**58.71**	**67.80**	**59.72**	**72.48**
alanwar2	**57.59**	**63.80**	**64.90**	**73.31**	**59.60**	**66.36**	**57.95**	**67.82**	**52.65**	**67.46**
alqabas7	**52.88**	**60.47**	**56.55**	**67.02**	**50.26**	**59.42**	**52.62**	**61.78**	**54.97**	**67.02**
ommandaily3	**69.79**	**75.51**	**69.18**	**73.72**	**60.12**	**64.35**	**66.47**	**70.09**	**64.95**	**74.92**
cnn1	**52.69**	**58.60**	**66.16**	**74.19**	**60.75**	**66.13**	**53.76**	**69.89**	**65.52**	**70.97**
assaheefa1	**51.84**	**55.88**	**55.88**	**59.93**	**51.10**	**57.35**	**55.52**	**63.24**	**60.29**	**72.43**

Category: Art, Culture and Literature

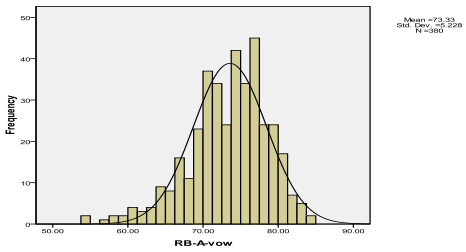
Name of Text *.txt	**S1 (%)**	**S1_cor r (%)**	**S2 (%)**	**S2_cor r (%)**	**S3 (%)**	**S3_cor r (%)**	**S4 (%)**	**S4_cor r (%)**	**RB (%)**	**Enh_RB (%)**
ahram4	**52.51**	**59.32**	**61.59**	**71.31**	**58.83**	**65.80**	**58.67**	**69.04**	**64.51**	**73.26**
alalam3	**48.28**	**51.72**	**54.02**	**63.22**	**56.32**	**57.47**	**55.17**	**60.92**	**65.52**	**78.16**
azzaman5	**58.17**	**64.78**	**64.92**	**73.09**	**62.74**	**69.21**	**60.49**	**69.62**	**64.24**	**76.43**
alquds-alarabi8	**55.11**	**63.78**	**60.68**	**72.14**	**58.36**	**65.65**	**60.22**	**71.36**	**56.81**	**68.89**
jeeran1	**50.43**	**53.85**	**64.10**	**67.95**	**61.97**	**62.39**	**58.55**	**62.82**	**55.13**	**64.53**

Analysis using SPSS for rule_based, Enh_rule_based, Adjusted Al-Shalabi, Adjusted Al-Shalabi-corr algorithms (All categories)

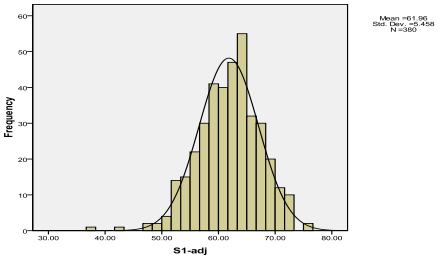
Histogram for rule_based algorithm:



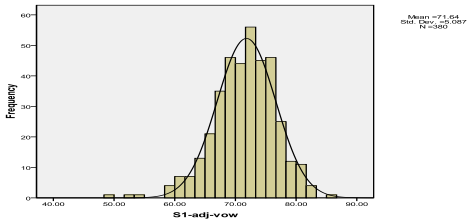
Histogram for Enh_rule_based algorithm:



Histogram for Adjusted Al-Shalabi:



Histogram for Adjusted Al-Shalabi-corr:



Model Description

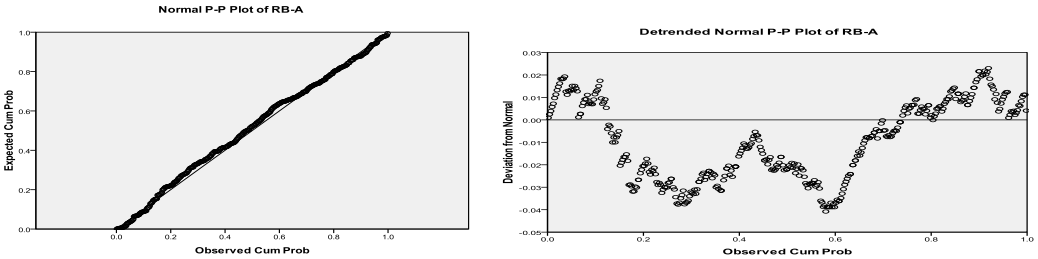
Model Name		**MOD_1**	**MOD_2**	**MOD_3**	**MOD_4**
Series or Sequence	1	**RB**	**Enh_RB**	**S2**	**S2-corr**
Transformation		None	None	None	None
Non-Seasonal Differencing		0	0	0	0
Seasonal Differencing		0	0	0	0
Length of Seasonal Period		No periodicity	No periodicity	No periodicity	No periodicity
Standardization		Not applied	Not applied	Not applied	Not applied
Distribution	**Type**	Normal	Normal	Normal	Normal
	Location	estimated	estimated	estimated	estimated
	Scale	estimated	estimated	estimated	estimated
Fractional Rank Estimation Method		Blom's	Blom's	Blom's	Blom's
Rank Assigned to Ties		Mean rank of tied values	Mean rank of tied values	Mean rank of tied values	Mean rank of tied values
Applying the model		MOD_1	MOD_2	MOD_3	MOD_4

specifications from				

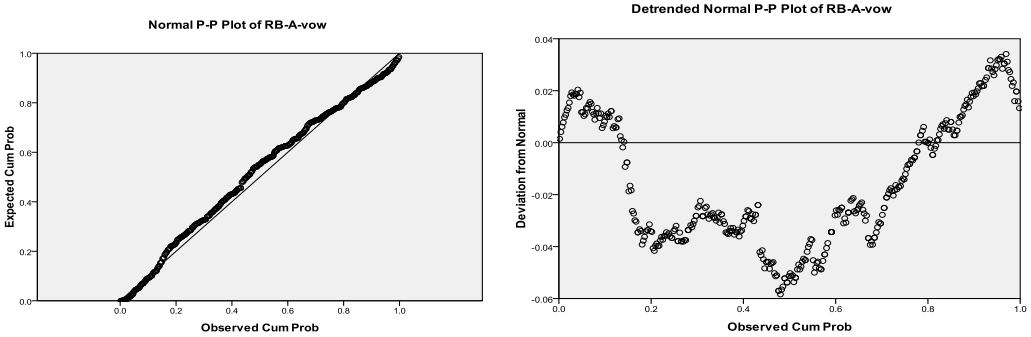
Case Processing Summary					
		RB	**Enh_RB**	**S2**	**S2-corr**
Series or Sequence Length		380	380	381	381
Number of Missing Values in the Plot	**User-Missing**	0	0	0	0
	System-Missing	0	0	1	1
The cases are unweighted					

Estimated Distribution Parameters					
		RB	**Enh_RB**	**S2**	**S2-corr**
Normal Distribution	Location	59.7016	73.3325	61.9559	71.6429
	Scale	5.73808	5.22792	5.45823	5.08720
The cases are unweighted.					

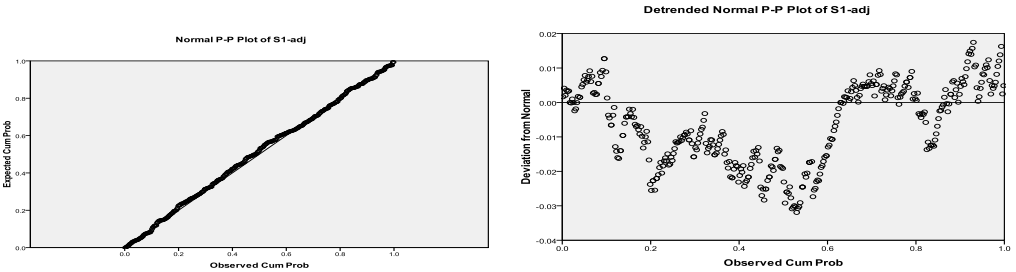
P-P plot for Rule-based algorithm:



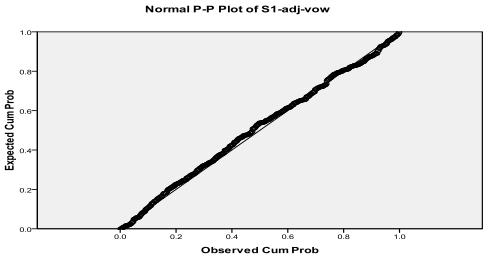
P-P plot for Enh_Rule-based algorithm:

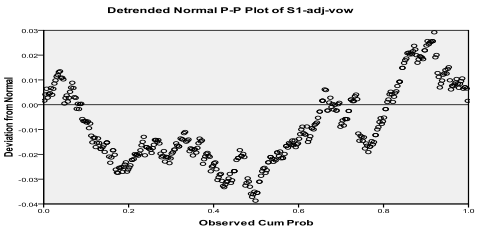


P-P plot for Adjusted Al-Shalabi algorithm:



P-P plot for Adjusted Al-Shalabi-corr algorithm:





K-S (NORM) test:

One-Sample Kolmogorov-Smirnov Test					
		RB	**Enh_RB**	**S2**	**S2-corr**
N		380	380	380	380
Normal Parameters[a,,b]	**Mean**	59.7016	73.3325	61.9559	71.6429
	Std. Deviation	5.73808	5.22792	5.45823	5.08720
Most Extreme Differences	**Absolute**	0.042	0.060	0.033	0.040
	Positive	0.025	0.036	0.019	0.031
	Negative	-0.042	-0.060	-0.033	-0.040
Kolmogorov-Smirnov Z		0.820	1.164	0.648	0.780
Asymp. Sig. (2-tailed)		0.511	0.133	0.796	0.578
a. Test distribution is Normal.					
b. Calculated from data.					

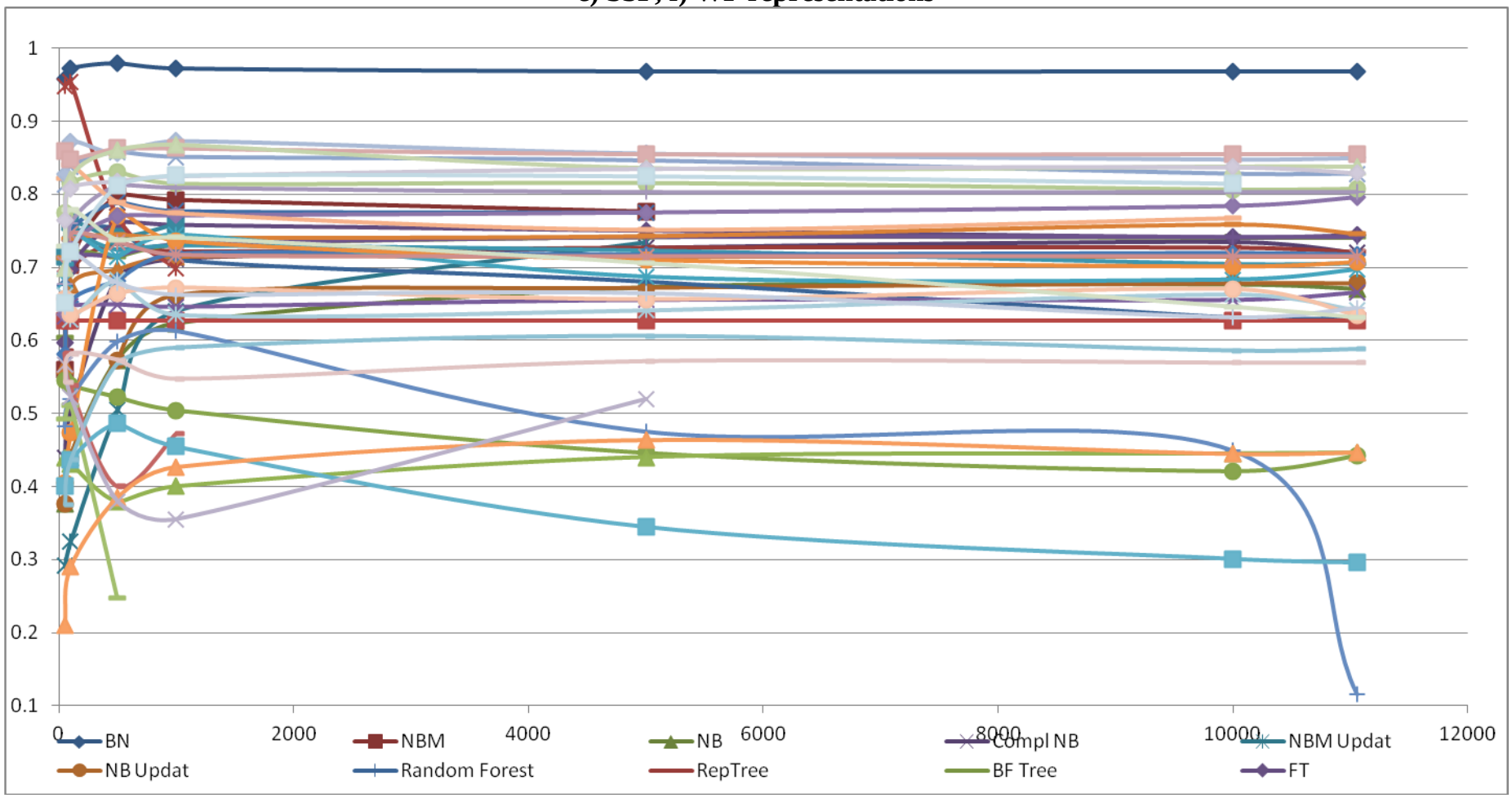
Appendix IV: Additional Detailed Informations, Tables and Figures of Chapter 5

Criteria of Applied Classifiers here in WEKA

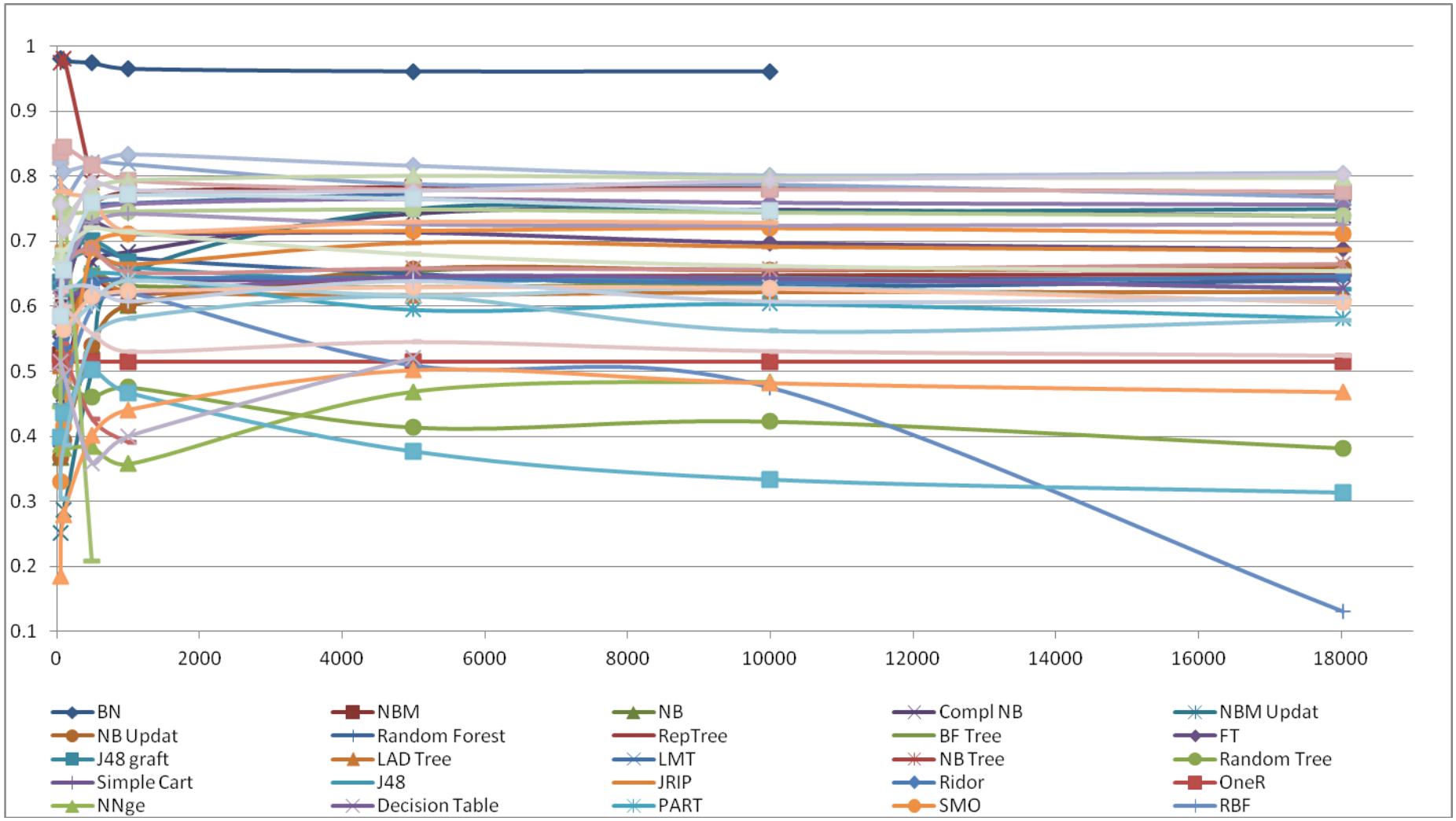
Type	Classifier	Criteria
Bayes-Based Learners	**BN**	uses various search algorithms and quality measures. debug = F, *estimator = SimpleEstimator A-0.5, searchAlgorithm = k2-P1-S Bayes*, useAD Tree = F.
	NBM	Class for building and using a multinomial Naive Bayes classifier. debug = F.
	NB	Class for a Naive Bayes classifier using estimator classes. debug = F, *displayModelInOldFormat = F, useKernelEstimator = F, useSupervisedDiscritization = F*.
	Complement NB	builds and uses a Complement class Naive Bayes classifier. *debug = F, normalizeWordWeights = F, smoothingParameter = 1*.
	NBMU	As NBM
	NBU	As NB
Tree Learners	**Random Forest**	Class for constructing a forest of random trees. debug = F, maxDepth = 0, numFeatures = 0, numTrees = 10, seed = 1.
	RepTree	Fast decision tree learner. debug = F, maxDepth = -1, minNum = 2, minVarianceProp = 0.0010, noPruning = F, numFolds = 3, seed = 1.
	BF Tree	Class for building a best-first decision tree classifier. debug = F, heurestic = T, minNumObj = 2, numFoldsPruning = 5, pruningStrategy = Post-Pruning, seed = 1, sizePer = 1, useErrorRate = T, useGini = T, useOneSE = F.
	NB Tree	Class for generating a decision tree with naive Bayes classifiers at the leaves., debug = F.
	FT	Classifier for building 'Functional trees', which are classification trees that could have logistic regression functions at the inner nodes and/or leaves. binSplit = F, debug = F, errorOnProbabilities = F, minNumInstances = 15, modelType = FT, numBoostingIterations = 15, useAIC = F, weightTrimBeta = 0.
	J48	Class for generating a pruned or unpruned C4. binarySplits = F, confidenceFactor = 0.25, debug = F, minNumObj = 2, numFolds =3, reducedErrorPruning = F, saveInstanceData = F, seed = 1, subTraaRaising = T, unpruned = F, useLaplace = F.
	J48 graft	Class for generating a grafted pruned or unpruned C4. (as J48).
	LAD Tree	Class for generating a multi-class alternating decision tree using the LogitBoost strategy. dbug = F, numOfBoostingIterations = 10.
	LMT	Classifier for building 'logistic model trees', which are classification trees with logistic regression functions at the leaves. convertNominal = F, debug = F, errorOnProbabilities = F, fastRegression = T, minNumInstances = 15, numBoostingIterations = -1, splitOnResiduals = F, useAIC = F, weightTrimBeta = 0.
	Random Tree	Class for constructing a tree that considers K randomly chosen attributes at each node. kValue = 0, allowUnclassifiedInstances = F, debug = F, maxDepth = 0, minNum = 1, numFolds = 0, seed = 1.
	Simple Cart	Class implementing minimal cost-complexity pruning. debug = F, heuristic = T, minNumObj = 2, numFoldsPruning = 5, seed = 1, sizePer = 1, useOneSE = F, usePrune = T.
Rule Learners	**JRip**	This class implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), which was proposed by William W. checkErrorRate = T, debug = F, folds = 3, minNo = 2, optimizations = 2, seed = 1, usePruning = T.
	PART	Class for generating a PART decision list. binarySplits = F, confidenceFactor = 0.25, debug = F, minNumObj = 2, numFolds = 3, reducedErrorPruning = F, seed = 1, unpruned = F.
	Ridor	An implementation of a RIpple-DOwn Rule learner. debug = F, folds = 3, majorityClass = F, minNo = 2, seed = 1, shuffle = 1, wholeDataErr = F.
	OneR	Class for building and using a 1R classifier; in other words, uses the minimum-error attribute for prediction, discretizing numeric attributes. debug = F, minBucketSize = 6.
	NNge	Nearest-neighbor-like algorithm using non-nested generalized exemplars (which are hyperrectangles that can be viewed as if-then rules). debug = F, numAttemptsOfGeneOption = 5, numFoldersMIOption = 5.
	Decision Table	Class for building and using a simple decision table majority classifier. crossVal = 1, debug = F, displayRules = F, evaluationMeasure = Default: accuracy(discrete class); RMSE (numeric class), search = BestFirst -D 1-N 5, useIbk = F.
	SMO	Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier. buildLogisticModel = F, c = 1, checksTurnedOff = F, debug = F, epsilon = 1.0E-12, filterType = Normalize training data, kernel = Poly kernel-C 250007-E1.0, numFolds = -1, randomSeed = 1, toleranceParameter = 0.0010.

Function Learners	**RBF**	Class that implements a normalized Gaussian radial basis function network. clusteringSeed = 1, dbug = F, maxIts = -1, minStdDev = 0.1, numClusters = 8, ridge = 1.0E-8.
	Logistic	Class for building and using a multinomial logistic regression model with a ridge estimator. debug = F, maxIts = -1, ridge = 1.0E-8.
	MLP	A Classifier that uses backpropagation to classify instances. GUI = F, autoBuild = T, debug = F, decay = F, hiddenLayers = a, learningRate = 0.3, momentum = 0.2, nominalToBinaryFilter = T, normalizeAttributes = T, NormalizeNumericClass = T, reset = T, seed = 0, trainingTime = 500, validationSetSize = 0, validationThreshold = 20.
	Simple Logistic	Classifier for building linear logistic regression models. debug = F, errorOnProbabilities = F, heuristicStop = 50, maxBoostingIterations = 500, numBoostingIterations = 0, useAIC = F, useCrossValidation = T, weightTrimBeta = 0.
Miscillaneous Learners	**VFI**	Classification by voting feature intervals. bias = 0.6, debug = F, weightByConfidence = T.
	Hyper Pipes	Class implementing a HyperPipe classifier. debug = F.
Meta Learners	**AdaBoost.M1**	Class for boosting a nominal class classifier using the Adaboost M1 method. *classifier = J48 -C 0.25 -M 2*, debug = F, numIterations = 10, seed = 1, useResampling = F, weightThreshold = 100.
	Attribute Selected Classifier	Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier. *classifier = J48 -C 0.25 -M 2*, debug = F, evaluator = CfsSubsetEval, search = BestFirst - D 1- N 5.
	Bagging	Class for bagging a classifier to reduce variance. bagSizePercent = 100, calOutOfBag = F, *classifier = J48 -C 0.25 -M 2*, debug = F, numIterations = 10, seed = 1.
	Classification Via Regression	Class for doing classification using regression methods. *classifier = M5P --M 4.0*, debug = F.
	Dagging	This meta classifier creates a number of disjoint, stratified folds out of the data and feeds each chunk of data to a copy of the supplied base classifier. *classifier = SMO -C 1.0 - L 0.0010 - P 1.0E-12 -N 0 -V -1*, debug = F, numFolds = 10, seed = 1, verbose = F.
	Decorate	is a meta-learner for building diverse ensembles of classifiers by using specially constructed artificial training examples. Artificial Size = 1, *classifier = J48 - C0.25-M2*, debug = F, desiredSize = 10, numIterations = 10.
	END	A meta classifier for handling multi-class datasets with 2-class classifiers by building an ensemble of nested dichotomies. *classifier = ND -S1 -W weka.classifiers.trees.j48-- C 0.25 M 2*, debug = F, numIterations = 10, seed = 1.
	Filtered Classifier	Class for running an arbitrary classifier on data that has been passed through an arbitrary filter. *classifier = J48 -C 0.25 -M 2*, debug = F, filter = Discritize -R first-last.
	Logit Boost	Class for performing additive logistic regression. *classifier = DecisionStump*, dbug = F, likelihoodThreshold = -1.7976931348623157E308, numFolds = 0, numIterations = 10, numRuns = 1, seed = 1, shrinkage = 1.0, useResampling = F, weightThreshold = 100.
	Multi Class Classifier	A metaclassifier for handling multi-class datasets with 2-class classifiers. *classifier = Logistic -R 1.0E-8 -M -1*, debug = F, method = 1-against-all, randomWidthFactor = 2.0, seed = 1, usePairwiseCoupling = F.
	CBND	A meta classifier for handling multi-class datasets with 2-class classifiers by building a random class-balanced tree structure. (as ND).
	DNBND	A meta classifier for handling multi-class datasets with 2-class classifiers by building a random data-balanced tree structure. (as ND)
	ND	A meta classifier for handling multi-class datasets with 2-class classifiers by building a random tree structure. *classifier = J48 -C 0.25 -M 2*, debug = F, seed = 1.
	Ordinal Class Classifier	Meta classifier that allows standard classification algorithms to be applied to ordinal class problems. *classifier = J48 -C 0.25 -M 2*, debug = F.
	Random Committee	Class for building an ensemble of randomizable base classifiers. *classifier = RandomTree - k 0 -M 1.0 -S1*, debug = F, numIterations = 10, seed = 1.
	Random SubSpace	This method constructs a decision tree based classifier that maintains highest accuracy on training data and improves on generalization accuracy as it grows in complexity. *classifier = RepTree - M2 -V 0.0010 - N3 - S1 -L-1*, debug = F, numIterations = 10, seed = 1, subSpaceSize = 0.5.
	Rotation Forest	Class for construction a Rotation Forest. *classifier = J48 -C 0.25 -M 2*, debug = F, maxGroup = 3, minGroup = 3, numIterations = 10, numberOfGroups = F, *projectionFilter = PrincipleComponents -R 1.0 -A5 -M-1*, removedPercentage = 50, seed = 1.

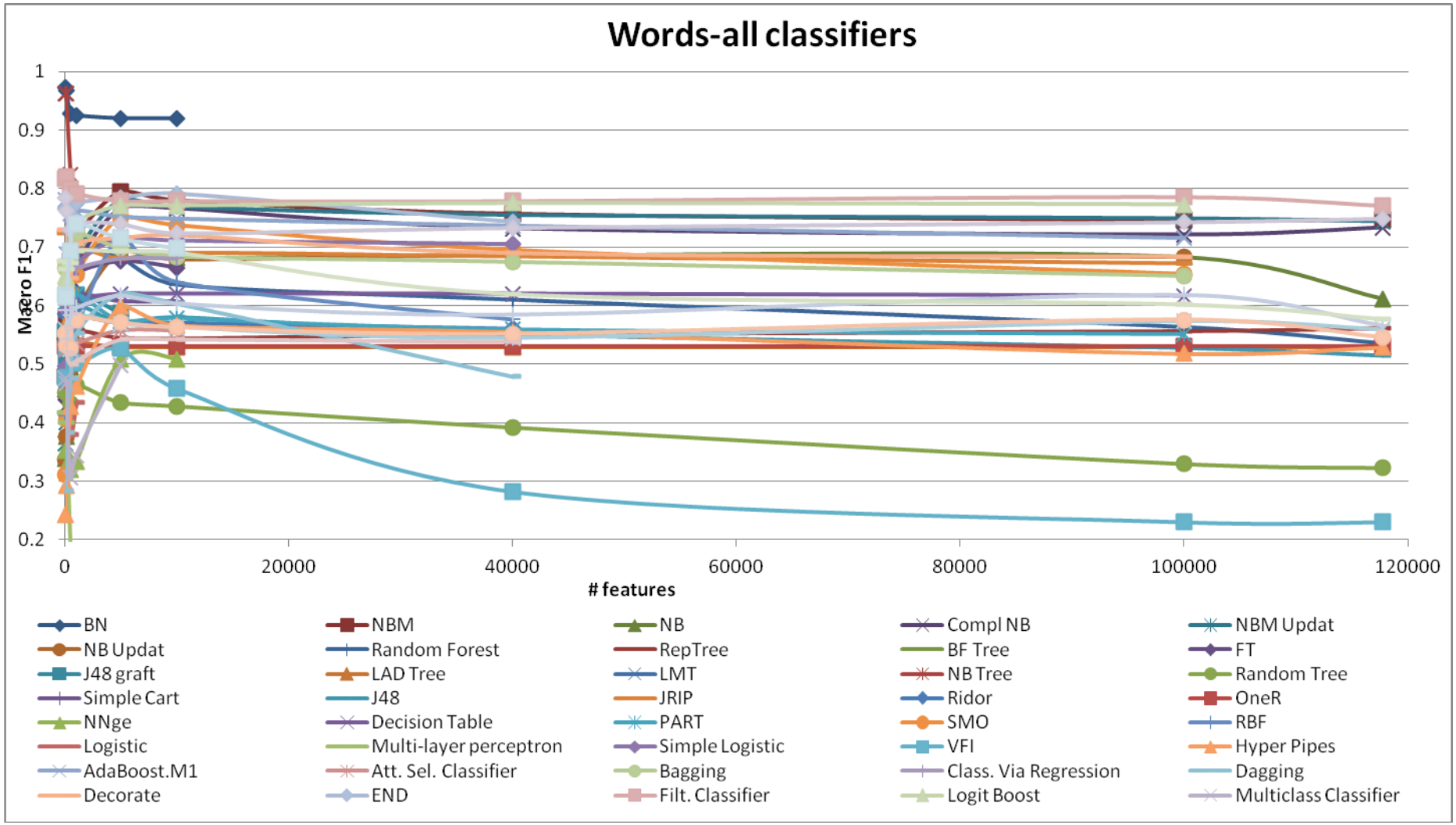
Performance of all classifiers of corpus as number of selected features increased among a) Roots, b) Stems, c) Words, d) RRP, e) SSP, f) WP representations



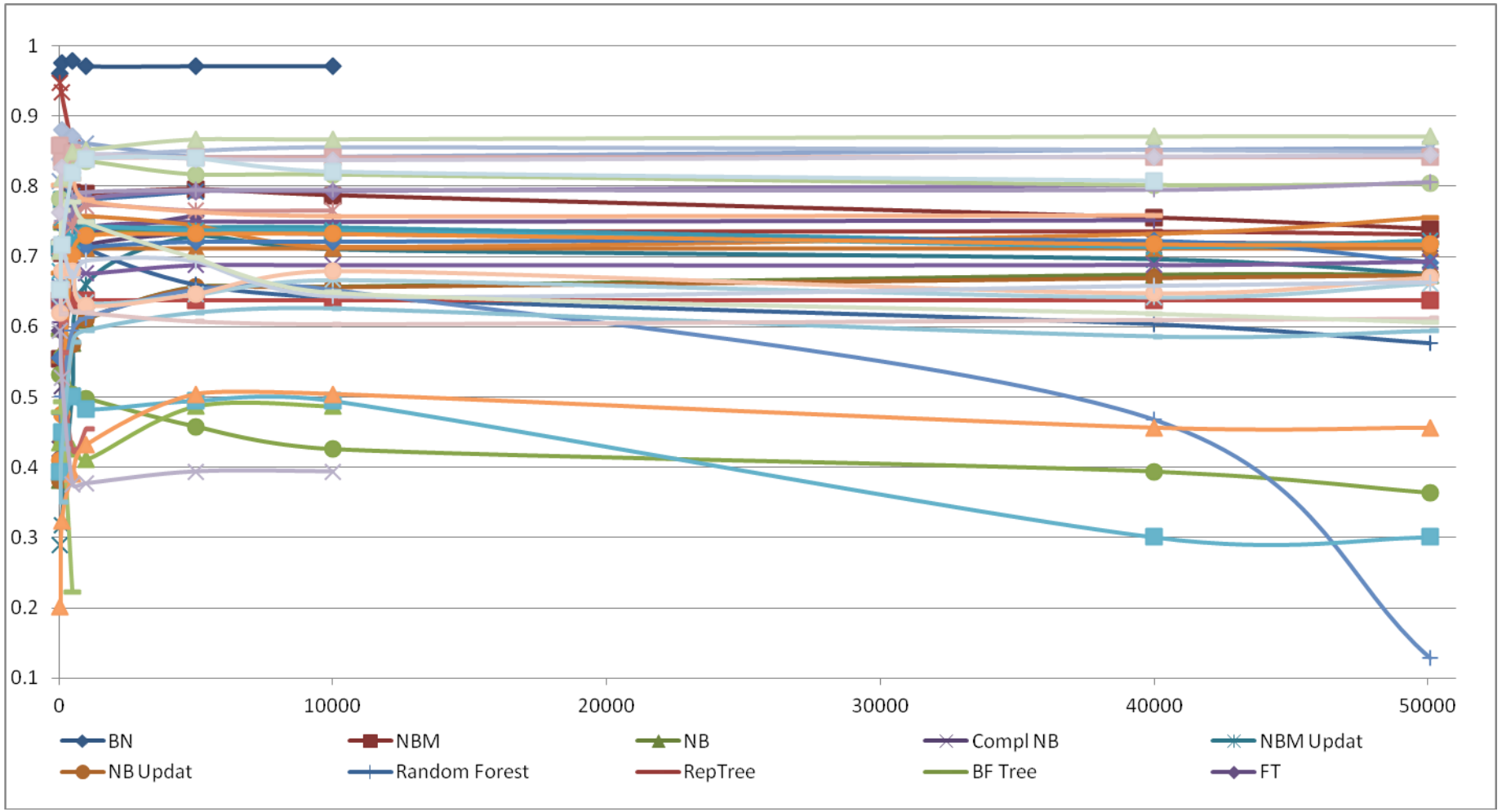
(a)


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
BN
NBM
NB
Compl NB
NBM Updat
NB Updat
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree
Random Tree
Simple Cart
J48
JRIP
Ridor
OneR
NNge
Decision Table
PART
SMO
RBF

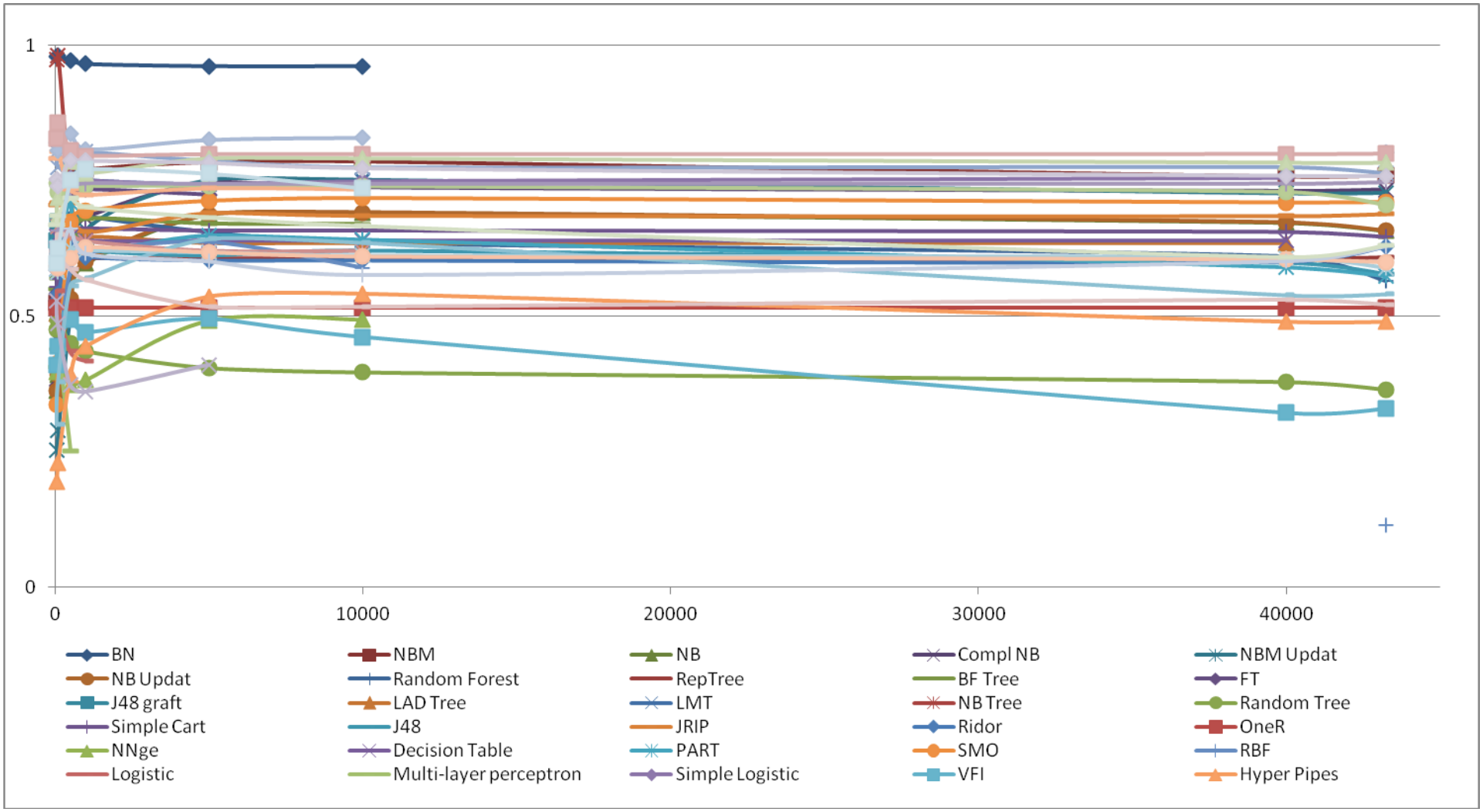
(b)


Words-all classifiers
Macro F1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20000
40000
60000
80000
100000
120000
features
BN
NBM
NB
Compl NB
NBM Updat
NB Updat
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree
Random Tree
Simple Cart
J48
JRIP
Ridor
OneR
NNge
Decision Table
PART
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic
VFI
Hyper Pipes
AdaBoost.M1
Att. Sel. Classifier
Bagging
Class. Via Regression
Dagging
Decorate
END
Filt. Classifier
Logit Boost
Multiclass Classifier

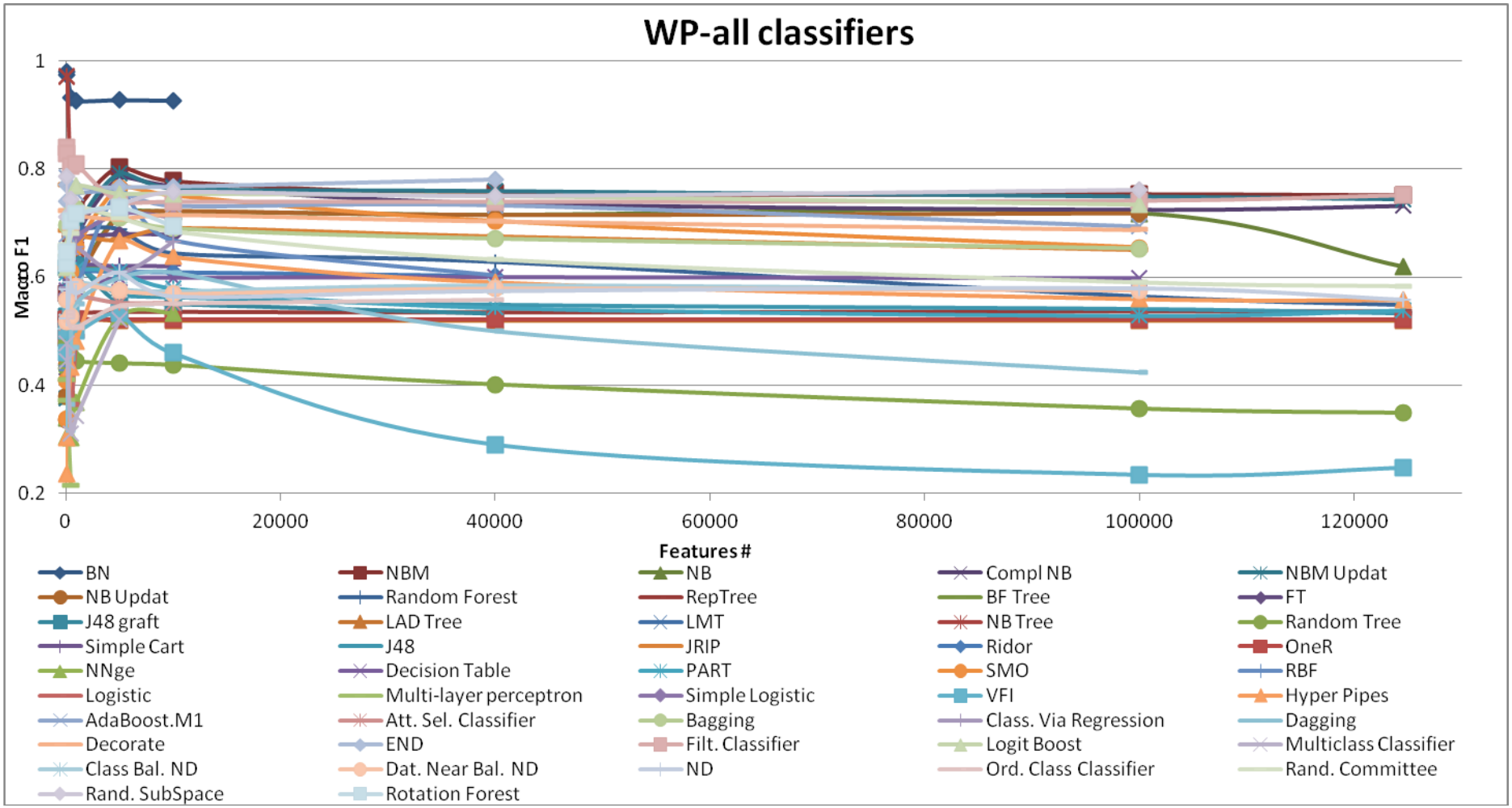
(c)


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10000
20000
30000
40000
50000
BN
NBM
NB
Compl NB
NBM Updat
NB Updat
Random Forest
RepTree
BF Tree
FT

(d)


1
0.5
0
0
10000
20000
30000
40000
BN
NBM
NB
Compl NB
NBM Updat
NB Updat
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree
Random Tree
Simple Cart
J48
JRIP
Ridor
OneR
NNge
Decision Table
PART
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic
VFI
Hyper Pipes

(e)


WP-all classifiers
1
0.8
0.6
0.4
0.2
Macro F1
0
20000
40000
60000
80000
100000
120000
Features #
BN
NBM
NB
Compl NB
NBM Updat
NB Updat
Random Forest
RepTree
BF Tree
FT
J48 graft
LAD Tree
LMT
NB Tree
Random Tree
Simple Cart
J48
JRIP
Ridor
OneR
NNge
Decision Table
PART
SMO
RBF
Logistic
Multi-layer perceptron
Simple Logistic
VFI
Hyper Pipes
AdaBoost.M1
Att. Sel. Classifier
Bagging
Class. Via Regression
Dagging
Decorate
END
Filt. Classifier
Logit Boost
Multiclass Classifier
Class Bal. ND
Dat. Near Bal. ND
ND
Ord. Class Classifier
Rand. Committee
Rand. SubSpace
Rotation Forest

(f)

Performance of most Classifiers among categories for all representations at 1000 best selected features

	BN, F1 values for 1000 features						NB, F1 values for 1000 features						NBM, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.986	0.986	0.972	0.986	0.986	0.972	0.675	0.63	0.486	0.663	0.562	0.519	0.779	0.722	0.659	0.771	0.718	0.679
Religious	0.956	0.956	0.915	0.949	0.955	0.915	0.406	0.371	0.433	0.333	0.377	0.441	0.648	0.649	0.658	0.657	0.636	0.645
Arts	0.988	0.988	0.973	0.988	0.988	0.976	0.678	0.663	0.681	0.657	0.688	0.714	0.793	0.783	0.762	0.796	0.779	0.771
Social	0.988	0.976	0.968	0.988	0.976	0.964	0.406	0.39	0.403	0.414	0.362	0.385	0.627	0.602	0.47	0.615	0.606	0.457
Economics	0.986	0.979	0.902	0.986	0.979	0.909	0.556	0.52	0.457	0.586	0.495	0.436	0.846	0.782	0.671	0.824	0.792	0.686
Sports	0.909	0.844	0.643	0.935	0.844	0.653	0.806	0.784	0.701	0.803	0.835	0.672	0.934	0.927	0.869	0.943	0.918	0.893
Music	0.916	0.942	0.968	0.875	0.951	0.98	0.915	0.86	0.791	0.928	0.867	0.782	1	0.99	0.915	1	0.979	0.896
Educational	0.981	0.975	0.93	0.981	0.978	0.93	0.695	0.694	0.695	0.671	0.693	0.693	0.857	0.863	0.83	0.854	0.856	0.839

	SMO, F1 values for 1000 features						Simple Logistic, F1 values for 1000 features						Decision Table, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.69	0.639	0.604	0.723	0.621	0.562	0.818	0.832	0.649	0.816	0.766	0.667	0.562	0.473	0.488	0.57	0.528	0.503
Religious	0.638	0.57	0.587	0.642	0.519	0.606	0.765	0.69	0.539	0.835	0.675	0.575	0.662	0.564	0.564	0.702	0.569	0.454
Arts	0.754	0.741	0.693	0.744	0.731	0.726	0.761	0.719	0.689	0.752	0.749	0.688	0.628	0.565	0.614	0.67	0.565	0.567
Social	0.591	0.568	0.467	0.576	0.541	0.494	0.551	0.575	0.466	0.581	0.552	0.484	0.579	0.714	0.5	0.68	0.667	0.524
Economics	0.774	0.738	0.667	0.739	0.719	0.634	0.825	0.783	0.647	0.849	0.713	0.567	0.661	0.476	0.614	0.678	0.59	0.564
Sports	0.696	0.693	0.667	0.688	0.703	0.656	0.889	0.912	0.821	0.894	0.956	0.814	0.512	0.532	0.593	0.534	0.619	0.589
Music	0.926	0.94	0.818	0.925	0.922	0.8	0.923	0.96	0.866	0.923	0.96	0.914	0.819	0.619	0.684	0.847	0.624	0.592
Educational	0.848	0.828	0.751	0.838	0.821	0.746	0.822	0.81	0.804	0.814	0.82	0.799	0.743	0.8	0.76	0.718	0.822	0.74

	JRIP, F1 values for 1000 features						PART, F1 values for 1000 features						J48, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.873	0.734	0.699	0.851	0.667	0.624	0.808	0.692	0.532	0.863	0.639	0.513	0.797	0.696	0.477	0.883	0.681	0.474
Religious	0.841	0.611	0.709	0.818	0.646	0.646	0.743	0.619	0.658	0.707	0.537	0.596	0.736	0.602	0.577	0.746	0.536	0.671
Arts	0.63	0.589	0.595	0.661	0.584	0.571	0.712	0.634	0.619	0.704	0.613	0.648	0.667	0.637	0.614	0.674	0.582	0.65
Social	0.603	0.563	0.728	0.632	0.535	0.715	0.603	0.506	0.453	0.579	0.498	0.482	0.552	0.54	0.432	0.579	0.453	0.438
Economics	0.809	0.685	0.563	0.808	0.644	0.565	0.816	0.632	0.475	0.831	0.65	0.5	0.841	0.623	0.455	0.835	0.675	0.452
Sports	0.875	0.829	0.836	0.87	0.809	0.814	0.923	0.815	0.797	0.902	0.853	0.75	0.908	0.842	0.748	0.919	0.853	0.672
Music	0.851	0.878	0.857	0.878	0.827	0.903	0.86	0.835	0.811	0.909	0.84	0.73	0.911	0.884	0.825	0.929	0.874	0.845
Educational	0.738	0.685	0.721	0.782	0.727	0.731	0.725	0.661	0.706	0.722	0.612	0.699	0.731	0.673	0.717	0.697	0.627	0.761

	Random Forest, F1 values for 1000 features						FT, F1 values for 1000 features						RepTree, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.636	0.646	0.551	0.658	0.585	0.561	0.815	0.732	0.662	0.797	0.706	0.646	0.865	0.654	0.549	0.865	0.647	0.507
Religious	0.566	0.55	0.581	0.645	0.606	0.612	0.717	0.615	0.548	0.725	0.611	0.556	0.824	0.589	0.56	0.779	0.593	0.506
Arts	0.739	0.723	0.753	0.726	0.735	0.734	0.738	0.675	0.703	0.698	0.736	0.691	0.639	0.603	0.621	0.65	0.589	0.566
Social	0.581	0.524	0.571	0.641	0.548	0.525	0.58	0.496	0.471	0.554	0.569	0.476	0.567	0.415	0.359	0.559	0.425	0.422
Economics	0.693	0.624	0.621	0.634	0.608	0.625	0.791	0.748	0.618	0.797	0.733	0.644	0.803	0.63	0.378	0.809	0.611	0.321
Sports	0.853	0.837	0.851	0.862	0.857	0.837	0.871	0.91	0.776	0.863	0.892	0.835	0.914	0.879	0.654	0.928	0.827	0.657
Music	0.97	0.949	0.884	0.98	0.949	0.839	0.906	0.941	0.752	0.841	0.923	0.793	0.918	0.9	0.723	0.928	0.882	0.66
Educational	0.764	0.697	0.746	0.704	0.712	0.775	0.81	0.797	0.747	0.812	0.824	0.784	0.643	0.602	0.648	0.728	0.598	0.622

	BF Tree, F1 values for 1000 features						Random Tree, F1 values for 1000 features						Decorate, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.863	0.726	0.533	0.865	0.689	0.553	0.524	0.731	0.344	0.456	0.315	0.413	0.826	0.707	0.654	0.823	0.699	0.667
Religious	0.793	0.522	0.548	0.825	0.577	0.588	0.552	0.577	0.433	0.356	0.268	0.397	0.813	0.687	0.651	0.763	0.675	0.605
Arts	0.635	0.588	0.667	0.644	0.636	0.671	0.523	0.59	0.558	0.51	0.516	0.513	0.75	0.717	0.71	0.739	0.705	0.731
Social	0.55	0.478	0.392	0.543	0.598	0.49	0.259	0.486	0.316	0.361	0.335	0.332	0.649	0.586	0.578	0.69	0.638	0.659
Economics	0.803	0.642	0.511	0.818	0.667	0.444	0.432	0.656	0.323	0.44	0.301	0.288	0.761	0.676	0.613	0.797	0.718	0.636
Sports	0.91	0.887	0.627	0.881	0.863	0.627	0.471	0.881	0.609	0.713	0.532	0.631	0.896	0.871	0.872	0.916	0.881	0.813
Music	0.911	0.871	0.734	0.92	0.9	0.879	0.32	0.843	0.587	0.863	0.685	0.624	0.939	0.9	0.837	0.939	0.874	0.874
Educational	0.725	0.632	0.629	0.702	0.705	0.692	0.505	0.644	0.522	0.513	0.518	0.44	0.764	0.735	0.787	0.773	0.751	0.753

	Rotation Forest, F1 values for 1000 features						Ridor, F1 values for 1000 features						OneR, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.899	0.789	0.694	0.902	0.797	0.671	0.904	0.696	0.62	0.871	0.684	0.564	0.823	0.481	0.556	0.791	0.519	0.556
Religious	0.863	0.748	0.765	0.859	0.72	0.743	0.775	0.645	0.614	0.736	0.67	0.636	0.649	0.511	0.486	0.635	0.496	0.486
Arts	0.797	0.754	0.716	0.818	0.742	0.747	0.607	0.61	0.635	0.637	0.581	0.669	0.703	0.563	0.63	0.728	0.546	0.548
Social	0.685	0.597	0.586	0.715	0.616	0.516	0.533	0.509	0.417	0.559	0.402	0.522	0.627	0.607	0.38	0.676	0.593	0.43
Economics	0.895	0.844	0.705	0.878	0.855	0.616	0.822	0.569	0.477	0.834	0.582	0.42	0.591	0.388	0.404	0.554	0.306	0.408
Sports	0.955	0.905	0.855	0.963	0.938	0.828	0.94	0.849	0.709	0.894	0.845	0.63	0.41	0.354	0.286	0.41	0.415	0.282
Music	0.929	0.887	0.854	0.938	0.902	0.893	0.929	0.86	0.636	0.939	0.909	0.774	0	0	0.537	0	0.154	0.551
Educational	0.801	0.811	0.831	0.824	0.8	0.798	0.667	0.645	0.614	0.637	0.57	0.644	0.742	0.681	0.71	0.769	0.684	0.71

	LAD Tree, F1 values for 1000 features						Simple Cart, F1 values for 1000 features						Complement NB, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.819	0.583	0.538	0.825	0.638	0.5	0.83	0.731	0.623	0.848	0.699	0.581	0.712	0.713	0.637	0.729	0.694	0.663
Religious	0.792	0.63	0.616	0.791	0.641	0.585	0.824	0.577	0.566	0.848	0.615	0.591	0.563	0.538	0.641	0.4	0.524	0.653
Arts	0.654	0.631	0.647	0.627	0.633	0.571	0.653	0.59	0.661	0.667	0.625	0.688	0.733	0.718	0.731	0.674	0.71	0.74
Social	0.516	0.357	0.235	0.557	0.378	0.351	0.553	0.486	0.449	0.555	0.544	0.47	0.517	0.435	0.328	0.591	0.465	0.331
Economics	0.847	0.667	0.412	0.841	0.713	0.395	0.8	0.656	0.593	0.811	0.698	0.403	0.686	0.649	0.578	0.706	0.63	0.593
Sports	0.779	0.763	0.546	0.767	0.78	0.546	0.902	0.881	0.722	0.902	0.859	0.715	0.908	0.881	0.874	0.725	0.87	0.876
Music	0.905	0.874	0.571	0.891	0.88	0.46	0.893	0.843	0.787	0.911	0.832	0.872	0.936	0.913	0.841	0.833	0.923	0.867
Educational	0.701	0.687	0.677	0.697	0.727	0.671	0.715	0.644	0.644	0.811	0.663	0.667	0.786	0.774	0.79	0.701	0.781	0.794

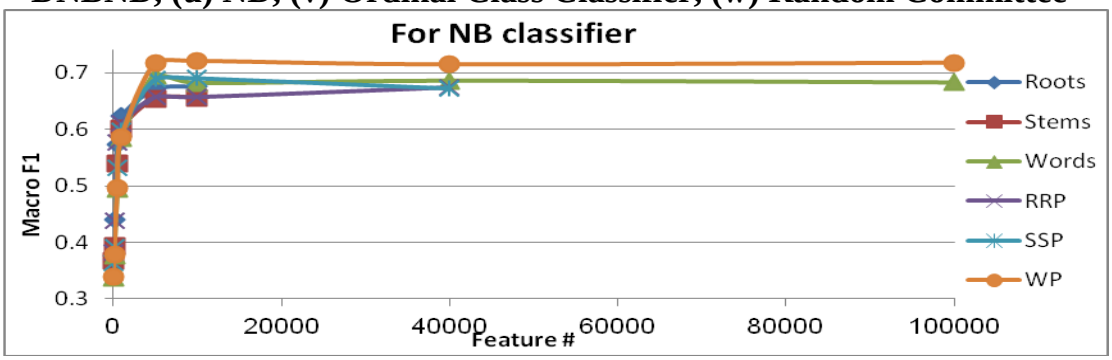
	AdaBoost.M1, F1 values for 1000 features						Attribute Selected Classifier, F1 values for 1000 features						Bagging, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.905	0.841	0.662	0.933	0.848	0.709	0.821	0.743	0.491	0.892	0.623	0.564	0.905	0.754	0.657	0.933	0.725	0.676
Religious	0.857	0.805	0.72	0.877	0.766	0.725	0.76	0.588	0.454	0.807	0.703	0.5	0.843	0.654	0.739	0.859	0.653	0.691
Arts	0.836	0.828	0.794	0.848	0.81	0.793	0.685	0.611	0.618	0.716	0.617	0.622	0.803	0.766	0.745	0.822	0.761	0.793
Social	0.727	0.703	0.621	0.719	0.68	0.614	0.552	0.566	0.403	0.622	0.442	0.447	0.643	0.597	0.541	0.683	0.589	0.544
Economics	0.871	0.826	0.729	0.91	0.811	0.68	0.855	0.614	0.427	0.855	0.658	0.411	0.869	0.731	0.647	0.889	0.717	0.606
Sports	0.963	0.902	0.844	0.962	0.899	0.848	0.882	0.866	0.61	0.901	0.866	0.65	0.928	0.887	0.848	0.914	0.882	0.833
Music	0.959	0.917	0.863	0.959	0.94	0.868	0.893	0.92	0.549	0.911	0.868	0.673	0.938	0.905	0.837	0.938	0.882	0.869
Educational	0.854	0.833	0.857	0.855	0.817	0.827	0.63	0.605	0.634	0.748	0.627	0.632	0.796	0.791	0.818	0.831	0.806	0.813

	Classification Via Regression, F1 values for 1000 features						Dagging, F1 values for 1000 features						END, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.907	0.784	0.699	0.893	0.774	0.634	0.547	0.53	0.528	0.569	0.475	0.491	0.918	0.861	0.691	0.903	0.873	0.733
Religious	0.907	0.78	0.667	0.843	0.725	0.586	0.477	0.469	0.48	0.46	0.381	0.417	0.897	0.831	0.717	0.91	0.81	0.733
Arts	0.774	0.749	0.714	0.755	0.774	0.691	0.694	0.663	0.651	0.671	0.678	0.671	0.847	0.846	0.808	0.802	0.813	0.795
Social	0.65	0.597	0.494	0.624	0.55	0.523	0.38	0.4	0.368	0.433	0.423	0.385	0.795	0.726	0.654	0.738	0.661	0.639
Economics	0.832	0.759	0.533	0.861	0.792	0.595	0.574	0.523	0.451	0.49	0.462	0.433	0.905	0.841	0.772	0.919	0.792	0.657
Sports	0.903	0.836	0.764	0.894	0.825	0.721	0.414	0.451	0.496	0.419	0.447	0.443	0.913	0.924	0.87	0.934	0.892	0.885
Music	0.926	0.891	0.745	0.946	0.907	0.796	0.851	0.754	0.517	0.865	0.778	0.624	0.928	0.928	0.905	0.948	0.929	0.936
Educational	0.795	0.721	0.731	0.773	0.768	0.73	0.722	0.749	0.727	0.76	0.748	0.723	0.881	0.828	0.828	0.825	0.827	0.818

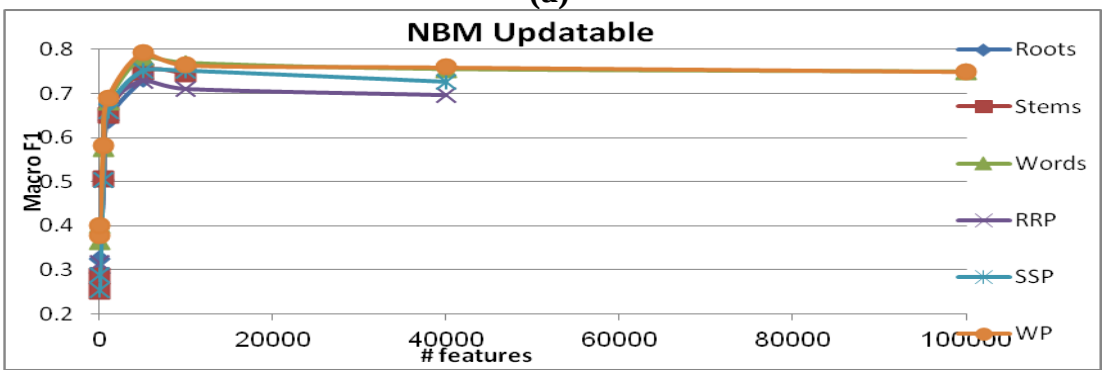
	Filtered Classifier, F1 values for 1000 features						Logit Boost, F1 values for 1000 features						Ordinal Class Classifier, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.837	0.816	0.835	0.837	0.766	0.835	0.947	0.808	0.727	0.939	0.789	0.698	0.688	0.591	0.446	0.753	0.553	0.463
Religious	0.889	0.821	0.735	0.888	0.81	0.747	0.907	0.838	0.769	0.874	0.815	0.81	0.465	0.438	0.492	0.571	0.492	0.468
Arts	0.829	0.806	0.825	0.824	0.773	0.854	0.83	0.746	0.738	0.801	0.753	0.774	0.577	0.574	0.552	0.581	0.62	0.556
Social	0.84	0.726	0.72	0.767	0.788	0.709	0.75	0.677	0.529	0.727	0.558	0.59	0.434	0.354	0.374	0.454	0.424	0.378
Economics	0.881	0.814	0.739	0.87	0.788	0.797	0.923	0.819	0.713	0.939	0.725	0.778	0.438	0.555	0.389	0.767	0.615	0.377
Sports	0.929	0.756	0.765	0.922	0.813	0.752	0.962	0.896	0.868	0.934	0.899	0.851	0.62	0.448	0.535	0.549	0.49	0.596
Music	0.948	0.865	0.756	0.948	0.869	0.826	0.931	0.917	0.863	0.938	0.941	0.878	0.694	0.805	0.692	0.795	0.814	0.684
Educational	0.856	0.791	0.869	0.824	0.806	0.885	0.864	0.82	0.835	0.857	0.82	0.849	0.557	0.585	0.58	0.669	0.61	0.567

	Random Committee, F1 values for 1000 features						Random Sub Space, F1 values for 1000 features						LMT, F1 values for 1000 features					
	Roots	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**	**Roots**	**Stems**	**Words**	**RRP**	**SSP**	**WP**
Politics	0.711	0.688	0.606	0.732	0.588	0.652	0.921	0.859	0.732	0.927	0.842	0.685	0.899	0.35	0.68	0.816	0.766	0.703
Religious	0.658	0.595	0.637	0.65	0.608	0.675	0.846	0.797	0.671	0.88	0.797	0.658	0.863	0.345	0.563	0.835	0.675	0.573
Arts	0.746	0.772	0.72	0.794	0.771	0.776	0.798	0.784	0.763	0.821	0.779	0.741	0.797	0.505	0.693	0.752	0.749	0.696
Social	0.598	0.5	0.573	0.608	0.55	0.561	0.669	0.605	0.61	0.702	0.629	0.632	0.685	0.402	0.487	0.581	0.552	0.474
Economics	0.73	0.667	0.571	0.752	0.653	0.657	0.887	0.76	0.699	0.922	0.827	0.585	0.895	0.35	0.662	0.849	0.713	0.61
Sports	0.894	0.929	0.901	0.91	0.859	0.872	0.942	0.892	0.8	0.934	0.859	0.797	0.955	0.612	0.813	0.894	0.956	0.824
Music	0.98	0.949	0.817	0.979	0.96	0.896	0.928	0.913	0.845	0.96	0.904	0.869	0.929	0.74	0.887	0.923	0.96	0.883
Educational	0.776	0.762	0.767	0.736	0.731	0.789	0.819	0.785	0.832	0.818	0.814	0.791	0.801	0.55	0.806	0.814	0.82	0.796

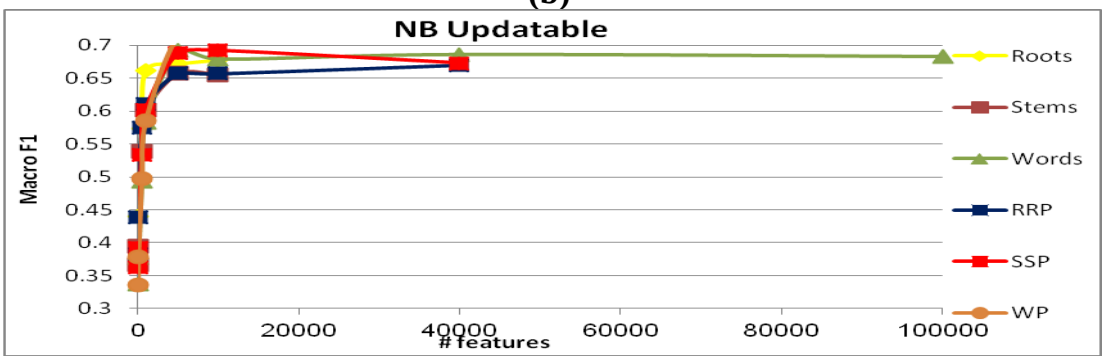
Performance of different VSM representations as number of selected features varied using (a) NB, (b) NBMU, (c) NBU, (d) Logistic, (e) RBF, (f) MLP, (g) NNge, (h) OneR, (i) Random Tree, (j) J48 graft, (k) LAD Tree, (l) BF Tree, (m) NB Tree, (n) VFI, (o) Attribute Selected Classifier, (p) Dagging, (q) Decorate, (r) Multi Class Classifier, (s) CBND, (t) DNBND, (u) ND, (v) Ordinal Class Classifier, (w) Random Committee



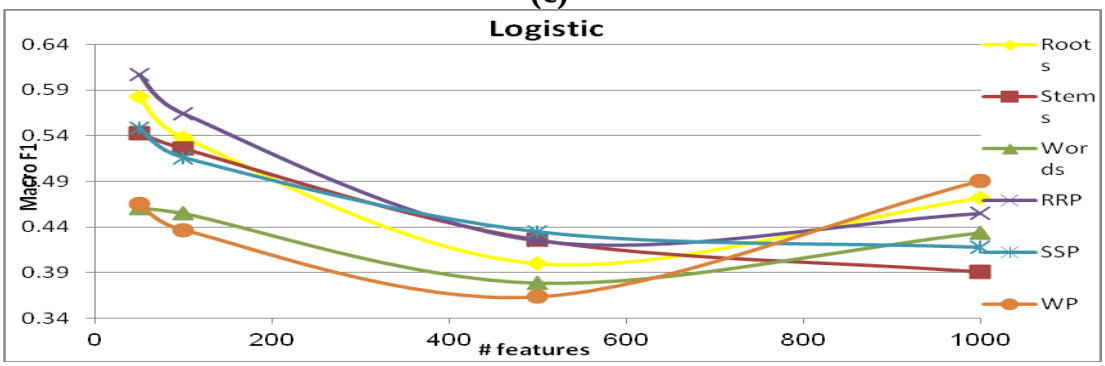
(a)



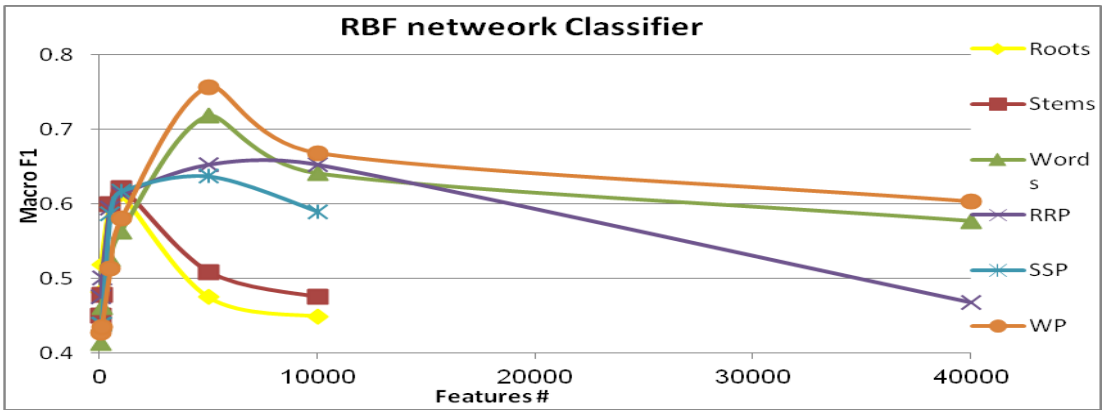
(b)



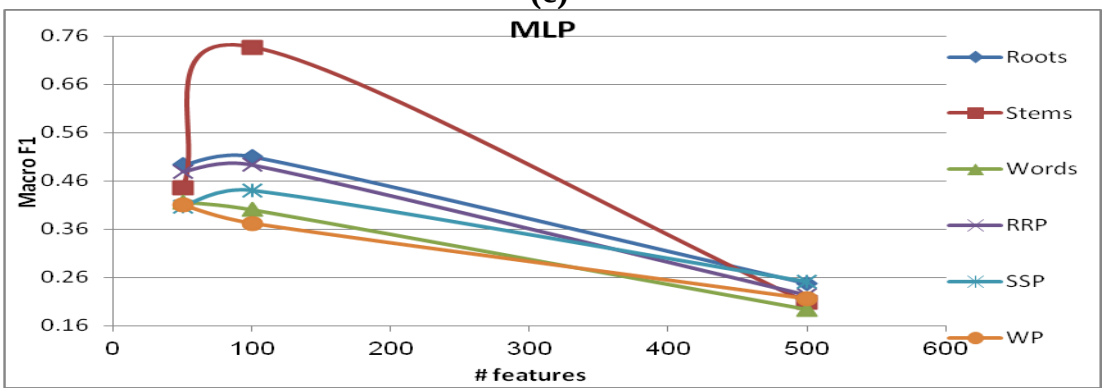
(c)



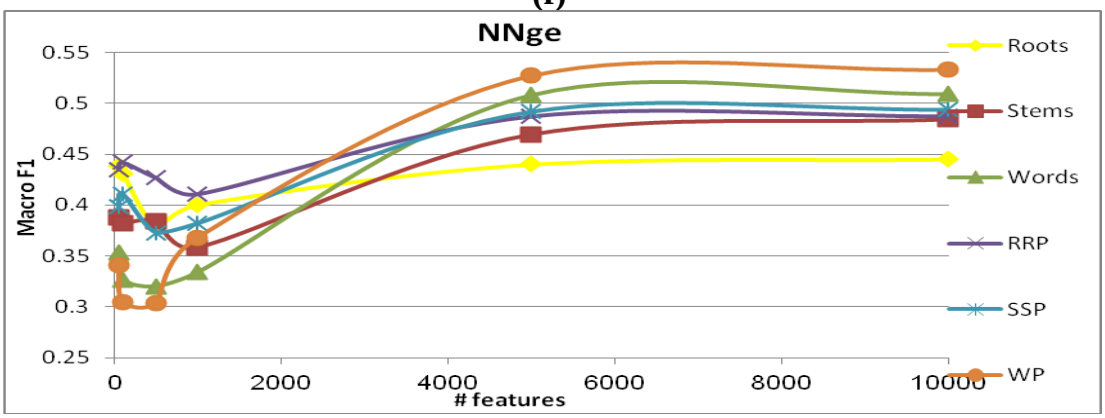
(d)


RBF netweork Classifier
Macro F1
0.8
0.7
0.6
0.5
0.4
0
10000
20000
30000
40000
Features #
Roots
Stems
Words
RRP
SSP
WP

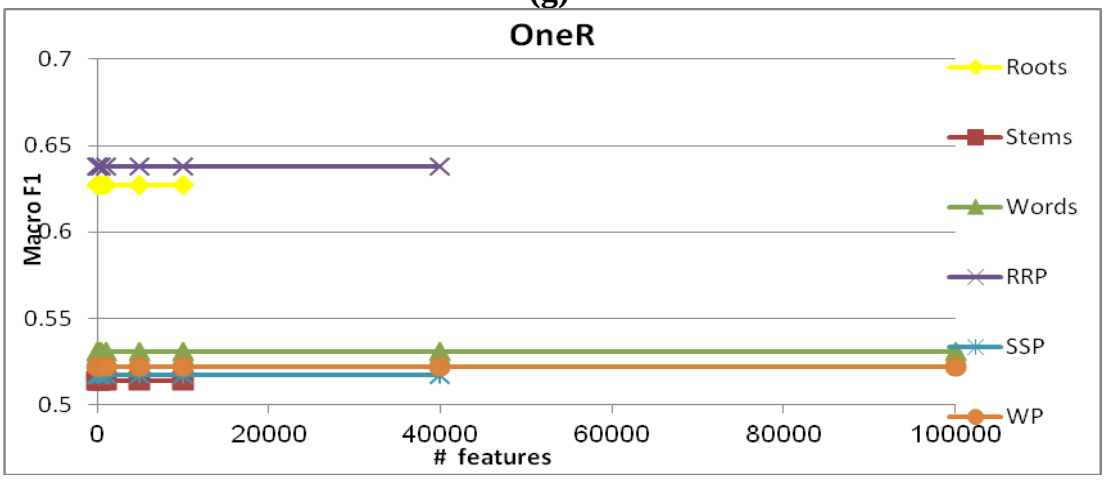
(e)


MLP
Macro F1
0.76
0.66
0.56
0.46
0.36
0.26
0.16
0
100
200
300
400
500
600
features
Roots
Stems
Words
RRP
SSP
WP

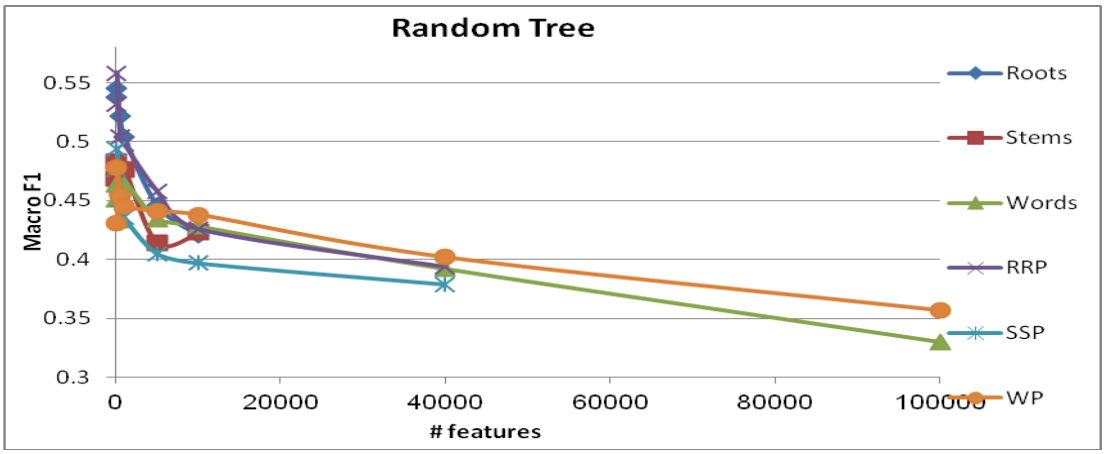
(f)


NNge
Macro F1
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0
2000
4000
6000
8000
10000
features
Roots
Stems
Words
RRP
SSP
WP

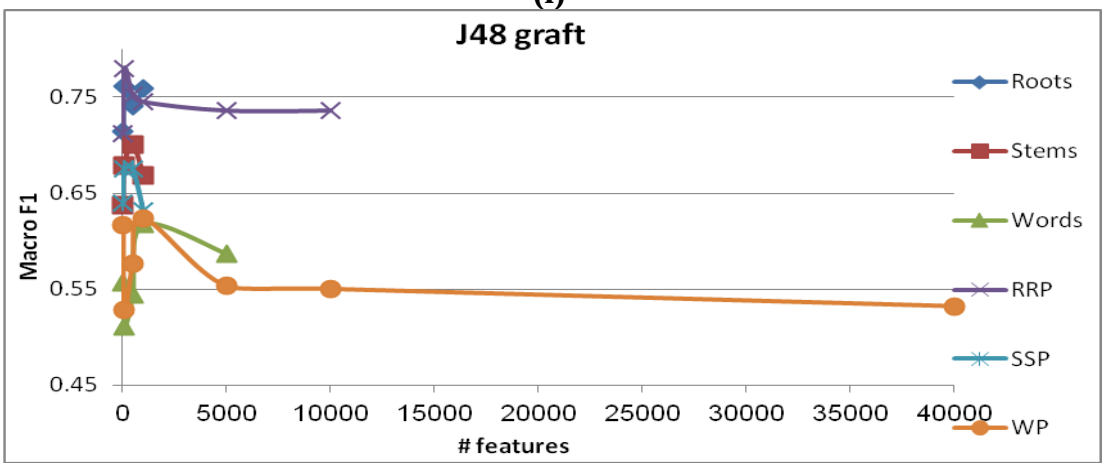
(g)


OneR
Macro F1
0.7
0.65
0.6
0.55
0.5
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

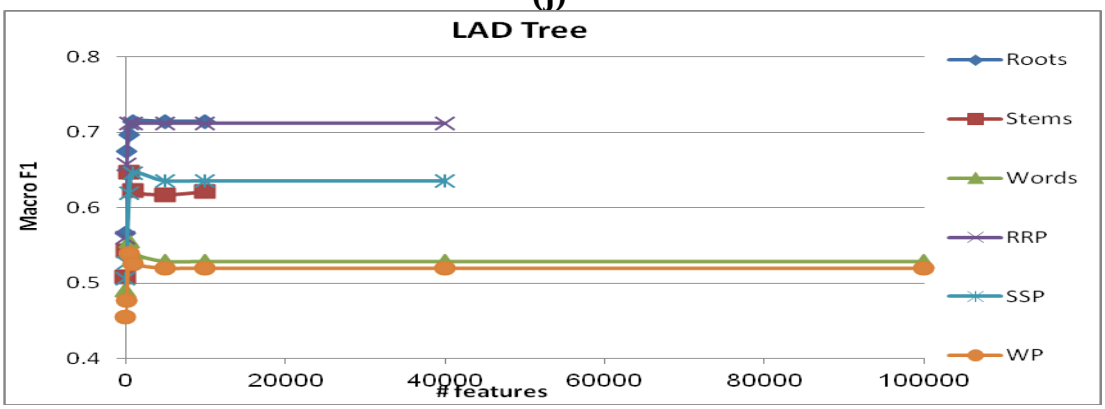
(h)


Random Tree
Macro F1
0.55
0.5
0.45
0.4
0.35
0.3
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

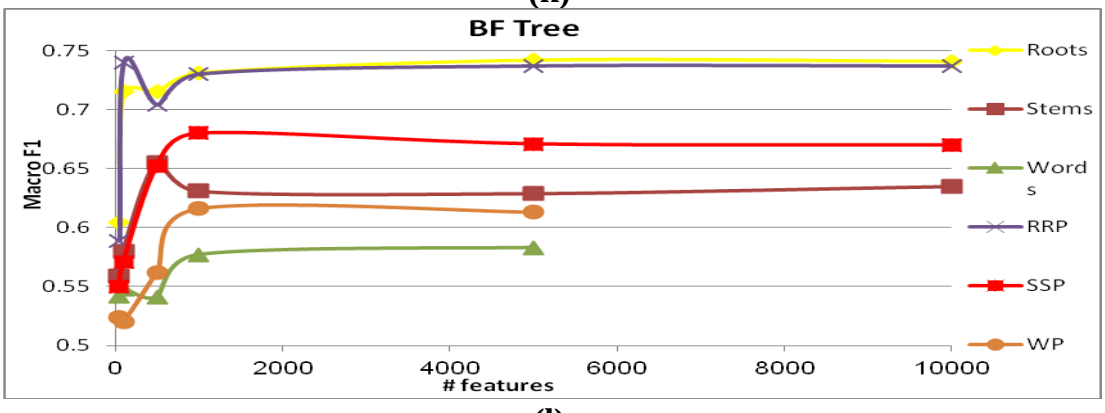
(i)


J48 graft
Macro F1
0.75
0.65
0.55
0.45
0
5000
10000
15000
20000
25000
30000
35000
40000
features
Roots
Stems
Words
RRP
SSP
WP

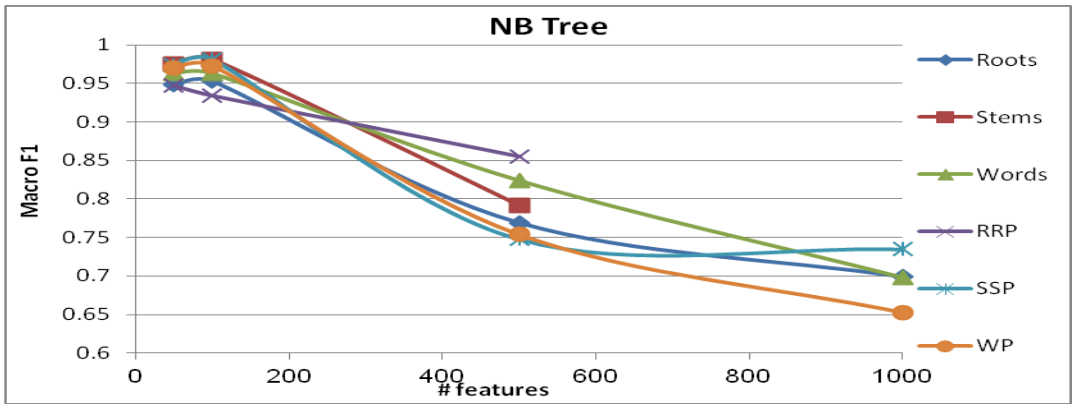
(j)


LAD Tree
Macro F1
0.8
0.7
0.6
0.5
0.4
0
20000
40000
60000
80000
100000
features
Roots
Stems
Words
RRP
SSP
WP

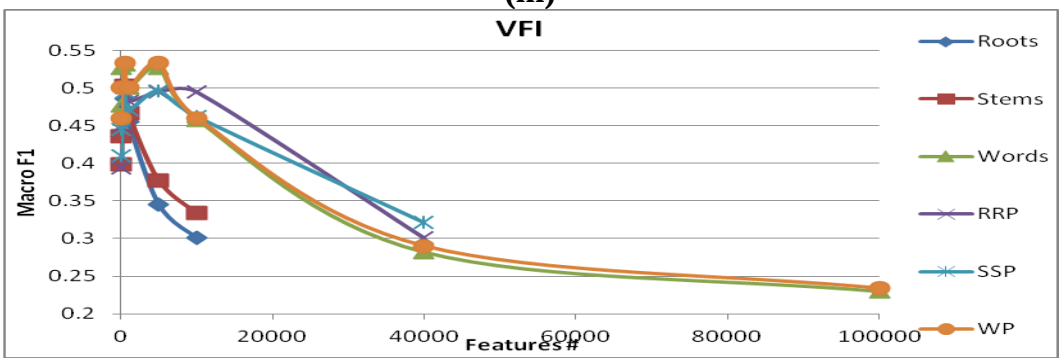
(k)


BF Tree
Macro F1
0.75
0.7
0.65
0.6
0.55
0.5
0
2000
4000
6000
8000
10000
features
Roots
Stems
Words
RRP
SSP
WP

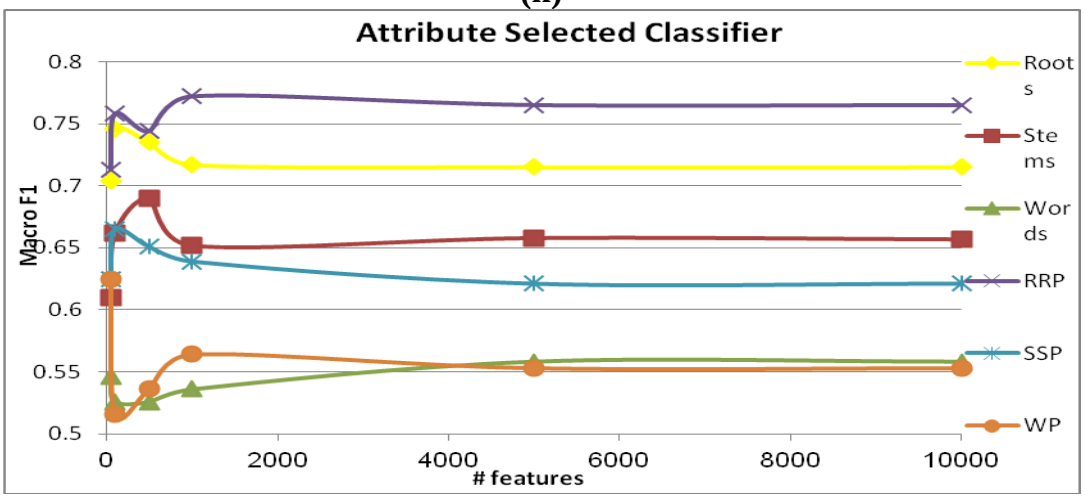
(l)


NB Tree
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

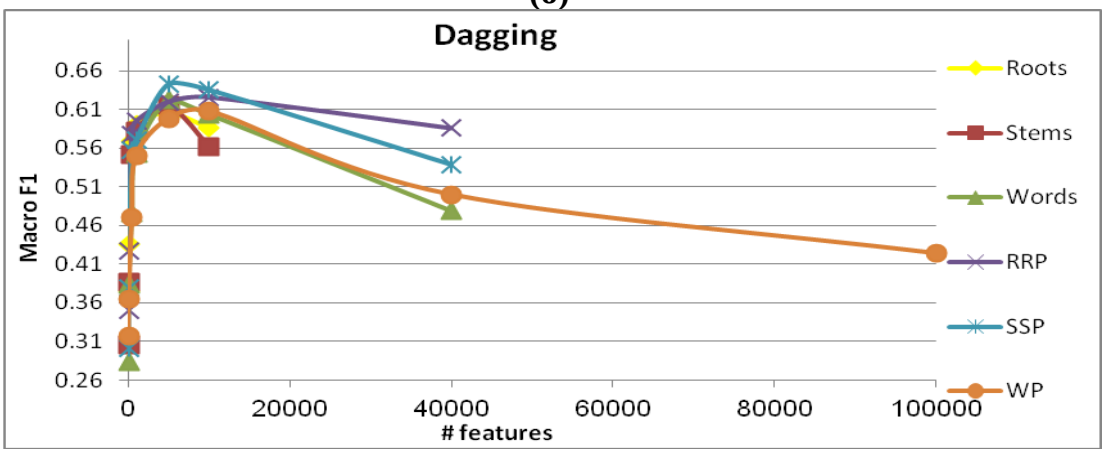
(m)


VFI
Macro F1
Features#
Roots
Stems
Words
RRP
SSP
WP

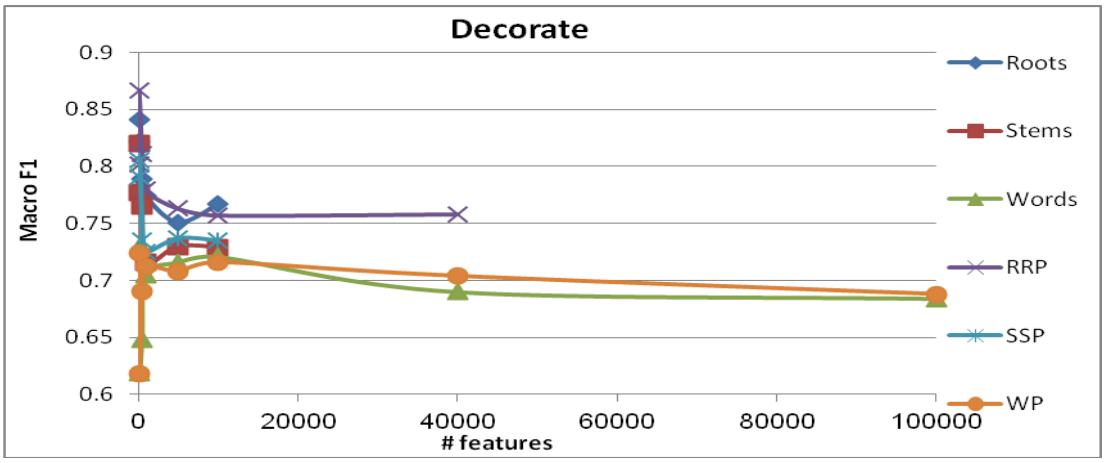
(n)


Attribute Selected Classifier
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

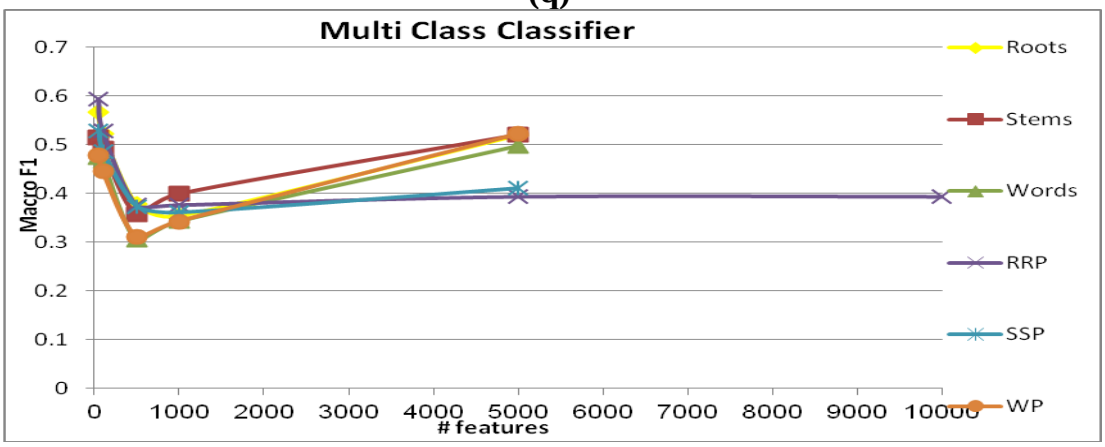
(o)


Dagging
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

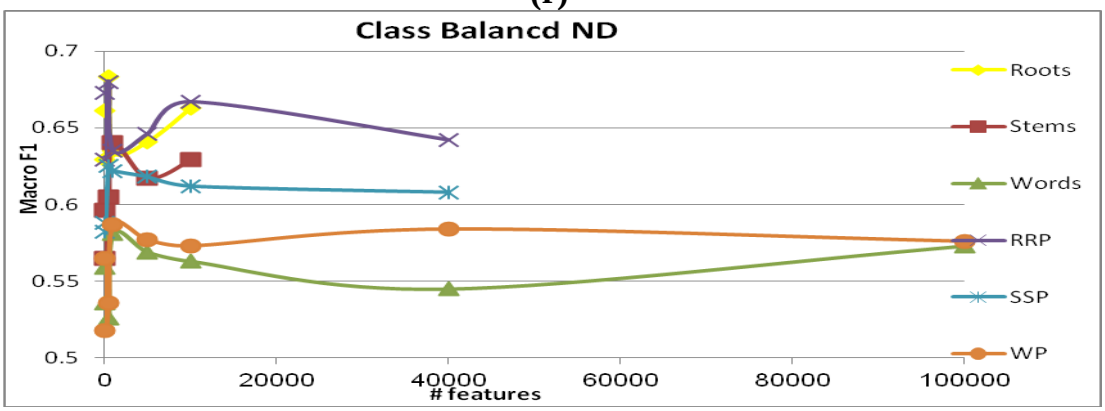
(p)


Decorate
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

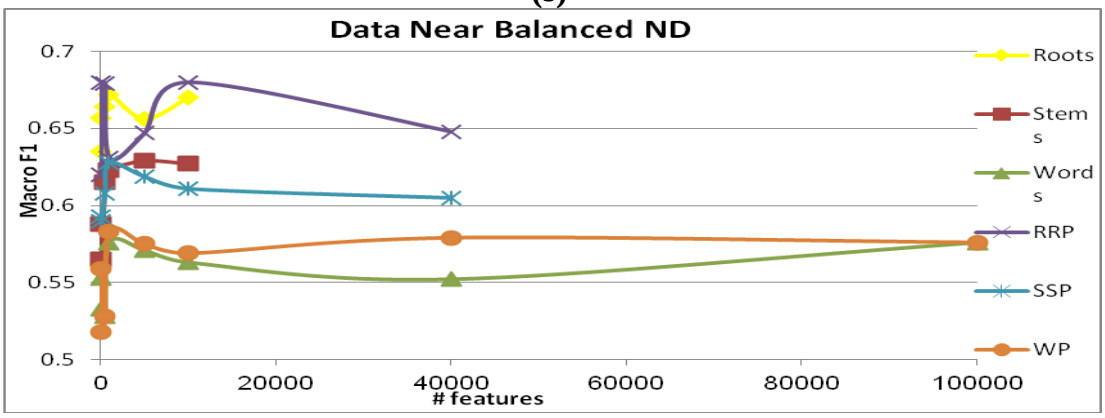
(q)


Multi Class Classifier
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

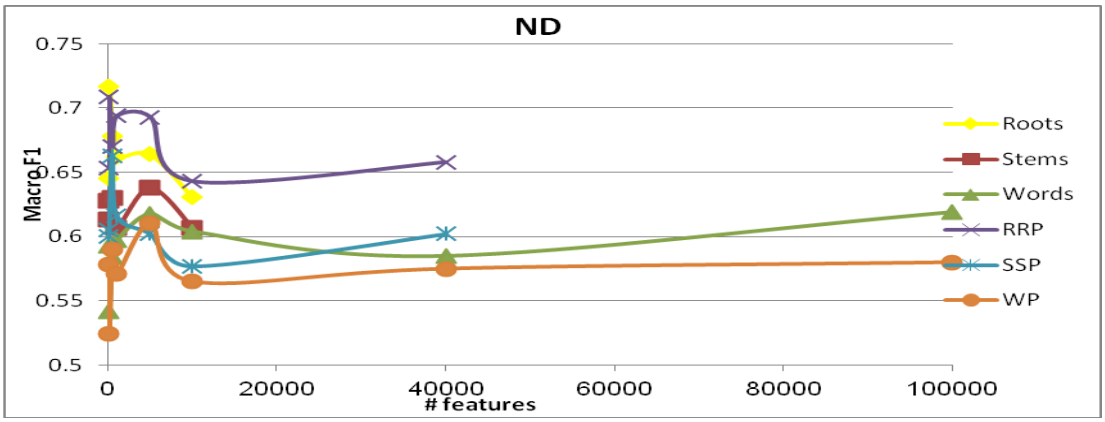
(r)


Class Balancd ND
Macro F1
features
Roots
Stems
Words
RRP
SSP
WP

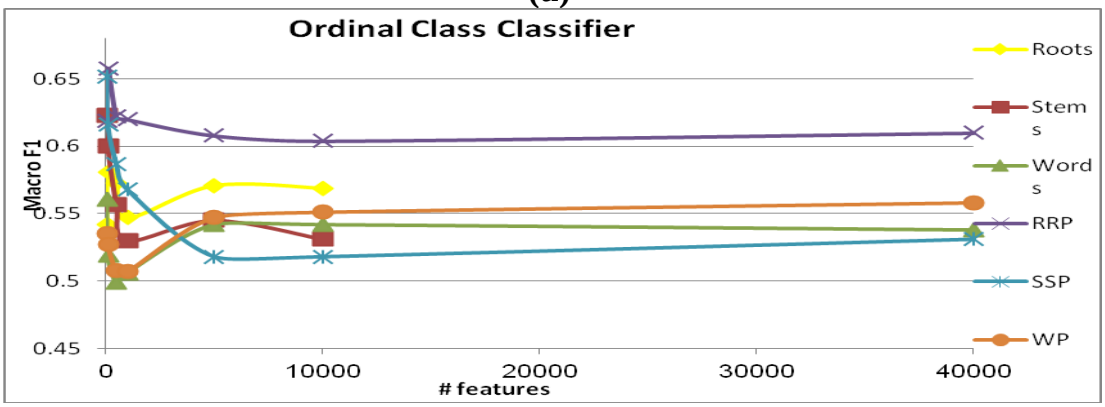
(s)


Data Near Balanced ND
Macro F1
features
Roots
Stem
s
Word
s
RRP
SSP
WP

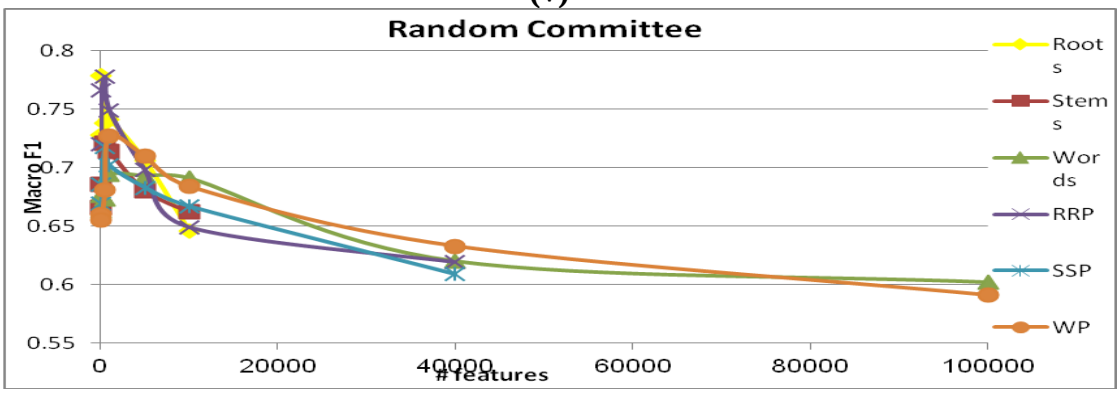
(t)



(u)



(v)



(w)

Maximum $F1^M$ values at specific features number for implemented VSM representations along each classifier.

Classifier		VSM representation					
		Roots	Stems	Words	RRP	SSP	WP
NB	# features	10000	10000	5000	40000	10000	10000
	$F1^M$	0.676	0.656	0.695	0.674	0.691	0.721
NBMU	# features	5000	5000	5000	5000	5000	5000
	$F1^M$	0.734	0.75	0.781	0.729	0.752	0.793
NBU	# features	10000	5000	5000	40000	10000	10000
	$F1^M$	0.678	0.657	0.693	0.67	0.692	0.722
Logistic	# features	50	50	50	50	50	1000
	$F1^M$	0.583	0.543	0.461	0.607	0.548	0.491
RBF	# features	1000	1000	5000	5000	5000	5000
	$F1^M$	0.613	0.621	0.718	0.653	0.637	0.757
MLP	# features	100	100	50	100	100	50
	$F1^M$	0.511	0.737	0.417	0.493	0.44	0.41
NNge	# features	10000	10000	10000	5000	10000	10000
	$F1^M$	0.445	0.484	0.509	0.487	0.494	0.533
OneR	# features	50	50	50	50	50	50
	$F1^M$	0.627	0.514	0.531	0.638	0.517	0.522
Random Tree	# features	50	100	100	100	100	100

	$F1^M$	0.545	0.483	0.464	0.558	0.494	0.478
J48 graft	# features	100	500	1000	100	100	1000
	$F1^M$	0.761	0.701	0.618	0.78	0.675	0.624
LAD Tree	# features	1000	500	500	1000	1000	500
	$F1^M$	0.714	0.647	0.556	0.712	0.646	0.539
BF Tree	# features	5000	500	5000	5000	1000	1000
	$F1^M$	0.742	0.655	0.583	0.737	0.68	0.616
NB Tree	# features	100	100	50	50	100	100
	$F1^M$	0.953	0.981	0.963	0.947	0.981	0.973
VFI	# features	500	500	500	500	5000	500
	$F1^M$	0.486	0.503	0.531	0.501	0.496	0.534
Atribute Selected Classifier	# features	100	500	5000	1000	100	50
	$F1^M$	0.746	0.69	0.558	0.772	0.665	0.625
Dagging	# features	5000	5000	5000	10000	5000	10000
	$F1^M$	0.606	0.615	0.622	0.626	0.643	0.608
Decorate	# features	100	50	50	100	100	10000
	$F1^M$	0.841	0.82	0.73	0.867	0.805	0.716
Multi Class Classifier	# features	50	5000	5000	50	50	5000
	$F1^M$	0.567	0.521	0.498	0.595	0.529	0.522
CBND	# features	500	1000	1000	500	500	1000
	$F1^M$	0.683	0.64	0.581	0.68	0.625	0.587
DNBND	# features	1000	5000	1000	100	1000	1000
	$F1^M$	0.672	0.629	0.576	0.68	0.628	0.583
ND	# features	100	5000	5000	100	500	5000
	$F1^M$	0.717	0.638	0.618	0.709	0.663	0.61
Ordinal Class Classifier	# features	100	50	50	100	50	40000
	$F1^M$	0.581	0.623	0.561	0.658	0.652	0.558
Random Committee	# features	100	500	1000	500	500	1000
	$F1^M$	0.779	0.721	0.694	0.778	0.718	0.727

$F1^M$ Improvement/Degradation by comparing implemented VSM representations performances at feature numbers presented above for each classifier.

Classifier	Max w-$F1^M$, VSM type	Improvement/degradation of first compared to second VSM type			Improvement/degradation of second compared to first VSM type			
		Roots, RRP (%)	Stems, SSP (%)	Words, WP (%)	Roots, Stems (%)	Roots, Words (%)	RRP, SSP (%)	RRP, WP (%)
NB	0.721, WP	-0.2	+3.5	+2.6	+2	-1.9	-1.7	-4.7
NBMU	0.793, WP	-0.5	+0.2	+1.2	-1.6	-3.7	-2.3	-6.4
NBU	0.722, WP	-0.8	+3.5	+2.9	+2.1	-1.5	-3.5	-6.2
Logistic	0.607, RRP	+2.4	+0.5	+3	+4	+12.2	+5.9	+11.6
RBF	0.757, WP	+4	+1.5	+3.5	-0.8	-10.5	+1.6	-10.4
MLP	0.737, Stems	-1.8	-29.7	-0.6	-22.6	+9.4	+5.3	+8.3
NNge	0.533, WP	+4.2	+1	+2.4	+3.9	-6.4	+0.7	-4.6
OneR	0.638, RRP	+1.1	+0.3	-0.9	+11.3	+9.6	+12.1	+11.6
Random Tree	0.558, RRP	+1.3	+1.1	+1.4	+6.2	+8.1	+6.4	+8.0
J48 graft	0.78, RRP	+1.9	-2.6	+0.6	+6	+14.3	+10.5	+15.6
LAD Tree	0.714, Roots	-0.2	-0.1	-1.7	+6.7	+15.8	+6.6	+17.3
BF Tree	0.742, Roots	-0.5	+2.5	+3.3	+8.7	+15.9	+5.7	+12.1
NB Tree	0.981, Stems	-0.6	0	+1.0	+2.8	-1	+3.4	-2.6
VFI	0.534, WP	+1.5	-0.7	+0.3	+1.7	-4.5	+0.5	-3.3
Attrib.Sel.Cla	0.772, RRP	+2.6	-2.5	+6.7	+5.6	+18.8	+10.7	+14.7
Dagging	0.643, SSP	+2	+2.8	-1.4	+0.9	-1.6	+1.7	+1.8
Decorate	0.867, RRP	+2.6	-1.5	-1.4	+2.1	+11.1	+6.6	+15.1
MuliClassClas	0.595, RRP	+2.8	+1.4	+2.4	+4.6	+6.9	+6.6	+7.3
CBND	0.683, Roots	-0.3	-1.5	+0.6	+7.8	+10.2	+5.5	+9.3
DNBND	0.68, RRP	+0.8	-0.1	+0.7	+4.3	+9.6	+5.2	+9.7
ND	0.717, Roots	-0.8	+2.5	-0.8	+7.9	+9.9	+4.6	+9.9
Ord.ClaaClass	0.658, RRP	+7.7	+2.9	-0.3	-4.2	+2	-0.6	+10.0
Rand.Comm.	0.779, Roots	-0.1	-0.3	+3.3	+5.8	+8.5	+8	+5.1

Performance of implemented classifiers along different representations by selecting best 5000 features with significance testing.

Classifiers	Roots	RRP	Stems	SSP	Words	WP	v/ /*
Bayes based, significance relative to NBM							
BN	0.99 (0.03) v	0.98 (0.03) v	0.99 (0.03 v)	0.99 (0.03) v	0.97 (0.05) v	0.98 (0.05) v	6/0/0

NB	**0.71 (0.15)**	**0.68 (0.12) ***	**0.64 (0.13)**	**0.66 (0.12)**	**0.64 (0.16)**	**0.67 (0.12)**	**0/5/1**
NBM	**0.74 (0.11)**	**0.76 (0.10)**	**0.71 (0.10)**	**0.73 (0.10)**	**0.72 (0.11)**	**0.74 (0.11)**	
Complement NB	**0.71 (0.10)**	**0.74 (0.11)**	**0.73 (0.10)**	**0.74 (0.13)**	**0.74 (0.12)**	**0.76 (0.11)**	**0/6/0**
NBMU	**0.73 (0.13)**	**0.72 (0.12)**	**0.72 (0.11)**	**0.68 (0.15)**	**0.72 (0.13)**	**0.73 (0.11)**	**0/6/0**
NBU	**0.71 (0.15)**	**0.68 (0.12) ***	**0.64 (0.13)**	**0.66 (0.12)**	**0.64 (0.16)**	**0.67 (0.12)**	**0/5/1**
Functions, significance relative to SMO							
SMO	**0.64 (0.12)**	**0.71 (0.12)**	**0.67 (0.15)**	**0.67 (0.14)**	**0.69 (0.14)**	**0.69 (0.15)**	
Simple Logistic	**0.85 (0.09) v**	**0.85 (0.10) v**	**0.79 (0.12)**	**0.77 (0.12) v**	**0.68 (0.13)**	**0.73 (0.12)**	**3/3/0**
RBF	**0.37 (0.16) ***	**0.63 (0.14)**	**0.48 (0.14) ***	**0.59 (0.15)**	**0.68 (0.14)**	**0.73 (0.11)**	**0/4/2**
Rules, significance relative to PART							
JRip	**0.86 (0.11)**	**0.89 (0.10)**	**0.73 (0.12) v**	**0.72 (0.13)**	**0.66 (0.15)**	**0.66 (0.13)**	**1/5/0**
PART	**0.82 (0.09)**	**0.85 (0.10)**	**0.60 (0.12)**	**0.64 (0.11)**	**0.56 (0.12)**	**0.58 (0.17)**	
Ridor	**0.85 (0.09)**	**0.85 (0.09)**	**0.68 (0.11)**	**0.68 (0.18)**	**0.60 (0.15)**	**0.56 (0.17)**	**0/6/0**
OneR	**0.79 (0.15)**	**0.80 (0.14)**	**0.51 (0.17)**	**0.52 (0.17)**	**0.54 (0.20)**	**0.53 (0.18)**	**0/6/0**
NNge	**0.45 (0.16) ***	**0.48 (0.19) ***	**0.46 (0.20)**	**0.45 (0.20) ***	**0.52 (0.20)**	**0.55 (0.16)**	**0/3/3**
Decision Table	**0.49 (0.16) ***	**0.51 (0.15) ***	**0.47 (0.15)**	**0.46 (0.15) ***	**0.45 (0.13) ***	**0.44 (0.15)**	**0/2/4**
Trees, significance relative to J48							
J48	**0.87 (0.09)**	**0.88 (0.08)**	**0.69 (0.11)**	**0.69 (0.16)**	**0.57 (0.14)**	**0.57 (0.13)**	
Random Forest	**0.62 (0.16) ***	**0.62 (0.12) ***	**0.60 (0.13)**	**0.60 (0.13)**	**0.60 (0.15)**	**0.62 (0.16)**	**0/4/2**
RepTree	**0.87 (0.09)**	**0.86 (0.10)**	**0.64 (0.11)**	**0.68 (0.11)**	**0.52 (0.15)**	**0.51 (0.16)**	**0/6/0**
BF Tree	**0.89 (0.07)**	**0.88 (0.08)**	**0.68 (0.12)**	**0.70 (0.11)**	**0.52 (0.17)**	**0.51 (0.13)**	**0/6/0**
FT	**0.78 (0.11)**	**0.81 (0.12)**	**0.71 (0.14)**	**0.73 (0.12)**	**0.67 (0.12)**	**0.67 (0.12)**	**0/6/0**
LAD Tree	**0.84 (0.10)**	**0.84 (0.10)**	**0.57 (0.16)**	**0.60 (0.13)**	**0.54 (0.15)**	**0.52 (0.15)**	**0/6/0**
Random Tree	**0.35 (0.17) ***	**0.34 (0.19) ***	**0.33 (0.15) ***	**0.30 (0.18) ***	**0.38 (0.14) ***	**0.37 (0.14) ***	**0/0/6**
Simple Cart	**0.86 (0.08)**	**0.85 (0.10)**	**0.70 (0.10)**	**0.68 (0.10)**	**0.56 (0.18)**	**0.53 (0.15)**	**0/6/0**
Miscillanueousm significance relative to VFI							
VFI	**0.36 (0.18)**	**0.59 (0.14)**	**0.36 (0.17)**	**0.53 (0.15)**	**0.59 (0.18)**	**0.54 (0.17)**	
Hyper Pipes	**0.46 (0.19)**	**0.56 (0.19)**	**0.57 (0.18) v**	**0.64 (0.19)**	**0.65 (0.19)**	**0.74 (0.15) v**	**2/4/0**
Meta, significance relative to AdaBoost.M1							
AdaBoost.M1	**0.90 (0.07)**	**0.92 (0.07)**	**0.79 (0.10)**	**0.81 (0.12)**	**0.72 (0.13)**	**0.70 (0.14)**	
Attr Sel Classifier	**0.84 (0.09)**	**0.86 (0.10)**	**0.66 (0.14) ***	**0.67 (0.15) ***	**0.50 (0.16) ***	**0.52 (0.14) ***	**0/2/4**
Bagging	**0.91 (0.06)**	**0.91 (0.07)**	**0.75 (0.09)**	**0.75 (0.10)**	**0.68 (0.12)**	**0.68 (0.17)**	**0/6/0**
CVR	**0.90 (0.09)**	**0.88 (0.08)**	**0.76 (0.12)**	**0.76 (0.11**	**0.64 (0.16)**	**0.61 (0.13)**	**0/6/0**
Dagging	**0.44 (0.24) ***	**0.58 (0.16) ***	**0.56 (0.17) ***	**0.62 (0.17) ***	**0.58 (0.17) ***	**0.56 (0.18)**	**0/1/5**
END	**0.92 (0.08)**	**0.92 (0.06)**	**0.83 (0.09)**	**0.84 (0.10)**	**0.73 (0.14)**	**0.73 (0.14)**	**0/6/0**
Filtered Classifier	**0.83 (0.09) ***	**0.82 (0.08) ***	**0.75 (0.13)**	**0.76 (0.13)**	**0.76 (0.11)**	**0.74 (0.16)**	**0/4/2**
Logit Boost	**0.94 (0.05)**	**0.94 (0.06)**	**0.82 (0.12)**	**0.81 (0.11)**	**0.70 (0.12)**	**0.70 (0.15)**	**0/6/0**
CBND	**0.73 (0.16) ***	**0.72 (0.16) ***	**0.62 (0.16) ***	**0.58 (0.14) ***	**0.45 (0.19) ***	**0.50 (0.16) ***	**0/0/6**
DNBND	**0.73 (0.15) ***	**0.74 (0.17) ***	**0.62 (0.17) ***	**0.58 (0.13) ***	**0.47 (0.18) ***	**0.50 (0.15) ***	**0/0/6**
ND	**0.69 (0.14) ***	**0.71 (0.17) ***	**0.65 (0.15) ***	**0.58 (0.18) ***	**0.51 (0.16) ***	**0.51 (0.16) ***	**0/0/6**
Ordinal Class Classifier	**0.72 (0.14) ***	**0.71 (0.12) ***	**0.56 (0.19) ***	**0.51 (0.19) ***	**0.50 (0.19) ***	**0.53 (0.18) ***	**0/0/6**
Random Committee	**0.64 (0.15) ***	**0.62 (0.13) ***	**0.57 (0.15) ***	**0.58 (0.16) ***	**0.65 (0.12)**	**0.66 (0.13)**	**0/2/4**
RSS	**0.93 (0.05)**	**0.93 (0.06)**	**0.83 (0.09)**	**0.85 (0.08)**	**0.69 (0.16)**	**0.69 (0.15)**	**0/6/0**
Rotation Forest	**0.91 (0.06)**	**0.90 (0.08)**	**0.73 (0.11)**	**0.78 (0.11)**	**0.69 (0.13)**	**0.70 (0.15)**	**0/6/0**

Numbers in brackets are for standard deviation, win/tie/loose is abbr. by v/ /*

Significance tests to compare results of some classifiers on Roots and RRP representations

For 1000 features:

Tester: weka.experiment.PairedCorrectedTTester
Analysing: F_measure
Datasets: 7
Resultsets: 2
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 20/06/12 04:17 م
base test: roots
Dataset (1) 'root2-classe | (2) 'RootsAndR
---
rules.PART '-M 2 -C 0.25 (50) 0.83(0.10) | 0.84(0.11)
rules.JRip '-F 3 -N 2.0 - (50) 0.87(0.09) | 0.85(0.11)
trees.J48 '-C 0.25 -M 2' (50) 0.86(0.08) | 0.87(0.09)
trees.REPTree '-M 2 -V 0. (50) 0.87(0.09) | 0.86(0.10)
functions.SMO '-C 1.0 -L (50) 0.68(0.13) | 0.68(0.11)
bayes.NaiveBayesMultinomi (50) 0.78(0.11) | 0.78(0.11)
bayes.BayesNet '-D -Q wek (50) 0.99(0.03) | 0.98(0.03)

(v/ /*) | (0/7/0)

Key:
(1) 'root2-classes - 804 - 11063-supervised.attribute.AttributeSelection-EChiSquaredAttributeEval-SRanker -T -1.7976931348623157E308 -N -1-unsupervised.attribute.Remove-R10001-11063-unsupervised.attribute.Remove-R1001-10000'
(2) 'RootsAndRootPhrases-supervised.attribute.AttributeSelection-EChiSquaredAttributeEval-SRanker -T -1.7976931348623157E308 -N -1-unsupervised.attribute.Remove-R40001-50091-unsupervised.attribute.Remove-R5001-40000-unsupervised.attribute.Remove-R1001-5000'

Significance tests to compare results of few classifiers on Stems and SSP representations

For 1000 features:

Tester: weka.experiment.PairedCorrectedTTester
Analysing: F_measure
Datasets: 7
Resultsets: 2
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 20/06/12 03:57 م
Base test: Stems

Dataset	(1) 'Stems-superv	(2) 'StemsAndS
bayes.BayesNet '-D -Q wek (50)	0.99(0.03)	0.99(0.03)
bayes.NaiveBayesMultinomi (50)	0.72(0.11)	0.72(0.10)
functions.SMO '-C 1.0 -L (50)	0.64(0.12)	0.64(0.11)
rules.JRip '-F 3 -N 2.0 - (50)	0.73(0.14)	0.74(0.13)
rules.PART '-M 2 -C 0.25 (50)	0.68(0.13)	0.70(0.12)
trees.J48 '-C 0.25 -M 2' (50)	0.70(0.13)	0.69(0.11)
trees.REPTree '-M 2 -V 0. (50)	0.64(0.11)	0.67(0.10)
(v/ /*)	(0/7/0)	

Key:
(1) 'Stems-supervised.attribute.AttributeSelection-EChiSquaredAttributeEval-SRanker -T -1.7976931348623157E308 -N -1-unsupervised.attribute.Remove-R10001-14945,14947-18019-unsupervised.attribute.Remove-R10001-unsupervised.attribute.Remove-R1001-10000'
(2) 'StemsAndStemPhrases-supervised.attribute.AttributeSelection-EChiSquaredAttributeEval-SRanker -T -1.7976931348623157E308 -N -1-unsupervised.attribute.Remove-R40001-40016,42600-43255-unsupervised.attribute.Remove-R40001-42583-unsupervised.attribute.Remove-R5001-40000-unsupervised.attribute.Remove-R1001-5000'

Significance Testing among Best Performing Classifiers for 1000 features relative to LMT then BN classifiers

Significance Testing-1000-best classifiers from different types (rel to LMT):

Tester: weka.experiment.PairedCorrectedTTester
Analysing: F_measure
Datasets: 6
Resultsets: 17
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 28/04/13 03:14 م

Dataset	(8) trees.LMT '-I	(1) bayes.Baye	(2) bayes.Naiv	(3) functions.	(4) functions.	(5) rules.JRip	(6) rules.Rido	(7) rules.PART	(9) trees.J48	(10) trees.FT	(11) trees.BFT	(12) trees.REP	(13) meta.AdaB	(14) meta.Logi	(15) meta.END	(16) meta.Rand	(17) meta.Filt
'root2-classes - 804 - 11 (50)	0.83(0.10)	0.99(0.03) v	0.78(0.11)	0.83(0.09)	0.68(0.13) *	0.87(0.09)	0.86(0.08)	0.83(0.10)	0.86(0.08)	0.81(0.10)	0.88(0.07)	0.87(0.09)	0.92(0.07) v	0.94(0.06) v	0.92(0.07) v	0.93(0.07) v	0.83(0.09)
'RootsAndRootPhrases-weka (50)	0.83(0.10)	0.98(0.03) v	0.78(0.11)	0.83(0.11)	0.68(0.11) *	0.85(0.11)	0.85(0.09)	0.84(0.11)	0.87(0.09)	0.81(0.08)	0.88(0.08)	0.86(0.10)	0.93(0.05) v	0.95(0.05) v	0.92(0.06) v	0.93(0.07) v	0.84(0.08)
'Stems-weka.filters.super (50)	0.79(0.13)	0.99(0.03) v	0.72(0.11)	0.79(0.13)	0.64(0.12) *	0.73(0.14)	0.69(0.11)	0.68(0.13)	0.70(0.13)	0.73(0.13)	0.69(0.09)	0.64(0.11) *	0.84(0.09)	0.80(0.12)	0.85(0.08)	0.83(0.08)	0.81(0.10)
'StemsAndStemPhrases-weka (50)	0.78(0.11)	0.99(0.03) v	0.72(0.10)	0.78(0.11)	0.64(0.11) *	0.74(0.13)	0.69(0.13)	0.70(0.12)	0.69(0.11)	0.72(0.11)	0.68(0.12)	0.67(0.10)	0.82(0.11)	0.80(0.10)	0.85(0.09)	0.85(0.09)	0.81(0.11)
'Words-weka.filters.super (50)	0.64(0.15)	0.96(0.06) v	0.67(0.11)	0.63(0.15)	0.59(0.16)	0.69(0.16)	0.58(0.15)	0.49(0.17) *	0.47(0.16) *	0.64(0.16)	0.55(0.15)	0.52(0.18)	0.71(0.12)	0.69(0.12)	0.70(0.16)	0.72(0.14)	0.80(0.12) v
'WordsAndPhrases-weka.fil (50)	0.65(0.14)	0.96(0.06) v	0.68(0.12)	0.63(0.14)	0.56(0.15)	0.67(0.16)	0.57(0.14)	0.50(0.14) *	0.46(0.16) *	0.64(0.13)	0.52(0.15)	0.53(0.16)	0.74(0.11)	0.72(0.12)	0.74(0.12)	0.71(0.14)	0.80(0.13) v
(v/ /*)	(6/0/0)	(0/6/0)	(0/6/0)	(0/2/4)	(0/6/0)	(0/6/0)	(0/4/2)	(0/4/2)	(0/6/0)	(0/5/1)	(2/4/0)	(2/4/0)	(2/4/0)	(2/4/0)	(2/4/0)		

Key:
(1) bayes.BayesNet '-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimate.SimpleEstimator -- -A 0.5' 746037443258775954
(2) bayes.NaiveBayesMultinomial '' 5932177440181257085
(3) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(4) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"' -6585883636378691736
(5) rules.JRip '-F 3 -N 2.0 -O 2 -S 1' -6589312996832147161
(6) rules.Ridor '-F 3 -S 1 -N 2.0' -7261533075088314436
(7) rules.PART '-M 2 -C 0.25 -Q 1' 8121455039782598361
(8) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943
(9) trees.J48 '-C 0.25 -M 2' -217733168393644444
(10) trees.FT '-I 15 -F 0 -M 15 -W 0.0' -1113212459618105000
(11) trees.BFTree '-S 1 -M 2 -N 5 -C 1.0 -P POSTPRUNED' -7035607375962528217
(12) trees.REPTree '-M 2 -V 0.0010 -N 3 -S 1 -L -1' -9216785998198681299
(13) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.J48 -- -C 0.25 -M 2' -7378107808933117974
(14) meta.LogitBoost '-P 100 -F 0 -R 1 -L -1.7976931348623157E308 -H 1.0 -S 1 -I 10 -W trees.DecisionStump' -3905660358715833753
(15) meta.END '-S 1 -I 10 -W meta.nestedDichotomies.ND -- -S 1 -W trees.J48 -- -C 0.25 -M 2' -4143242362912214956
(16) meta.RandomSubSpace '-P 0.5 -S 1 -I 10 -W trees.REPTree -- -M 2 -V 0.0010 -N 3 -S 1 -L -1' 1278172513912424947
(17) meta.FilteredClassifier '-F \"supervised.attribute.Discretize -R first-last\" -W trees.J48 -- -C 0.25 -M 2' -4523450618538717400

Significance Testing-1000-best classifiers from different types (rel to BN):

Tester: weka.experiment.PairedCorrectedTTester
Analysing: F_measure
Datasets: 6
Resultsets: 17

Confidence: 0.05 (two tailed)
Sorted by: -
Date: 28/04/13 03:15 م

Dataset (1) bayes.BayesNe | (2) bayes.Naiv (3) functions. (4) functions. (5) rules.JRip (6) rules.Rido (7) rules.PART (8) trees.LMT (9) trees.J48 (10) trees.FT (11) trees.BFT (12) trees.REP (13) meta.AdaB (14) meta.Logi (15) meta.END (16) meta.Rand (17) meta.Filt

---

'root2-classes - 804 - 11 (50) 0.99(0.03) | 0.78(0.11) * 0.83(0.09) * 0.68(0.13) * 0.87(0.09) * 0.86(0.08) * 0.83(0.10) * 0.83(0.10) * 0.86(0.08) * 0.81(0.10) * 0.88(0.07) * 0.87(0.09) * 0.92(0.07) * 0.94(0.06) 0.92(0.07) * 0.93(0.07) 0.83(0.09) *
'RootsAndRootPhrases-weka (50) 0.98(0.03) | 0.78(0.11) * 0.83(0.11) * 0.68(0.11) * 0.85(0.11) * 0.85(0.09) * 0.84(0.11) * 0.83(0.10) * 0.87(0.09) * 0.81(0.08) * 0.88(0.08) * 0.86(0.10) * 0.93(0.05) * 0.95(0.05) 0.92(0.06) * 0.93(0.07) * 0.84(0.08) *
'Stems-weka.filters.super (50) 0.99(0.03) | 0.72(0.11) * 0.79(0.13) * 0.64(0.12) * 0.73(0.14) * 0.69(0.11) * 0.68(0.13) * 0.79(0.13) * 0.70(0.13) * 0.73(0.13) * 0.69(0.09) * 0.64(0.11) * 0.84(0.09) * 0.80(0.12) * 0.85(0.08) * 0.83(0.08) * 0.81(0.10) *
'StemsAndStemPhrases-weka (50) 0.99(0.03) | 0.72(0.10) * 0.78(0.11) * 0.64(0.11) * 0.74(0.13) * 0.69(0.13) * 0.70(0.12) * 0.78(0.11) * 0.69(0.11) * 0.72(0.11) * 0.68(0.12) * 0.67(0.10) * 0.82(0.11) * 0.80(0.10) * 0.85(0.09) * 0.85(0.09) * 0.81(0.11) *
'Words-weka.filters.super (50) 0.96(0.06) | 0.67(0.11) * 0.63(0.15) * 0.59(0.16) * 0.69(0.16) * 0.58(0.15) * 0.49(0.17) * 0.64(0.15) * 0.47(0.16) * 0.64(0.16) * 0.55(0.15) * 0.52(0.18) * 0.71(0.12) * 0.69(0.12) * 0.70(0.16) * 0.72(0.14) * 0.80(0.12) *
'WordsAndPhrases-weka.fil (50) 0.96(0.06) | 0.68(0.12) * 0.63(0.14) * 0.56(0.15) * 0.67(0.16) * 0.57(0.14) * 0.50(0.14) * 0.65(0.14) * 0.46(0.16) * 0.64(0.13) * 0.52(0.15) * 0.53(0.16) * 0.74(0.11) * 0.72(0.12) * 0.74(0.12) * 0.71(0.14) * 0.80(0.13) *

(v/ /*) | (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/0/6) (0/2/4) (0/0/6) (0/1/5) (0/0/6)

Key:
(1) bayes.BayesNet '-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimate.SimpleEstimator -- -A 0.5' 746037443258775954
(2) bayes.NaiveBayesMultinomial '' 5932177440181257085
(3) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(4) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"' -6585883636378691736
(5) rules.JRip '-F 3 -N 2.0 -O 2 -S 1' -6589312996832147161
(6) rules.Ridor '-F 3 -S 1 -N 2.0' -7261533075088314436
(7) rules.PART '-M 2 -C 0.25 -Q 1' 8121455039782598361
(8) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943
(9) trees.J48 '-C 0.25 -M 2' -217733168393644444
(10) trees.FT '-I 15 -F 0 -M 15 -W 0.0' -1113212459618105000
(11) trees.BFTree '-S 1 -M 2 -N 5 -C 1.0 -P POSTPRUNED' -7035607375962528217
(12) trees.REPTree '-M 2 -V 0.0010 -N 3 -S 1 -L -1' -9216785998198681299
(13) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.J48 -- -C 0.25 -M 2' -7378107808933117974
(14) meta.LogitBoost '-P 100 -F 0 -R 1 -L -1.7976931348623157E308 -H 1.0 -S 1 -I 10 -W trees.DecisionStump' -3905660358715833753
(15) meta.END '-S 1 -I 10 -W meta.nestedDichotomies.ND -- -S 1 -W trees.J48 -- -C 0.25 -M 2' -4143242362912214956
(16) meta.RandomSubSpace '-P 0.5 -S 1 -I 10 -W trees.REPTree -- -M 2 -V 0.0010 -N 3 -S 1 -L -1' 1278172513912424947
(17) meta.FilteredClassifier '-F \"supervised.attribute.Discretize -R first-last\" -W trees.J48 -- -C 0.25 -M 2' -4523450618538717400