إقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Combining Different Approaches to Improve Arabic Text Documents Classification

دمج طرق مختلفة لتحسين تصنيف المستندات النصية العربية

أقر بأن ما اشتملت عليه هذه الرسالة إنما هي نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل، أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أية مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name: Hassan M. Dawoud	اسم الطالب: حسن محمد داود
Signature:	التوقيع: [illegible]
Date:	التاريخ: 20/10/2013

Computer Engineering Department
Faculty of Engineering
Deanery of Higher Studies
Islamic University – Gaza
Palestine

Combining Different Approaches to Improve Arabic Text Documents Classification

Hassan Mohammad Dawoud

Supervisor

Prof. Ibrahim S. I. Abuhaiba

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

1434H (2013)

الرقم ...ج س غ/35/... Ref

التاريخ 2013/10/20 Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **حسن محمد حسن داود** لنيل درجة الماجستير في **كلية *الهندسة*** قسم **هندسة الحاسوب** وموضوعها:

دمج طرق مختلفة لتحسين تصنيف المستندات النصية العربية

Combining Different Approaches to Improve Arabic Text Documents Classification

وبعد المناقشة التي تمت اليوم الأحد 15 ذو الحجة 1434هـ، الموافق 2013/10/20م الساعة الثالثة مساءً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

أ.د. إبراهيم سليمان أبو هيبة	**مشرفاً ورئيساً**	إبراهيم أبو هيبة [illegible]
د. محمد أحمد الحنجوري	**مناقشاً داخلياً**	[illegible]
د. تامر سعد فطاير	**مناقشاً خارجيًا**	تامر سعد فطاير [illegible]

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *الهندسة*** / قسم **هندسة الحاسوب**.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

مساعد نائب الرئيس للبحث العلمي وللدراسات العليا

[illegible]

أ.د. فؤاد علي العاجز

Dedication

To My Father, and Mother,

To My Wife,

To My Son,

To My Sisters and Brothers,

To My Friends,

To those who gave me all kinds of support,

To all, I dedicate this work.

Acknowledgements

Praise is to ***Allah****, the Almighty for having guided me at every stage of my life.*

I would also like to take this opportunity to thank my research supervisor, ***Prof. Ibrahim S. I. Abuhaiba*** *for giving me the opportunity to work with him and guiding and helping me throughout this research and other courses.*

Also I would like to thank ***Dr. Mohammed A. Alhanjouri*** *and* ***Dr. Tamer S. Fatayer*** *for their encouragement and insightful comments.*

Table of Contents

LIST OF ABBREVIATIONS

AdaBoost	Adaptive Boosting
Bagging	Bootstrap Aggregating
DF	Document Frequency
DT	Decision Tree
IDF	Inverse Document Frequency
kNN	k Nearest Neighbor
kNNM	k Nearest Neighbor Model-based
LDC	Linguistic Data Consortium
LVQ	Learning Vector Quantization
MLP	Multilayer Perceptron
NB	Naïve Bayes
NN	Neural Networks
OSAC	Open Source Arabic Corpora
OWA	Ordered Weighted Averaging
RBFNN	Radial Basis Function Neural Network
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TC	Text Classification , Text Categorization
TF	Term frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TR-Classifiers	Trigger Classifiers

LIST OF FIGURES

LIST OF TABLES

دمج طرق مختلفة لتحسين تصنيف المستندات النصية العربية

حسن محمد داود

الملخص

تصنيف المستندات النصية هي عملية يتم فيها تصنيف المستندات إلى مجموعات محددة مسبقا بناءً على محتويات هذه المستندات. هنالك العديد من الابحاث التي تمت لتصنيف المستندات العربية باستخدام خوارزميات تصنيف مختلفة. الهدف الرئيسي من هذا البحث هو تحسين دقة تصنيف المستندات العربية باستخدام طرق دمج مختلفة بين المصنفات.

في هذا البحث سنقوم باستخدام عدة طرق لدمج المصنفات للحصول على دقة تصنيف عالية. لقد قمنا باستخدام اربع طرق لدمج العديد من المصنفات. الطريقة الاولى وتسمى Fixed rule combination تعتمد على دمج المصنفات بإستخدام قواعد دمج ثابتة لدمج نتائج مصنفات مختلفة وقد حققت هذه الطريقة نتائج ممتازة باستخدام قاعدة التصويت بالاغلبية وقد وصلت دقة التصنيف الى 95.3% باستخدام سبعة مصنفات مختلفة , وقد احتاج بناء هذا النموذج الى 835.94 ثانية.

الطريقة الثانية التي تم استخدامها تسمى Stacking، حيث يتم فيها تصنيف المستندات على مرحلتين, المرحلة الاولى تتم باستخدام عدة مصنفات تسمى المصنفات الأساسية، ويتم بعد ذلك استخدام نتائج التصنيف كمدخلات للمرحلة الثانية التي تتم باستخدام مصنف واحد مثل Naïve Bayes حيث يعتمد تعليم هذا المصنف على نتائج المرحلة الاولى بالاضافة الى التصنيفات الصحيحة للمستندات التى سيتم تصنيفها.

لقد حققت هذه الطريقة دقة عالية في التصنيف وصلت الى 99.2% باستخدام Naïve Bayes في المرحلة الثانية من التصنيف و 99.4% عند استخدام Linear Regression , لكن هذه الطريقة تحتاج الى وقت اطول في عملية التصنيف لانها تتكون من مرحلتين, لذلك فقد احتاج هذا النموذج الى 1962.73 ثانية لبناءه باستخدام Naïve Bayes و الى 3718.07 ثانية باستخدام Linear Regression.

الطريقة الثالثة تمت باستخدام AdaBoost لتعزيز كفاءة المصنف C4.5 وقد حققت هذه الطريقة دقة تصنيف عالية تصل الى 95.3% عند تكرار عملية تعليم المصنف خمس مرات خلال فترة زمنية تساوي 1174.58 ثانية, ووصلت الى 99.5% عند تكرار العملية عشر مرات خلال 1965.72 ثانية.

التجربة الاخيرة التي تمت باستخدام Bagging، حيث ان هذه الطريقة تهدف الى تحسين استقرار ودقة خوارزميات التصنيف، وقد تم استخدام المصنف Decision Tree مع هذه الطريقة وحصلنا على نتائج عالية لدقة تصنيف المستندات تصل الى 93.6% عند تكرار تعليم المصنف خمس مرات وذلك خلال 295.85 ثانية و الى 99.4 % عند تكرار العملية عشر مرات خلال 470.99 ثانية.

لقد اعتمدنا في تجاربنا على اداة WEKA واداة RapidMiner ولقد تم استخدام ثلاث مجموعات من النصوص العربية وهي BBC Arabic, CNN Arabic, OSAC وذلك للتأكد من دقة النتائج. جميع التجارب تمت باستخدام جهاز ذو معالج من نوع Core i3 بسرعة 2.2GHz و ذاكرة عشوائية سعتها 4GB. لقد تمت مقارنة النتائج التي حصلنا عليها من دمج المصنفات بالطرق التى تستخدام مصنف واحد فقط، وقد أظهرت طرق الدمج تفوق واضح في دقة تصنيف المستندات العربية.

Combining Different Approaches to Improve Arabic Text Documents Classification

Hassan Mohammad Dawoud

ABSTRACT

Text classification is the process of classifying documents into a predefined set of categories based on their content. A variety of classifiers are used to classify Arabic text documents. The main objective of this research is to improve Arabic text documents classification by combining different classification algorithms. To achieve this objective we build four models using different combination methods.

The first combined model was built using fixed combination rules, we used five fixed rules to combine different classifiers; and for each rule we used different number of classifiers; the best classification accuracy was achieved using majority voting rule and it was 95.3% using seven classifiers, the time required to build this model was 835.94 seconds.

The second combination approach we used was stacking, which consists of two stages of classification; the first stage was done by the base classifiers, where the second one was done by a Meta classifier. In our experiments we used two different Meta classifiers Naïve Bayes and Linear Regression; and we used different number of base classifiers. Stacking achieved a very high classification accuracy of 99.2% when using Naïve Bayes as a Meta classifier and 99.4% when using Linear Regression as a Meta classifier. Stacking needed a long time to build the models because it consists of two stages of learning and it was 1962.73 seconds using naïve Bayes and 3718.07 seconds using Linear Regression.

The third experiment was done using AdaBoost; we boosted C4.5 classifier with different number of iterations. Boosting improves the classification accuracy of C4.5 classifier; it was 95.3% using 5 iterations and needed 1174.58 seconds to build the model, where the accuracy was 99.5% using 10 iterations and needed 1965.72 seconds to build the model.

The fourth approach was done using Bagging, which was designed to improve the stability and accuracy of machine learning algorithms, we used decision tree with bagging, the results were 93.7% achieved through 295.85 seconds when using 5 iterations and 99.4% when using 10 iteration which needed 470.99 seconds. We used three datasets to test the combined models, BBC Arabic, CNN Arabic and OSAC datasets. The experiments were done using WEKA and RapidMiner data mining tools. We used a platform of Intel Core i3 Processing power of 2.2 GHz CPU with 4GB RAM.

The results of all models showed that combining classifiers can effectively improve the accuracy of Arabic text documents classification.

Keywords: *Text classification, Combining classifiers, Fixed combining rules, Stacking, Boosting, Bagging.*

CHAPTER 1

INTRODUCTION

TEXT Classification (TC) is a technique often used as a basis for applications in document processing, Web mining, topic identifications, text filtering and documents organization etc. Many methods and algorithms have been applied to the problem of text classification. These methods vary in their accuracy. Assessment of different methods by experiment is the basis for choosing a classifier as a solution to a particular problem instance. There are several methods used to classify text such as Support Vector Machine (SVM), K-Nearest Neighbor (kNN), Artificial Neural Networks (ANN), Naïve Bayes Classifier (NB), and Decision Trees (DT). Often none of the basic set of traditional classifiers, ranging from Bayes-normal to Decision Trees, Neural Networks and Support Vector Classifiers is powerful enough to distinguish the pattern classes optimally; it has become clear that for more complicated data sets the traditional set of classifiers can be improved by various types of combining rules [1]. So, for practical purposes, we need an effective methodology for combining them.

Several researches have reported that combining classifiers can improve the accuracy of a standalone classifier, so based on these researches we will combine different classification models to improve Arabic text documents classification.

According to Dietterich [2], there are three main motivations to combine classifiers, the worst case, the best case and the computational motivation:

- **Statistical motivation:** it is possible to avoid the worst classifier by averaging several classifiers. It was confirmed theoretically by Fumera and Roli [3]. This simple combination was demonstrated to be efficient in many applications.
- **Representational motivation:** under particular situations, fusion of multiple classifiers can improve the performance of the best individual classifier. It happens when the optimal classifier for a problem is outside the considered "classifier space". There are many experimental evidences that it is possible if the classifiers in an ensemble make different errors. This assumption has a theoretical support in some cases when linear combination is performed.

- **Computational motivation:** some algorithms perform an optimization task in order to learn and suffer from local minima. Algorithms such as the back propagation for neural networks are initialized randomly in order to avoid locally optimum solutions. In this case, it is a difficult task to find the best classifier, and it is often used several (hundreds or even thousands) initializations in order to find a presumable optimal classifier. Combination of such classifiers showed to stabilize and improve the best single classifier result.

According to the benefits of combining classifiers, we will use these techniques in classifying Arabic text documents. The idea of combining classifiers is motivated by the observation of their complementary characteristics. It is desirable to take advantage of the strengths of individual classifiers and to avoid their weakness, resulting in the improvement of classification accuracy.

1.1 Combining Classifiers

The general idea of combining classifiers can be summarized by the use of a methodology to create an ensemble of learners and to produce a final decision given the outputs of those learners. This kind of models is intuitive since it imitates our nature to seek several opinions before making a crucial decision [4].

The research field of multiple classifier systems becomes very popular after the half of the 1990 decade, with many papers published on the creation of ensembles of classifiers that provided some theoretical insights of why combining classifiers could be interesting. Classifier ensemble is a set of learning machines whose decisions are combined to improve performance of the pattern recognition system. Much of the efforts in classifier combination researches focus on improving the accuracy of difficult problems, managing weaknesses and strengths of each model in order to give the best possible decision. The use of combination of multiple classifiers was demonstrated to be effective, under some conditions, for several pattern recognition applications. Many studies showed that classification problems are often more accurate when using combination of classifiers rather than an individual base learner [5].

In this research we will use four models to combine classifiers to improve the classification of Arabic text documents. These models are Fixed Combining Rules, Stacking, AdaBoost and Bagging.

The fixed combining rules provide the classification decision by combining the outputs of several classifiers [6]. In this approach, all classifiers in the model are learned and each classifier give its decision, then the combiner uses the results of classifiers to give the final decision according to the rule used for combination. Many rules such as majority voting, maximum rule, minimum rule, average rule and product rule can be used in the combiner [7].

The second approach is stacking algorithm [8]. Stacking is probably the most popular meta-learning technique. Stacking is usually employed to combine models built by different classifiers. The stacking algorithm is based on two level of classification. The first level contains the base classifiers which are trained using the original dataset. Then a new dataset is generated using the original dataset and the prediction of base classifiers. This dataset is used to learn the Meta classifier. This classifier combines the different predictions into a final one [4].

The third algorithm that we will use in our research is AdaBoost [9]. AdaBoost tries to combine weak base classifier in order to produce an accurate "strong" classifier. The approach is an iterative process that builds an ensemble of classifiers. The algorithm trains classifier sequentially, a new model per round. At the end of each round, the misclassified patterns are weighted in order to be considered more important in the next round, so that the subsequent models compensate error made by earlier classifiers. The learning algorithm of the classifier used in AdaBoost must allow the use of a weight for each training pattern. The idea is to give higher weights to the patterns that are misclassified and in the next iteration try to construct a classifier capable of classify correctly these kinds of patterns [4].

The fourth approach that we will use is the Bagging. The Bagging technique (bootstrap aggregating) [10] is based on the idea that bootstrap samples of the original training set will present a small change with respect to the original training set, but sufficient difference to produce diverse classifiers. Each member of the ensemble is trained using a different training set, and the predictions are combined by averaging or voting. The different datasets are generated by sampling from the original set, choosing N items uniformly at random with replacement [6]. All of these models will be explained in chapter 3.

1.2 Arabic Language

Arabic Language is one of the widely used languages in the world. Arabic language is a Semitic language that has a complex and much morphology than English; it is a highly inflected language and that due to this complex morphology [11].

Arabic Language consists of 28 alphabet characters: ا ب ت ث ج ح خ د ذ ر ز ش س ش ص ض ع ط ظ ع غ ف ق ك ل و ي. In addition to the hamza (ء) which is considered as a letter by some Arabic linguistics. Arabic is written from right to left. Arabic letters have different styles when appearing in a word depending on the letter position at beginning, middle or end of a word and on whether the letter can be connected to its neighbor letters or not [12].

Arabic words have two genders, feminine and masculine; three numbers, singular, dual, and plural; and three grammatical cases, nominative, accusative, and genitive. A noun has the nominative case when it is subject; accusative when it is the object of a verb; and the genitive when it is the object of a preposition. Words are classified into three main parts of speech, nouns (including adjectives and adverbs), verbs, and particles. All verbs and some nouns are morphologically derived from list of roots. Words are formed by the following fixed patterns, the prefixes and suffixes are added to the word to indicate its number, gender and tense [12].

1.3 Arabic Language Challenges

Arabic is a challenging language for a number of reasons [13]:

1. Orthographic with diacritics is less ambiguous and more phonetic in Arabic, certain combinations of characters can be written in different ways. For example, sometimes in glyphs combining HAMZA with ALEF (أ) the HAMZA is dropped (ا). This makes the glyph ambiguous as to whether the HAMZA is present.
2. Arabic has a very complex morphology recording as compared to English language. For example, to convey the possessive, a word shall have the letter (ى) attached to it as a suffix. There is no disjoint Arabic-equivalent of "my".
3. Arabic words are derived: Arabic words are usually derived from a root (a simple bare verb form) that usually contains three letters. In some derivations,

one or more of the root letters may be dropped. In such cases tracing the root of the derived word would be a much more difficult problem.

4. Broken plurals are common. Broken plurals are somewhat like irregular English plurals except that they often do not resemble the singular form as closely as irregular plurals resemble the singular in English. Because broken plurals do not obey normal morphological rules, they are not handled by existing stemmers.
5. In Arabic we have short vowels which give different pronunciation. Grammatically they are required but omitted in written Arabic texts.
6. Arabic synonyms are widespread. Arabic is considered one of the richest languages in the world. This makes exact keyword match is inadequate for Arabic retrieval and classification

1.4 Topic Area

Text classification has been considered as a vital method to manage and process a vast amount of documents in digital forms that are widespread and continuously increasing. In general, text classification can be applied in important operations such as real time sorting of files into folder hierarchies, topic identifications, dynamic task-based interests, automatic meta-data organization, text filtering and documents organization for databases and web pages.

There are a lot of researches for text classification using different classification techniques where, a lot of these researches are applied to English documents, but in Arabic it is still limited [11, 12, 14, 15, 16]. All previous researchers applied single classifiers to classify Arabic documents, but in this research we will combine multiple classifiers aiming to more accurate classification decision.

1.5 Thesis Questions

The main research problem is classifying Arabic text documents. There are many researches that aimed to classify Arabic documents using a single classifier. These works may have more accurate results if we use a combination of different classification models. Where several studies have reported that combining classifiers can improve the accuracy of a standalone classifier, so based on these researches the main objective of this research is to classify Arabic text by using a combination of different classification models to improve Arabic text documents classification in terms of accuracy.

1.6 Thesis Significance

Text classification has always been an important application and research topic since the inception of digital documents. Today, text classification is a necessity due to the very large amount of text documents that we have to deal with it daily. The current study has a valuable significance, because it derived from the urgent need of classifying the huge number of electronic Arabic documents. These documents are available through the rapid and increasing growth of the Internet. Also the manual classification of such huge documents needs a considerable time and effort as well as it is costly.

There are many classification algorithms applied on Arabic text using single classifier, such as SVM, Naïve Bayes, and kNN, these algorithms have given good classification accuracy, but this accuracy can be improved using a combination approach.

The combination of classifiers has not applied on Arabic documents classification before, although many researchers have shown that combining different classifiers can improve classification accuracy [17, 18, 19, 20]. Many researchers make a comparison between the best individual classifier and the combined approach, and showed that the performance of the combined approach is superior [21].

1.7 Thesis Contribution

The contributions of this research are highlighted hereunder:

1- Combining different classification algorithms using five fixed combining rules to improve the accuracy of Arabic text documents classification.
2- Combining different classification approaches using stacking algorithm with two different Meta classifiers to improve the accuracy of Arabic text documents classification.
3- Applying boosting algorithm on a weak classifier to enhance the accuracy of that classifier on classifying Arabic text documents.
4- Applying bagging algorithm on a weak classifier to enhance the accuracy of that classifier on classifying Arabic text documents.

5- As a results, the classification accuracy achieved by combining classifiers are :

   a. 95.3% of classification accuracy achieved by combining seven classifiers using majority voting rule as a fixed combination rule.
   b. 99.4% of classification accuracy achieved by combining five classifiers using stacking algorithm.
   c. 99.5% of classification accuracy achieved by using AdaBoost algorithm.
   d. 99.4% of classification accuracy achieved by using Bagging algorithm.

1.8 Thesis Organization

The rest of this thesis is organized as follows:

Chapter 2 introduces the related work; we will show some researches that have been done in classifying Arabic text documents. The related works are divided into three categories; in the first one we will show some researches which have been used classification algorithms onto Arabic text documents, in the next one we will show some researches that have been compared between classification algorithms which applied onto Arabic Documents, where in the last section we will show some related works that have using combined classifiers.

In Chapter 3, we will overview some background theory, in the first part we will explain some classification algorithms that we will use in our approaches. Then we will explain the four combining approaches that we will use in our thesis.

In Chapter 4, we will show the methodology that we will use in our thesis such as pre-processing steps, term weighting, datasets and evaluation metrics.

Chapter 5 shows the experimental results of our work. All results of our approaches will be discussed and analyzed; also we will compare our approaches with different single classifiers which have been applied in Arabic text documents.

Finally the conclusion of the research will be in Chapter 6. In this chapter we will summarizes the research, remarks, and some notes around the work.

CHAPTER 2
RELATED WORK

Many researchers have been worked on text classification in English and other European languages. However, researches on text classification for Arabic language are fairly limited **[11, 12, 14, 15, 16]**. In this chapter we will show some researches that are related to our work, these researches are categorized to:

1- Applying classification algorithms on Arabic text.
2- Comparing between different classification algorithms applied on Arabic text.
3- Combining classification algorithms.

2.1 Applying Classification Algorithms on Arabic Text

Mesleh [22] applied SVMs to classify Arabic articles with Chi Square feature selection in the pre-processing step. The reported F-measure by Mesleh is 88.11%. Mesleh also compared six feature selection methods with SVMs. He concludes that Chi Square method is the best. He used an in-house collected corpus from online Arabic newspaper archives, including Al-Jazeera, Al-Nahar, Al-hayat, Al-Ahram, and Al-Dostor as well as a few other specialized websites. The collected corpus contains 1445 documents that vary in length. These documents fall into 9 classification categories that vary in the number of documents (Computer, Economics, Education, Engineering, Law, Medicine, Politics, Religion and Sports). In the pre-processing step, each article in the data set is processed to remove the digits and punctuation marks. He has applied normalization of some Arabic letters such as the normalization of (hamza) in all its forms to (alef). Also, all non Arabic text was filtered, and he does not apply stemming.

Harrag and El-Qawasmah [23] applied neural networks (NN) on Arabic text. Their experimental results show that using NN with Singular Value Decomposition (SVD) as a feature selection technique gives better result 88.3% than the basic NN (without SVD) 85.7%. They also experienced scalability problem with high dimensional text dataset using NN. Harrag collected his corpus from Hadith encyclopedia from the nine books. It contains 435 documents belonging to 14 categories. He applied light stemming and stop words removal on his corpus. Term Frequency-Inverse Document Frequency (TF-IDF) is used as a weighting scheme.

El-Kourdi et. al. [24] classified Arabic text documents automatically using NB. The average accuracy reported was about 68.78%, and the best accuracy reported was about 92.8%. El-Kourdi used a corpus of 1500 text documents belonging to 5 categories; each category contains 300 text documents. All words in the documents are converted to their roots. The vocabulary size of resultant corpus is 2,000 terms/roots. Cross-validation was used for evaluation.

Maximum entropy was used by **El-Halees [25]** for Arabic text classification, and by **Sawaf [26]** to classify and cluster News articles. The best classification accuracy reported by El-Halees was 80.4% and 62.7% by Sawaf.

kNN has been applied by **Al-Shalabi [27]** on Arabic text. They used TF-IDF as a weighting scheme and got accuracy of 95%. They also applied stemming and feature selection. The authors reported in their paper the problem of lacking freely publically availability of Arabic corpus. They collected a corpus from newspapers (Al-Jazeera, An-Nahar, Al-Hayat, Al-Ahram, and Ad-Dostor) and from Arabic Agriculture Organization website. The corpus consists of 621 documents belonging to 1 of 6 categories (politics 111, economic 179, sport 96, health and medicine 114, health and cancer 27, agriculture 100). They preprocessed the corpus by applying stop words removal and light stemming.

2.2 Comparing between different Classification Algorithms applied on Arabic Text

Hmeidi [28] compared kNN and SVM for Arabic text classification; they used full word features and considered TF-IDF as the weighting method for feature selection, and CHI statistics for ranking metrics. Hmeidi showed that both SVM and kNN have superior performance, and SVM has better accuracy and time. Authors collected documents from online newspaper (Al-Ra'i and Ad-Dostor). They collected 2206 documents for training and 29 documents for testing. The collected documents belong to one of two categories (sport and economic).

Abbas [29] compared Triggers Classifier (TR-Classifier) and kNN to identify Arabic topic. kNN uses the whole vocabulary (800), while TR uses reduced vocabulary (300), the average recall and precision for kNN and TR are 0.75, 0.70 and 0.89, 0.86% respectively. Abbas collected 9,000 articles from Omani newspaper (Al-Watan) of year 2004. The corpus belongs to 1 of 6 categories (culture, economic, religious, local news, international news). The corpus includes 10M word including stop words. After removing stop words and infrequent words the vocabulary size became 7M words. TF-IDF was used as weighting schemes.

Duwairi [12] compared three popular text classification algorithms; (kNN, NB, and Distance-Based classifier). Duwairi experimental results show that NB outperforms the other two algorithms. Duwairi collected 1,000 text documents belonging to 1of 10 categories (sport, economic, internet, art, animals, technology, plants, religious, politics, and medicine). Each category contains 100 documents. She preprocessed the corpus by applying stop words removal and stemming. She used 50% for training and 50% for testing.

Kannan [16] also compared three classification algorithms on Arabic text. The three algorithms were kNN, NB, and Rocchio. Kannan revealed that NB is the best performing algorithm. The author collected the corpus from online newspapers (Al-Jazeera, An-Nahar, Al-Hayat, Al-Ahram, and Ad-Dostor). The corpus consists of 1,445 documents belonging to 9 categories (medicine 232, sport 232, religious 227, economic 220, politics 184, engineering 115, low 97, computer 70, and education 68). They applied light stemming for feature reduction. Cross-validation was performed for evaluation.

Al-Harbi [11] evaluated the performance of two popular text classification algorithms (SVMs and C5.0) to classify Arabic text using seven Arabic corpora. The average accuracy achieved by SVMs is 68.65%, while the average accuracy achieved by C5.0 is 78.42%. One of the goals of their paper is to compile Arabic corpora to be benchmark corpora. The authors compiled 7 corpora consisting of 17,658 documents and 11,500,000 words including stop words. The corpora are not available publically.

Bawaneh [30] applied kNN and NB on Arabic text and concluded that kNN has better performance than NB, they also concluded that feature selection and the size of training set and the value of K affect the performance of classification. The researchers also posed the problem of unavailability of freely accessible Arabic corpus. The in-house collected corpus consists of 242 documents belonging to 1of 6 categories. Authors applied light stemming as a feature reduction technique and TF-IDF as weighting scheme, they also performed cross-validation test.

El-Halees [15] compared six well know classifiers applied on Arabic text; ANN, SVM, NB, kNN, Maximum Entropy and Decision Tree. El-Halees showed that the NB and SVMs are the best classifiers in terms of F-Measure with 91% and 88%, respectively. El-Halees also applied information grain feature selection; the reported F-Measure was 83% and 88% for NB and SVMs, respectively. El-Halees collected Arabic documents collected from the Internet. It is mainly collected from Aljazeera Arabic news channel. The documents categorized into six domains: politics, sports, culture and arts, science and technology, economy and health. The author applied stop words removal and normalization and used 10-folds cross-validation for testing.

Saad [14] compares the impact of text preprocessing on Arabic text classification using popular text classification algorithms; Decision Tree, K Nearest Neighbors, Support Vector Machines, Naïve Bayes and its variations. He applied different term weighting schemes, and Arabic morphological analysis (stemming and light stemming). Saad used seven Arabic corpora (3 in-house collected and 4 existing corpora). The experiments showed that light stemming with term pruning is the best feature reduction technique which reduced features to average of 50% of the original feature space. Support Vector Machines and Naïve Bayes variations achieved the best classification accuracy and outperform other algorithms. Weighting schemes impact the performance of distance based classifier.

2.3 Combining Classification Algorithms

Many researches show that combining classifiers can enhance the results of classification in general, but combining classifiers does not used previously to classify Arabic documents. The idea of combining classifiers is motivated by the observation of their complementary characteristics. It is desirable to take advantage of the strengths of individual classifiers and to avoid their weakness, resulting in the improvement of classification accuracy [17]**.**

El-Halees [19] presented a combined approach that automatically extracts opinions from Arabic documents. They used a combined approach that consists of three methods. At the beginning, lexicon based method is used to classify as much documents as possible. The resultant classified documents are used as training set for maximum entropy method which subsequently classifies some other documents. Finally, k-Nearest Neighbor method used the classified documents from lexicon based method and maximum entropy as training set and classifies the rest of the documents. Experiments showed that in average, the accuracy moved from 50% when using only lexicon based method to 60% when used lexicon based method and maximum entropy together, to 80% when using the three combined methods.

Danesh [31] proposed a novel approach in text classification. Their approach is a supervised method, meaning that the list of categories should be defined and a set of training data should be provided for training the system. Documents are represented as vectors where each component is associated with a particular word. They propose voting methods and ordered weighted averaging (OWA) operator and Decision Template method for combining classifiers. Experimental results show that these methods decrease the classification error to 15 percent as measured on 2000 training data from 20 newsgroups dataset.

Fujino [32] provide good statistical classifiers with generalization ability for multi-label categorization and present a classifier design method based on approach combination and F1-score maximization. They design multiple models for binary classification per category, and then they combine these models to maximize the F1-score of a training dataset. Experimental results confirmed that the method was useful especially for datasets where there were many combinations of category labels.

Y. Bi [17] presents an investigation into the combination of four different classification methods for text categorization using Dempster's rule of combination. These methods include the SVM, kNN, kNN model-based approach (kNNM), and Rocchio methods. They present an approach for effectively combining the different classification methods. Then, they apply these methods to a benchmark data collection of 20-newsgroup, individually and in combination. Experimental results show that the performance of the best combination of the different classifiers on the 10 groups of the benchmark data can achieve 91.07% classification accuracy, which is 2.68% better than SVM.

CHAPTER 3

BACKGROUND

3.1 Text classification

Text classification, also known as text categorization and occasionally as topic spotting, is the process of algorithmically analyzing an electronic document to assign this document to one of predefined categories. This assignment can be used for classification, filtering, and retrieval purposes. Many researchers have explored a variety of machine learning methods for text classification. Machine learning algorithms can be divided into two types supervised and unsupervised learning algorithms. Supervised learning algorithms operate by learning the objective function from a set of training examples and then applying the learned function to the target set. Unsupervised learning operates by trying to find useful relations between the elements of the target set. Text categorization can be characterized as a supervised learning problem. We have a set of examples (documents) that have been correctly categorized (usually by human indexers). This set is then used to train a classifier based on a machine learning algorithm. The trained classifier is then used to categorize the target set.

More formally, let $C = \{c_1, . . . , c_n\}$ be a set of categories and $D = \{d_1, . . . , d_N\}$ be a set of documents. Given a set of examples of the form $\langle di, yi \rangle$ where $d_i \in D$, and if $d_i \in c_j$ then $y_j = 1$, otherwise $y_j = 0$, the objective is to learn a function f such that $f(x) = 1$ if $x \in c_j$ and $f(x) = 0$ if $x \notin c_j$. This function is called the classifier [20].

3.2 Text Classification Methods

A wide variety of techniques have been designed for text classification. In this section, we will show in a brief some techniques that are used for Arabic text classification, and then we will show in details different approaches to combine classifiers.

3.2.1 Decision Trees

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the sub-tree rooted at the new node [33].

In the general case, in order to develop a decision tree, the designer has to consider the following design elements in the training phase [34]:

- At each node, the set of candidate questions to be asked has to be decided. Each question corresponds to a specific binary split into two descendant nodes. Splitting of a node is equivalent to the split of the subset X_t into two disjoint descendant subsets X_{tY}, X_{tN}. The first of the two consists of the vectors in X_t that correspond to the answer "Yes" of the question and those of the second to the "No." The first (root) node of the tree is associated with the training set X. For every split, the following is true:

$$\begin{aligned} X_{tY} \cap X_{tN} &= \phi \\ X_{tY} \cup X_{tN} &= X_t \end{aligned} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.1)$$

  A splitting criterion must be adopted according to which the best split from the set of candidate ones is chosen.

- A stop-splitting rule is required that controls the growth of the tree, and a node is declared as a terminal one (leaf).
- A rule is required that assigns each leaf to a specific class.

There are a number of standard packages for DT learning, and most DT approaches to TC have made use of one such package. Among the most popular ones are Iterative Dichotomiser 3 (ID3), C4.5 and C5 [35].

3.2.2 SVM Classifiers

Support vector machines are based on the Structural Risk Minimization principle from computational learning theory. SVM seeks a decision surface to separate the training data points into two classes and makes decisions based on the support vectors that are selected as the only effective elements in the training set [36].

Given a set of N linearly separable points $S = \{x_i \in R^n \mid i = 1,2,..., N\}$, each point x_i belongs to one of the two classes, labeled as $y_i \in \{-1,+1\}$. A separating hyper-plane divides S into 2 sides, each side containing points with the same class label only. The separating hyper-plane can be identified by the pair (*w*, *b*) that satisfies:

$$y = w.x + b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.2)}$$

$$and \quad \begin{Bmatrix} w.x_i + b \geq +1 & \text{if } y_i = +1 \\ w.x_i + b \leq -1 & \text{if } y_i = -1 \end{Bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.3)}$$

for i = 1, 2, …., N and where

$$w.x = \sum_i w_i.x_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.4)}$$

for vectors **w** and **x**.

Thus the goal of the SVM learning is to find the optimal separating hyper plane that has the maximal margin to both sides. This can be formularized as:

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.5)}$$

$$\text{Subject to} \begin{Bmatrix} w.x_i + b \geq +1 & \text{if } y_i = +1 \\ w.x_i + b \leq -1 & \text{if } y_i = -1 \end{Bmatrix} \text{ for } i = 1, 2, \ldots, N \quad \ldots\ldots\ldots\ldots \text{(3.6)}$$

SVMs can be used for both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. With the new dimension, it searches for the linear optimal separating hyper plane. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyper plane. SVMs find this hyper plane using support vectors and margins (defined by the support vectors).

Figure 3.1 shows support vectors and how margins are maximized.

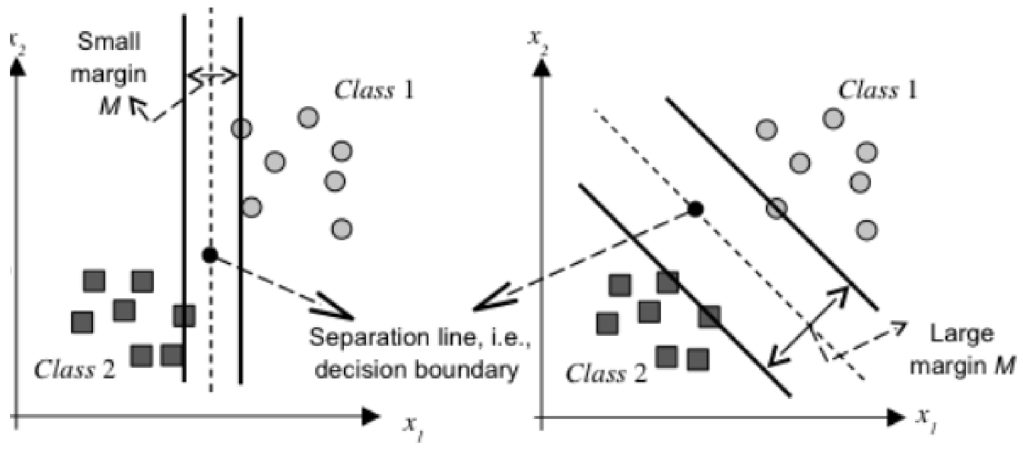

Figure 3.1: Support Vectors **[37]**.

The following summarizes SVMs steps [14]:

- Map the data to a predetermined very high-dimensional space via a kernel function.
- Find the hyper plane that maximizes the margin between the two classes.
- If data are not separable find the hyper plane that maximizes the margin and minimizes the (a weighted average of the) misclassifications.

3.2.3 Learning Vector Quantization Algorithms

The LVQ model is a classification method based on Kohonen 1995. The original of Kohonen model is known as Self Organizing Map networks (SOM) [38]. It uses a competitive unsupervised learning algorithm [39]. In this model, the output layer has neurons equal to the number of classes [38]. The weight vector associate to each output unit is known as codebook vector. Each class of input space is represented by its own codebook vector. Codebook vectors are defined according to the specific task. There are different types or algorithms for LVQ algorithm which used as optimization of original LVQ [40]:

- LVQ1 (LVQ)
- LVQ2.1
- LVQ3
- Optimized learning rate algorithms OLVQ1
- OLVQ3 for the classification task

The LVQ network finds the output unit *w* that is closest to the input vector *x*. If *x* and *w* belong to the same class, then we move the weights toward the new input vector; if *x* and *w*, belong to different classes, then we move the weights away from this input vector [40].

The LVQ algorithm is a competitive network, and thus, for each training vector, output units compete among themselves in order to find the winner according to some metric. The LVQ algorithm uses the Euclidean distance to find the winner unit. Only the winner unit (i.e., the output unit with the smallest Euclidean distance with regard to the input vector) will modify its weights using the LVQ learning rule. The basic LVQ algorithm is shown in **Figure 3.2** [39, 40, 41].

Algorithm 3.1: LVQ Algorithm
Purpose: Learning LVQ algorithm on a given dataset
Input: Training dataset
Output: A Learned neural network based on LVQ algorithm
Procedure:

```
1. Initialize the codebook vectors $W_i$ and the learning rate $\alpha$
2. Randomly select an input vector $X$
3. Find the winner unit closest to the input vector (i.e., the codebook vector $W_c$ with the smallest Euclidean distance with regard to the input vector $X$):
   $\|X - W_c\| = \min_k \|X - W_k\|, \; c = \arg\min_k \|X - W_k\|$
4. Modify the weights of the winner unit:
   • If $W_c$ and $X$ belong to the same class (the classification has been correct)
       $W_c(t+1) = W_c(t) + \alpha(t)[X(t) - W_c(t)]$ .
   • If $W_c$ and $X$ belong to different classes (the classification has not been correct)
       $W_c(t+1) = W_c(t) - \alpha(t)[X(t) - W_c(t)]$
5. Reduce the learning rate α
6. Repeat from step 2 until the neural network is stabilized or until a fixed number of iterations have been carried out.
```

Figure 3.2: LVQ algorithm procedure **[40]**.

The learning rate α(t) (0 < α(t) < 1) is a monotonically decreasing function of time which controls how quickly the weight vector is allowed to change. It is recommended that α(t) should already initially be rather small, say, smaller than 0.3 and it continues decreasing to a given threshold very close to 0.

3.2.4 Naïve Bayes

The Naive Bayes (NB) classifier is a probabilistic model that uses the joint probabilities of terms and categories to estimate the probabilities of categories given a test document [33]. The naive part of the classifier comes from the simplifying assumption that all terms are conditionally independent of each other given a category. Because of this independence assumption, the parameters for each term can be learned separately and this simplifies and speeds the computation operations compared to non-naive Bayes classifiers.

There are two common event models for NB text classification, discussed by [42], multinomial model and multivariate Bernoulli model. In both models classification of test documents is performed by applying the Bayes' rule [33]:

$$P(c_i \mid d_j) = \frac{P(c_i).P(d_j \mid c_i)}{P(d_j)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.7)}$$

Where d_j is a test document and c_i is a category. The posterior probability of each category c_i given the test document d_j, i.e. P(ci | dj), is calculated and the category with the highest probability is assigned to d_j.

In order to calculate $P(c_i \mid d_j)$, $P(c_i)$ and $P(d_j \mid c_i)$ have to be estimated from the training set of documents. Note that $P(d_j)$ is same for each category so we can eliminate it from the computation. The category prior probability $P(c_i)$ can be estimated as follows:

$$P(c_i) = \frac{\sum_{j=1}^{N} y(d_j, c_i)}{N} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.8)}$$

Where, N is number of training documents and $y(d_j, c_i)$ is defined as follows:

$$y(d_j, c_i) = \begin{cases} 1 & \text{if } d_j \in c_i \\ 0 & otherwise \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.9)}$$

So, prior probability of category c_i is estimated by the fraction of documents in the training set belonging to c_i. $P(d_j \mid c_i)$ parameters are estimated in different ways by the multinomial model and multivariate Bernoulli model.

a. Multinomial Model

In the multinomial model a document *dj* is an ordered sequence of term events, drawn from the term space *T*. The Naive Bayes assumption is that the probability of each term event is independent of term's context, position in the document, and length of the document. So, each document *dj* is drawn from a multinomial distribution of terms with number of independent trials equal to the length of *dj*. The probability of a document *dj* given its category *ci* can be approximated as:

$$P(d_j \mid c_i) \approx \prod_{k=1}^{|d_j|} P(t_k \mid c_i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.10)}$$

Where | d_j | is the number of terms in document d_j; and t_k is the k_{th} term occurring in document d_j. Thus the estimation of $P(d_j \mid c_i)$ is reduced to estimating each $P(t_k \mid c_i)$ independently. The following Bayesian estimate is used for $P(t_k \mid c_i)$:

$$P(t_k \mid c_i) = \frac{1 + TF\ (t_k, c_i)}{|T| + \sum_{t_l \in |T|} TF\ (t_l, c_i)} \quad \text{..............................} (3.11)$$

Here, $TF\ (t_l, c_i)$ is the total number of times term t_k occurs in the training set documents belonging to category c_i. The summation term in the denominator stands for the total number of term occurrences in the training set documents belonging to category c_i [43].

b. Multivariate Bernoulli Model

In Multivariate Bernoulli model a document is represented by a vector of binary features indicating the terms that occur and that do not occur in the document. Here, the document is the event and absence or presence of terms is the attributes of the event. The Naive Bayes assumption is that the probability of each term being present in a document is independent of the presence of other terms in a document. To state differently, the absence or presence of each term is dependent only on the category of the document. Then, $P(d_j \mid c_i)$ the probability of a document given its category is simply the product of the probability of the attribute values over all term attributes:

$$P(d_j \mid c_i) = \prod_{k=1}^{|T|} B_{jk} . P(t_k \mid c_i) + (1 - B_{jk})(1 - P(t_k \mid c_i))) \quad \text{........................} (3.12)$$

Where |T| is the number of terms in the training set and B_{jk} is defined as follows:

$$B_{jk} = \begin{cases} 1 & \text{if term } t \text{ appears i n } d_j \\ 0 & \text{otherwise} \end{cases} \quad \text{..} (3.13)$$

Thus, a document can be seen as a collection of multiple independent Bernoulli experiments, one for each term in the term space. The probabilities of each of these term events are defined by the class conditional term probabilities $P(t_k \mid c_i)$. We can estimate the probability of term t_k in category ci as follows:

$$P(t_k \mid c_i) = \frac{1 + \sum_{j=1}^{N} B_{jk} . y(d_j, c_i)}{2 + \sum_{j=1}^{N} y(d_j, c_i)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.14)$$

Where, N is number of training documents and $y(d_j \mid c_i)$ is defined as shown above in equation (3.14) [43].

3.2.5 K-Nearest Neighbor Classifier

The kNN algorithm is a similarity-based learning algorithm that has been shown to be very effective for a variety of problem domains including text categorization [33, 44]. Given a test document, the kNN algorithm finds the k nearest neighbors among the training documents, and uses the categories of the k neighbors to weight the category candidates. The similarity score of each neighbor document to the test document is used as the weight of the categories of the neighbor document.

Similarity may be measured by for example the Euclidean distance or the cosine between the two document vectors. The Euclidean distance is used as a conventional method for measuring distance between two documents, the formula of the Euclidean distance between documents *$d_1(w_{11},w_{12},\ldots,w_{1n})$* and *$d_2(w_{21},w_{22},\ldots,w_{2n})$* is as follow [45]:

$$E(d_1, d_2) = \sqrt{\sum_{i=1}^{n} (w_{2i} - w_{1i})^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.15)$$

kNN has a set of drawbacks. kNN is a lazy learning example-based method that does not have a off-line training phase. The main computation is the on-line scoring of training documents given a test document in order to find the k nearest neighbors, this makes kNN not efficient because nearly all computation takes place at classification time rather than when the training examples are first encountered, kNN time complexity is O(N*M) where N is number of training documents and M is the number of terms in the super vector. Moreover, kNN classifier has a major drawback of selecting the value of k; the success of classification is very much dependent on this value.

3.2.6 Radial Basis Function Neural Network

A Radial Basis Function Neural Network (RBFNN) is a special type of ANN with several distinctive features [46]. The radial basis function (RBF) network is a feed-forward artificial neural network that uses radial basis functions as activation functions. It is based on linear combination of radial basis functions. It has been shown that RBFNN had a simple structure and many excellent performances. Therefore, RBFNN has been widely used for pattern classification, functional approximation, signal processing, mixture models and many other fields [40].

The structure of a RBFNN consists of three different layers, namely the input layer, the hidden layer, and the output layer. Each RBF is a fixed two layer NN that hides all nonlinearities in its special hidden layer and performs a linear combination in the output layer as shown in Figure 3.2 [47]. The parameters that determine the output values are the centers of the hidden RBF units and the weights of the synapses from the hidden to the output layer. One important difference is that Multilayer Perceptron (MLP) is most of the times a multilayer network, whereas each RBFNN consists of only one hidden layer with radial basis function neurons.

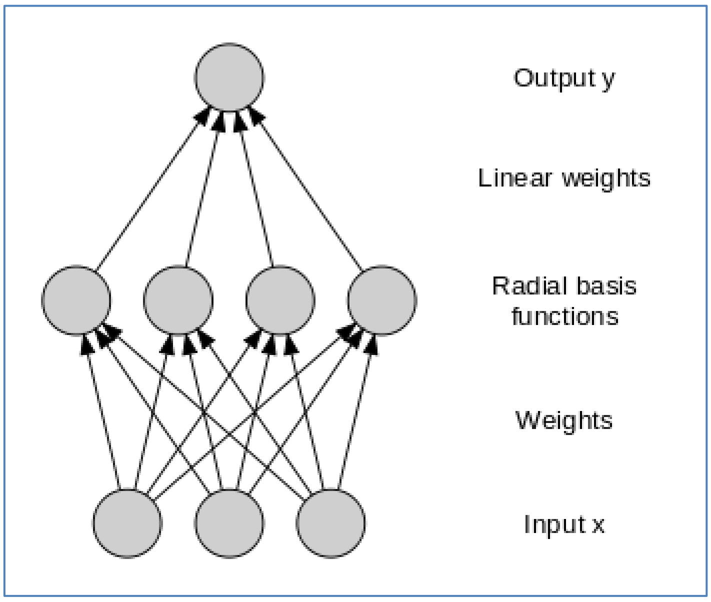

Figure 3.3: Architecture of Radial Basis Function (RBFNN) [47].

In a typical implementation, the hidden nodes and the output nodes of a MLP share a common computation and neuronal model, while the RBFNN neurons of the RBF hidden layer plays a totally different role in comparison to the output nodes which are most of the times linear and perform a linear combination of the hidden layer neurons responses [40].

3.3 Combining Classifiers

The practice of combining multiple classifiers into ensembles is inspired by the notion that the combined opinions of a number of human experts is more likely to be correct than that of a single expert. Ideally, a classifier ensemble will collectively perform better than any individual classifier in the ensemble. There are many different ways in which one can combine classifiers into ensembles, each of which can work well in certain scenarios but not in others. There are a wide variety of techniques and approaches available to combine classifiers. A good selection can potentially result in better ensemble success rates than any one of the component classifiers could provide individually. A poor selection, however, can result in reduced performance relative to what one would have received from a single well chosen classifier. In either case, the use of an ensemble will most often increase training and classification computational demands, as well as system complexity [18].

3.3.1 Reasons for combining classifiers

There are a number of important reasons why combined classifiers can in fact be a better choice than a single classifier. Dietterich [2] has organized these reasons into three categories:

a. Statistical

Suppose we have a labeled data set Z and a number of different classifiers with a good performance on Z. We can pick a single classifier as the solution, running onto the risk of making a bad choice for the problem. For example, suppose that we run the 1-nn classifier or a decision tree classifier for L different subsets of features thereby obtaining L classifiers with zero resubstitution error. Although these classifiers are indistinguishable with respect to their (resubstitution) training error, they may have different generalization performances. Instead of picking just one classifier, a safer option would be to use them all and “average” their outputs. The new classifier might not be better than the single best classifier but will diminish or eliminate the risk of picking an inadequate single classifier. Dietterich gives a graphical illustration of this argument as shown in Figure 3.4. The outer circle denotes the space of all classifiers.

The shaded inner region contains all classifiers with good performances on the training data. The best classifier for the problem (supposedly with a good performance on the training data too) is denoted by D*. The hope is that some form of aggregating of the L classifiers will bring the resultant classifier closer to D* than a classifier randomly chosen from the classifier space would be.

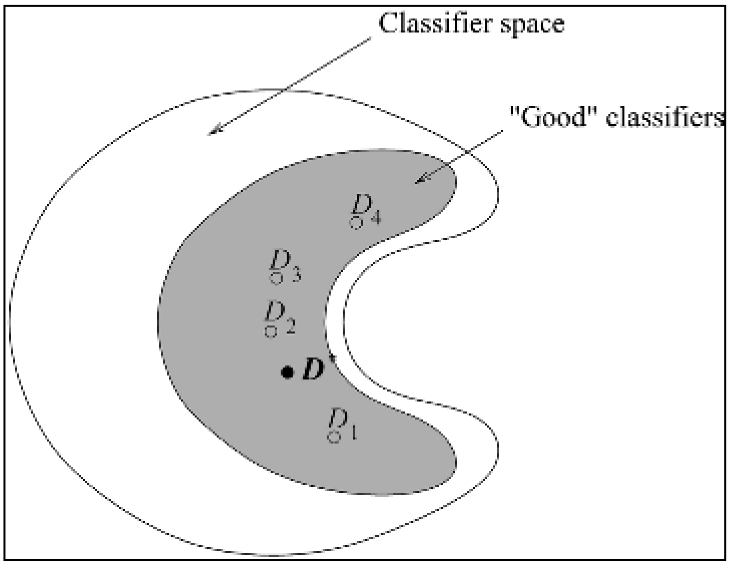

Figure 3.4: The statistical reason for combining classifiers. D1 through D4 are trained classifiers and D* is the theoretically optimum classifier. The shaded area represents the area in classifier space of classifiers that perform well on a given data set.

b. Computational

Some classifiers train using hill-climbing or random search techniques. Many classifiers also rely on random initializations which can influence the minima in error space that they converge to during training. Training multiple feed-forward neural networks, for example, on the same training data can very well result in significantly different trained classifiers. Although they will each most likely move closer to the optimal solution during training, there is no guarantee that they will do so in the same way or that they will converge to the same solution. Aggregating such classifiers into an ensemble can take advantage of the multiplicity of solutions offered by the different classifiers, none of which may be optimal, in order to arrive at a solution that is better than the solution offered by any one individual classifier. The computational argument is illustrated in Figure 3.5. Classifiers D1 through D4 move closer during training to the theoretically optimum classifier (D*) from their initial pre-training locations in classifier space. Each of D1 through D4 offers a different solution after training, and combining these solutions can result in better solution overall than the solution offered by any of the individual classifiers.

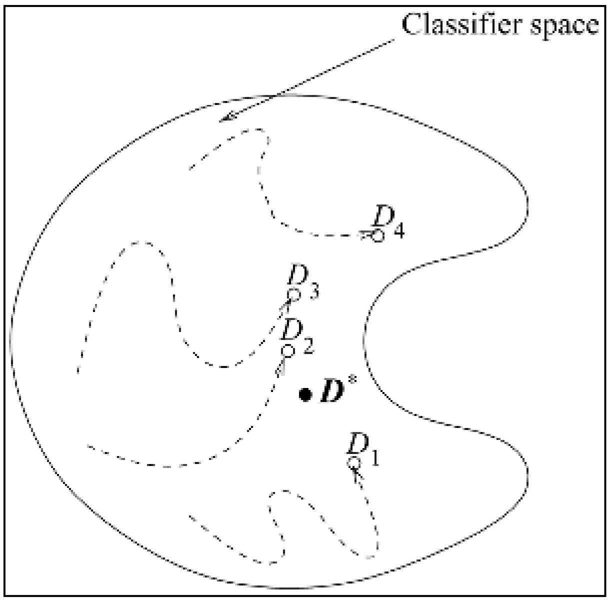

Figure 3.5: The computational reason for combining classifiers. D1 through D4 are hill climbing classifiers and D* is the theoretically optimum classifier. The dashed lines show the trajectories of each classifier during training.

The computation argument highlights the particular appropriateness of instable classifiers for ensemble classification. Instable classifiers are classifiers where small changes in the training set can have a significant effect on the classifier output. The use of multiple instable classifiers trained on slightly different but potentially overlapping training sets can lead to a variety of useful solutions that can be fruitfully combined.

c. Representational

There is no guarantee that the types of classifiers that one is using for a particular problem could ever converge to the theoretically optimal solution during training. For example, say a researcher mistakenly believes that a given problem is linear, and decides to use only linear classifiers. In reality, the optimal classifier will be non-linear, so it is not possible that any of the linear classifiers under consideration will perform optimally. However, an ensemble of linear classifiers can approximate a non-linear decision boundary, and could therefore potentially perform better than any single linear classifier possibly could. The representational argument is illustrated in Figure 3.6.

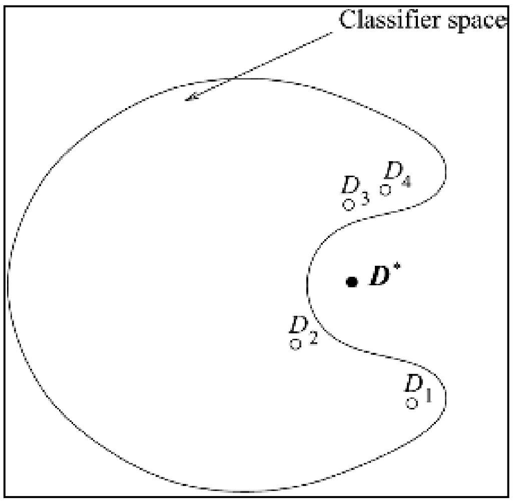

Figure 3.6: The representational reason for combining classifiers. The closed shape represents the range of classifiers that one is able or willing to construct. D1 through D4 represent four trained classifiers and D* represents the theoretically optimal classifier.

An alternative solution to the non-linear problem presented above, of course, would be to use a more sophisticated single classifier. One must remember, however, that the primary disadvantage of using classifier ensembles over single classifiers is that they introduce added complexity to the solution, but this complexity depends on the selected classifiers that we want to combine them. It may well be that an ensemble of simple classifiers can perform faster and be implemented more easily and intuitively than a single complex classifier. The argument against using classifier ensembles is therefore reversed in cases such as this.

The previous three reasons motivate us to use classifiers combination to improve the accuracy of classifying arabic text documents.

In the next sections we will show different approaches to combine classifiers, which we will use in our research.

3.3.2 Fixed Combining Rules

Fixed combining rules are the simplest combination approach and it is probably the most commonly used in the multiple classifier system [48]. This combination approach is called non-trainable combiner, because combiners are ready to operate as soon as the classifiers are trained and they do not require any further training of the ensemble as a whole [49]. A theoretical framework for fixed rules combination was proposed by Kittler [7]. It includes the sum, product, max, min, average and median rules.

Consider a pattern recognition problem where the pattern x is to be assigned to one of m possible classes $(\omega_1, \dots, \omega_m)$. Let us assume that we have R classifiers, the feature vector $x^{(i)}$ represents the given pattern on the *i*th classifier. In the feature space each class ω_k is modeled by the probability density function $p(x^{(i)} \mid \omega_k)$ and its a priori probability of occurrence $P(\omega_k)$ [48].

According to Bayesian theory, for given features $x^{(i)}$, $i \in \{1,\dots,R\}$ the pattern x should be assigned to class ω_j with the maximal value of the a posteriori probability such that:

$$f(x) = \omega_j,\ \mathrm{j} = \arg\max_{\mathrm{k}} \mathrm{P}(\omega_{\mathrm{k}} \mid x^{(1)},\dots,x^{(R)}) \quad \text{.....................} \quad (3.16)$$

Using Bayes' Theorem the a posteriori probability is:

$$\mathrm{P}(\omega_{\mathrm{k}} \mid x^{(1)},\dots,x^{(R)}) = \frac{P(x^{(1)},\dots,x^{(R)} \mid \omega_k) P(\omega_k)}{P(x^{(1)},\dots,x^{(R)})} \quad \text{..................} \quad (3.17)$$

Now the following fixed rules can be used to combine classifiers [49]:

a. Product Rule

Let us assume that the probability distributions $P(x^{(1)},\dots,x^{(R)} \mid \omega_k)$ are conditionally statistically independent. Then

$$P(x^{(1)},\dots,x^{(R)} \mid \omega_k) = \prod_{i=1}^{R} p(x^{(i)} \mid \omega_k) \quad \text{........................} \quad (3.18)$$

and the decision rule

$$f(x) = \omega_j \quad , \quad j = \arg\max_k P(\omega_k) \prod_i p(x^{(i)} \mid \omega_k) \quad \text{..................} \quad (3.19)$$

The product rule multiplies the score provided by each base classifiers and assigns the class label with the maximum score to given input pattern [7].

b. Majority Voting Rule

The voting method finds the class output of each classifier and counts its output as a vote for a class, and assigns the input pattern to the class with the majority vote as the following [49]:

$P(\omega_k \mid x^{(R)})$ will produce binary valued functions Δ_{ki} like :

$$\Delta_{ki} = \begin{cases} 1 & if \quad P(\omega_k \mid x^{(R)}) = \max_j P(\omega_j \mid x^{(R)}) \\ 0 & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.20)$$

Assuming that each a priori probability is equal, this leads to the following decision rule:

$$f(x) = \omega_j \ , \quad j = \arg \max_k \sum_i \Delta_{ki} \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.21)$$

The sum on the right side of equation (3.21) simply counts the votes received from each individual classifier [49].

c. The average rule

Let us assume that the probability distributions $P(x^{(1)},\ldots, x^{(R)} \mid \omega_k)$ are conditionally statistically independent [49]. Then

$$P(x^{(1)},\ldots, x^{(R)} \mid \omega_k) = \frac{1}{R}\sum_{i=1}^{R} p(x^{(i)} \mid \omega_k) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.22)$$

Then the average decision rule is:

$$f(x) = \omega_j \ , \quad j = \arg \max_k \ \frac{1}{R}\sum_{i=1}^{R} p(\omega_k \mid x^{(i)}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.23)$$

The average rule can be defined as finding the maximum value of the average of $P(\omega_k \mid x^{(i)})$ and assigns the class label with it to the given input pattern [49].

d. The Maximum rule

Maximum rule is based on the information provided by the maximum value of $P(x^{(i)} \mid \omega_k)$ across all class labels. It finds the maximum score of each class between the classifiers and assigns the input pattern to the class with the maximum score among the maximum scores as the following [49]:

$$f(x) = \omega_j \ , \quad j = \arg \max_k \{\max \ P(x^{(i)} \mid \omega_k)\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.24)$$

e. The Minimum rule

The minimum rule finds the minimum score of each class between the classifiers and assigns the input pattern to the class with the maximum score among the maximum scores; that means the Minimum rule selects the class that having the least objection. The decision of the minimum rule is defined as [49]:

$$f(x) = \omega_j \quad , \quad j = \arg \max_k \{\min \; P(x^{(i)} \mid \omega_k)\} \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; (3.25)$$

Figure 3.6 shows the architecture of combined classifiers using fixed rules.

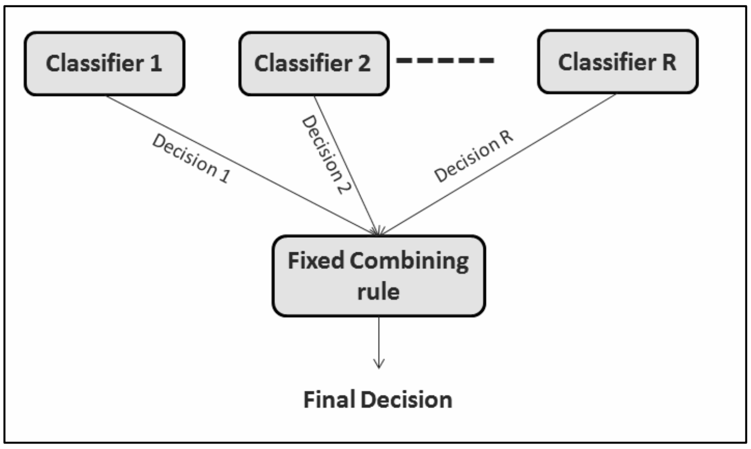

Figure 3.7: The architecture of combined classifiers using fixed rules.

As shown in Figure 3.7, the dataset (which are Arabic text documents in our case) are used to train and test the system, each classifier in the system is trained using the training data set, and then give an output. The outputs of all classifiers are combined using one of fixed rules that mentioned previously to give the final decision.

3.3.3 Stacking

Stacked generalization is a way of combining multiple models that have been learned for a classification task [8].Typically, different learning algorithms learn different models for the task, and in the most common form of stacking the first step is to collect the output of each model into a new set of data. For each instance in the original training set, this data set represents every model's prediction of that instance's class, along with its true classification. During this step, care is taken to ensure that the models are formed from a batch of training data that does not include the instance in question, in just the same way as ordinary cross-validation. The new data are treated as the data for another learning problem, and in the second step a learning algorithm is employed to solve this problem. The original data and the models constructed for them in the first step are referred to as level-0 data and level-0 models, respectively, while the set of cross-validated data and the second-stage learning algorithm are referred to as level-1 data and the level-1 generalizer.

The stacking framework

Stacking is concerned with combining multiple classifiers generated by using different learning algorithms $L_1, \ldots, L_N$ on a single dataset S, which consists of examples $s_i = (x_i, y_i)$, pairs of feature vectors (x_i) and their classifications (y_i). In the first phase, a set of base-level classifiers $C_1, C_2, \ldots, C_N$ is generated, where $C_i = L_i(S)$. In the second phase, a meta-level classifier is learned that combines the outputs of the base-level classifiers [50].

To generate a training set for learning the meta-level classifier, a leave-one-out or a cross validation procedure is applied. For leave-one-out, we apply each of the base-level learning algorithms to almost the entire dataset, leaving one example for testing: $\forall\ i = 1,\ldots, n : \forall\ k = 1,\ldots, N : C_k^i = L_k(S - s_i)$ where *i* is the number of examples, *k* is the number of base classifiers. We then use the learned classifiers to generate predictions for $s_i : \hat{y}_k^i = C_k^i(x_i)$. The meta-level dataset consists of examples of the form $((\hat{y}_i^1,\ldots, \hat{y}_i^n), y_i)$, where the features are the predictions of the base-level classifiers and the class is the correct class of the example at hand. When performing 10-fold cross validation, instead of leaving out one example at a time, subsets of size one-tenth of the original dataset are left out and the predictions of the learned base-level classifiers obtained on these.

Figure 3.8 shows pseudo code for the stacking algorithm, the first step is to learn the first level (base classifiers) using the original dataset as shown in line 2, in lines 4 to 10 a new data set is generated. The new dataset is then used to learn the second level (Meta) classifier.

Algorithm 3.2: Stacking Algorithm
Purpose: Combining classifiers using Stacking
Input:

- Data set D = $\{(x_1, y_1), (x_2, y_2),\ldots(x_m, y_m)\}$
- First-level learning algorithms $L_1 \ldots\ldots\ldots L_T$
- Second-level learning algorithm L.

Output: A composite model $H(x) = \hat{h}(h_1(x)\ldots\ldots h_T(x))$
Procedure:

```
1  for t = 1 to T
2  $h_t = L_t(D)$          Train a first-level individual learner ht by applying the first-level
3   endfor                 learning algorithm $L_t$ to the original data set D
4  $\hat{D} = \phi$          Generate a new data set
```

```
5  for i = 1 to m
6     for t = 1 to T
7  $z_{it} = h_t(x_i)$           Use $h_t$ to classify the training example $x_i$
8     endfor
9  $D^{\wedge} = D^{\wedge} \cup \{((z_{i1}, z_{i2}, \ldots\ldots z_{iT}), y_i)\}$
10 endfor
11 $h^{\wedge} = L(D^{\wedge})$          Train the second-level learner h0 by applying the second-level
                        learning algorithm L to the new data set $D_0$

Output: $H(x) = h^{\wedge}(h_1(x)\ldots\ldots h_T(x))$
```

Figure 3.8: Pseudo code for the stacking algorithm.

3.3.4 The Boosting Algorithm

Boosting is a general method for improving the performance of a weak learner. The method works by iteratively invoking a weak learner, on training data that is taken from various distributions. Similar to bagging, the classifiers are generated by resampling the training set. The classifiers are then combined into a single strong composite classifier. Contrary to bagging, the resampling mechanism in boosting improves the sample in order to provide the most useful sample for each of consecutive iteration.

The AdaBoost Algorithm

AdaBoost (Adaptive Boosting) [51] is a popular ensemble algorithm that improves the simple boosting algorithm via an iterative process. The main idea behind this algorithm is to give more focus to patterns that are harder to classify. The amount of focus is quantified by a weight that is assigned to every pattern in the training set. Initially, the same weight is assigned to all the patterns. In each iteration the weights of all misclassified instances are increased while the weights of correctly classified instances are decreased. As a consequence, the weak learner is forced to focus on the difficult instances of the training set by performing additional iterations and creating more classifiers.

Furthermore, a weight is assigned to every individual classifier. This weight measures the overall accuracy of the classifier and is a function of the total weight of the correctly classified patterns. Thus, higher weights are given to more accurate classifiers. These weights are used for the classification of new patterns. This iterative procedure provides a series of classifiers that complement one another.

In AdaBoost, the input includes a dataset *D* of *d* class-labeled objects, an integer k specifying the number of iterations, and a classification learning scheme as shown in Figure 3.9.

Each object in the dataset is assigned a weight. Initially, all weights are assigned a same value of *1/d*. The algorithm repeats *k* times. At each time, a model M_i is built on current dataset D_i, which is obtained by sampling with replacement on original training dataset *D*.

Algorithm 3.3: AdaBoost algorithm
Purpose: Generating an ensemble of classifiers using AdaBoost
Input:

- D, training set
- k, the number of rounds
- A classification learning algorithm

Output: A composite model
Procedure:

```
1   Initialize the weight of each object in D to 1/d;
2   for i = 1 to k do
3   Sample D with replacement according to the object weights to obtain Di;
4   Use training set Di to derive a model, Mi;
5   Compute the error rate error(Mi) of Mi;
6   if error(Mi) > 0.5 then
7   Reinitialize the weights to 1/d;
8   Go back to step 3 and try again;
9   endif
10  Update and normalize the weight of each object;
11  endfor
```

Figure 3.9: The framework of AdaBoost algorithm.

The error rate of *Mi* is the sum of the weights of all objects in *Di* that *Mi* misclassified:

$$error(M_i) = \sum_{j=1}^{d} w_j \times err(X_j) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.26)$$

Where $err(X_j) = 1$ if X_j is misclassified and $err(X_j) = 0$ otherwise.

Then the weight of each object is updated so that the weights of misclassified objects are increased and the weights of correctly classified objects are decreased. This can be done by multiplying the weights of each correctly classified object by *error(Mi)/(1 – error(Mi))*. The weights of all objects are then normalized so that the sum of them is equal to 1. In order to keep this constraint, the weight of each object is divided by the sum of the new weights.

After k rounds, a composite model will be generated, or an ensemble of classifiers, which is then used to classify new data. When a new object X comes, it is classified through the steps shown in Figure 3.10.

Algorithm 3.4: Classification Using AdaBoost algorithm
Purpose: Classify a new object using ensemble of classifiers build by AdaBoost
Input: a new object
Output: the class of an object
Procedure:

```
1  Initialize weight of each class to 0;
2  for $i = 1$ to $k$ do
3      Get weight $w_i$ of classifier $M_i$ ;
4      Get class prediction for X from $M_i$: $c = M_i(X)$;
5      Add $w_i$ to weight for class $c$;
6  endfor
7  Return the class with the largest weight;
```

Figure 3.10: The steps of classifying a new object by the ensemble of classifiers build by AdaBoost Algorithm.

The weight w_i of each classifier M_i is calculated by this equation:

$$w_i = \log \frac{1 - error(M_i)}{error(M_i)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ (3.27)$$

AdaBoost seems to improve the performance accuracy for two main reasons:

1- It generates a final classifier whose error on the training set is small by combining many hypotheses whose error may be large.

2- It produces a combined classifier whose variance is significantly lower than the variances produced by the weak base learner.

However, AdaBoost sometimes fails to improve the performance of the base inducer. According [9], the main reason for AdaBoost failure is overfitting. The objective of boosting is to construct a composite classifier that performs well on the data by iteratively improving the classification accuracy. Nevertheless, a large number of iterations may result in an overcomplex composite classifier, which is significantly less accurate than a single classifier. One possible way to avoid overfitting is to keep the number of iterations as small as possible.

3.3.5 Bagging Algorithm

Bagging (bootstrap aggregating) [6] is a simple effective method for generating an ensemble of classifiers. The ensemble of classifiers, which is created by this method, combines the outputs of various learned classifiers into a single classification. This results in a classifier whose accuracy is higher than the accuracy of each individual classifier. Specifically, each classifier in the ensemble is trained on a sample of instances taken with replacement (allowing repetitions) from the training set.

To ensure that there is a sufficient number of training instances in every sample, it is common to set the size of each sample to the size of the original training set. **Figure 3.11** presents the pseudo-code for building an ensemble of classifiers using the bagging. The algorithm receives an induction algorithm I which is used for training all members of the ensemble. The stopping criterion in line 6 terminates the training when the ensemble size reaches T. One of the main advantages of bagging is that it can be easily implemented in a parallel mode by training the various ensemble classifiers on different processors.
Since sampling with replacement is used, some of the original instances of S may appear more than once in S_t and some may not be included at all. Furthermore, using a large sample size causes individual samples to overlap significantly, with many of the same instances appearing in most samples.

Algorithm 3.5: Bagging Algorithm
Purpose: Generating an ensemble of classifiers using Bagging
Input :

- S, training set
- μ ,the sample size
- T, the number of rounds
- I , A classification learning algorithm

Output: A composite model
Procedure:

```
1  $t \leftarrow 1$
2  repeat
3  $St \leftarrow$ a sample of $\mu$ instances from $S$ with replacement.
4  Construct classifier $Mt$ using $I$ with $St$ as the training set
5  $t \leftarrow t + 1$
6  until $t > T$
```

Figure 3.11: The framework of Bagging algorithm.

So while the training sets in S_t may be different from one another, they are certainly not independent from a statistical stand point. In order to ensure diversity among the ensemble members, a relatively unstable inducer should be used. This will result is an ensemble of sufficiently different classifiers which can be acquired by applying small perturbations to the training set. If a stable inducer is used, the ensemble will be composed of a set of classifiers who produce nearly similar classifications, and thus will unlikely improve the performance accuracy.

In order to classify a new instance, each classifier returns the class prediction for the unknown instance. The composite bagged classifier as shown in **Figure 3.**12 returns the class with the highest number of predictions (also known as majority voting).

Algorithm 3.6: Classification Using Bagging algorithm
Purpose: Classify a new object using ensemble of classifiers build by Bagging
Input: A new object
Output: The class of an object
Procedure:

```
1  $Counter1, . . . , Counter|dom(y)| \leftarrow 0$  { initializes class votes counters }
2  for $i = 1$ to $T$ do
3  $vote_i \leftarrow M_i(x)$ { get predicted class from member i }
4  $Countervote_i \leftarrow Countervote_i + 1$ { increase by 1 the counter of the corresponding class }
5  end for
6  $C \leftarrow$ the class with the largest number votes
7  Return $C$
```

Figure 3.12: The steps of classifying a new object by the ensemble of classifiers build by Bagging Algorithm

Often, bagging produces a combined model that outperforms the model that is built using a single instance of the original data especially for unstable inducers since bagging can eliminate their instability [6].

CHAPTER 4
METHODOLOGY

This chapter explains methodology which we will follow in this research. To implement and evaluate our approaches we will use the following methodology steps as presented in Figure 4.1:

1. **Collecting data:** collect Arabic text documents from different domains**.**
2. **Preprocessing data:** through applying different text pre-processing techniques which include applying different term weighting schemes, and Arabic morphological analysis (stemming and light stemming).
3. **Combining classifiers:** through implementing models by combining different classification algorithms and by using different combining techniques. Combing classifiers can be built on different subsets of features. Feature selection aims at a more efficient computation and a higher accuracy; so we will evaluate different feature selection methods in our experiments.
4. **Evaluate the model:** to evaluate the classification performance of our model, we will use accuracy, precision, recall and f-Measure.
5. Compare the results of combining classifiers with other results using single classifiers.

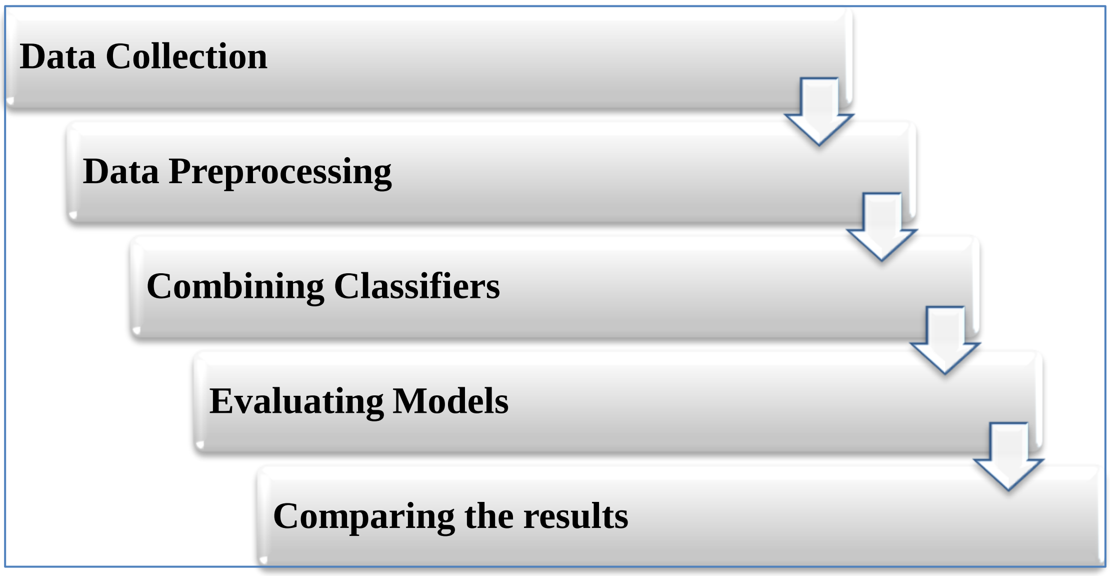

Figure 4.1: Methodology steps

4.1 Data Collection

One of difficulties for Arabic language is the lack of publicly available Arabic corpus for evaluating text categorization algorithms [27]. On the other side, English language has different public data set for English text classification. Researchers in Arabic text classification used their own data sets collected from several Arabic website like Al-Jazeera, Al-Nahar, Al-hayat, Al-Ahram, and Al-Dostor. The collected data has different size and different categories used for training and testing. On the other hand, the Linguistic Data Consortium (LDC) provides two non-free Arabic corpora, the Arabic NEWSWIRE and Arabic Gigaword corpus.

In my research, I will use a freely public data set published by *Saad* in [52]. The first dataset was collected from *CNN Arabic* website. It is free and public and contains a suitable number of documents for the classification process and also suitable to the hardware used in experiments. CNN Arabic dataset has different domains. Table 4.1 presents domains of CNN-Arabic corpus which includes 5070 documents. Each document belongs to 1 of the 6 domains or categories.

Table 4.1: Categories and number of documents per category for CNN Arabic corpus.

Number	Category	Number of text documents
1.	Business	836
2.	Entertainments	474
3.	Middle East News	1462
4.	Science & Technology	526
5.	Sports	762
6.	World News	1010
Total		**5070**

The second dataset to be used is called OSAC. OSAC dataset was collected from multiple websites. The corpus includes 22,429 text documents. Each text document belongs to 1 of 10 categories as shown in Table 4.2.

Table 4.2: Categories and number of documents per category for OSAC dataset.

Number	**Category**	**Number of text documents**
1.	Economic	3102
2.	History	3233
3.	Education and family	3608
4.	Religious and Fatwas	3171
5.	Sport	2419
6.	Health	2296
7.	Astronomy	557
8.	Low	944
9.	Stories	726
10.	Cooking Recipes	2373
Total		**22,429**

The third data set dataset was collected from BBC Arabic website. It is free and public and contains a suitable number of documents for the classification process and also suitable to the hardware used in experiments. BBC Arabic dataset has different domains. Table 4.3 presents domains of BBC-Arabic corpus which includes 4,763 documents. Each document belongs to 1 of the 7 domains or categories.

Table 4.3: Categories and number of documents per category for BBC Arabic corpus.

Number	Category	Number of text documents
1.	Middle East News	2356
2.	World News	1489
3.	Business	296
4.	Science & Technology	232
5.	Sports	219
6.	Entertainments	122
7.	World Press	49
Total		4,763

4.2 Data Preprocessing

Text preprocessing includes many steps, the process starts by tokenizing string to words, after that normalizing tokenized words, the stop word removed and applying stemming algorithm (stemming / light stemming), and finally term weighting for each word , Figure 4.2 shows these steps. In the next section we will describe text preprocess steps applied on Arabic text.

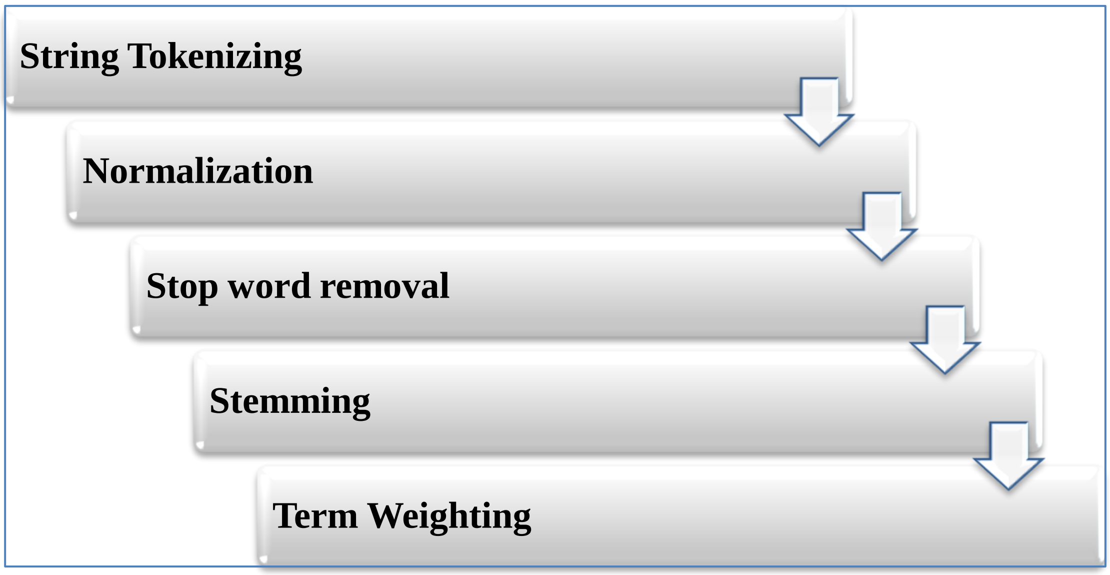

Figure 4.2: Arabic text documents preprocessing steps.

4.2.1 String Tokenizing

Tokenization is very important in natural language processing. It can be seen as a preparation stage for all other natural language processing tasks [53]. Tokenization is the task of separating out words from running text into units. These units could be characters, words, numbers, sentences or any other appropriate unit [54]. The definition of a word here is not the exact syntactic form, which is why we call it a 'token'. In the case of Arabic, where a single word can comprise up to four independent tokens, morphological knowledge needs to be incorporated into the tokenize. However, Tokenization closely related to the morphological analysis. The tokenize process is responsible for defining word boundaries such as white spaces and punctuation marks, multiword expressions, abbreviations and numbers [55].

One of the most useful features in detecting sentences boundaries and tokens is punctuation marks. However, the total number of punctuation marks and symbols used in Arabic corpus was 134, while in the corresponding English corpus only 54 punctuations and symbols were used [55]. There are several methods to apply tokenization; the simplest way we used is extracting any alphanumeric string between two white spaces.

4.2.2 Normalization

Normalization is the process of unification of different forms of the same letter. Before stemming and stop word removal, corpus was normalized as follows:

- Remove punctuation.
- Remove diacritics (primarily weak vowels).
- Remove non letters.
- Replace أ , إ , and آ with ا.
- Replace final ى with ي.
- Replace final ة with ه .

4.2.3 Stop Words

Stop words are frequently occurring, insignificant words that appear in an article or web page (i.e. pronouns, prepositions, conjunctions, etc.). Words like (هذه, تكون, قد, بين

انه, على, أين, كان) are considered stop words. These words carry no information. Stop words are filtered out prior to processing of natural language data [36].

4.2.4 Stemming Algorithms

One of the major techniques that have been used in preprocessing stage in document classification is stemming. Christopher et al. [56] define Stemming as a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. In other words it is the process of removing any affixes (prefixes that added to the beginning of the word, infixes that added to the middle of the word, or/and suffixes that added to the ending of the word) from words to reduce these words to their stems or roots under the assumption that words with the same stem are semantically related.

There are two major approaches that are followed for Arabic stemming. One approach is called light stemming (also called stem-based stemming) by which a word's prefixes and suffixes are removed; the other one called Root-based stemming (also called aggressive stemming) which reduces a word to its root. Another two approaches that have been researched are Statistical stemming and Manual constructing of dictionaries; the last one is not efficient and there for not so popular. Studies show that light stemming outperforms aggressive stemming and other stemming types [57].

1- Root–based Stemmer

Arabic words are formed from abstract forms named roots, the root is the basic form of word from which many derivations can be obtained by attaching certain affixes so we produce many nouns and verbs and adjectives from the same root [58]. A root based stemmer main goal is to extract the basic form for any given word by performing morphological analysis for the word [59], Table 4.4 shows an example root "لعب" and a set (not all) derivations can be obtained from this root [60]:

Table 4.4: Some derivations of the root "لعب".

يلعب	ملعب	لاعب	ملعوب	لعبة
Play	**Playground**	**Player**	**Played**	**Game**

Khoja stemmer [61] basically attempts to find roots for Arabic words which are far more abstract than stems. It first removes prefixes and suffixes, then attempts to find the root for the stripped form. The problem in this stemming technique is that many different word forms are derived from an identical root, and so the root extraction stemmer creates invalid conflation classes that result in an ambiguous query which leads to a poor performance [60].

2- Light Stemmer

Light stemming is to find the representative indexing form of a word by the application of truncation of affixes [62]. The main goal of light stemming is to retain the word mening intact and so improves the retrieval performance of an Arabic information retrieval system. Many light stemming methods like Leah [63] stemmer classifies the affixes to four kinds of affixes: antefixes, prefixes, suffixes and postfixes that can be attached to words. Thus an Arabic word can have a more complicated form if all these affixes are attached to its root. The following example, Table 4.5, shows a sample of a word and its affixes [62] :

Table 4.5: A word and its affixes "ليناقشوهم"

Antefix	Prefix	Core	Suffix	Postfix
ل	ي	ناقش	و	هم

So from the above example we see that if we could remove all affixes of a word then we will get the stemmed word which is not the root but basic word without any affixes and so we maintain the meaning of the word and improve the search effectiveness.

In this research we will apply light stemming and Khoja stemmer on our datasets.

4.2.5 Term weighting

Term weighting is one of pre-processing methods used for enhanced text document presentation as feature vector. Term weighting helps us to locate important terms in a document collection for ranking purposes [64]. There are several term weighting schemes the popular term weighting schemes are Boolean model, Term Frequency (TF), Inverse Document Frequency (IDF), and Term Frequency-Inverse Document Frequency (TF-IDF) [13]. Choosing an appropriate term weighting scheme is more important for text categorization [65].

a. Boolean model

The Boolean model is the simplest retrieval model based on Boolean algebra and set theory. Boolean model indicates to absence or presence of a word with Booleans 0 or 1 respectively [13].

b. Term Frequency

Term frequency TF (*t,d*) is the number that the term *t* occurs in the document *d* [13]. The TF measures the importance of term t_i within the particular document d_j can be calculated by equation [40]:

$$TF_{i,j} = \frac{n_{i,j}}{\sum n_{k,j}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(4.1)}$$

Where

$n_{i,j}$: The number of occurrences of the considered term (t_i) in the document d_j.

$\sum n_{k,j}$: Sum of number of occurrences of all terms in document d_j.

c. Inverse Document Frequency

The inverse document frequency (IDF) is one of the most widely used term weighting schemes for estimating the specificity of term in a document collection [66]. It is based on the idea that if a term appears in only a few documents in the collection, then such a term is expected to be a good discriminator of these documents. The IDF weight of a term t can be calculated from document frequency using the formula [40]:

$$IDF_{t} = \log(\frac{N}{n}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(4.2)}$$

Where

N: number of documents.

n: number of documents with word i.

The IDF of a term is low if it occurs in many documents and high if the term occurs in only a few documents [13].

d. Term Frequency-Inverse Document Frequency

Term Frequency and Inverse Document Frequency (TF-IDF), is a popular method of pre-processing documents in the information retrieval community [65]. TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document [67]. The TF-IDF calculated by using the formula [40]:

$$\text{TF-IDF} = TF_{i,j} \,.\, IDF_t \qquad \text{……………………………………..} \; (4.3)$$

$$\text{TF-IDF} = \frac{n_{i,j}}{\sum n_{k,j}} \,.\, \log(\frac{N}{n}) \qquad \text{……..…………………………..} \; (4.4)$$

In this research, TF-IDF term weighting schema is applied to our datasets, because it is the most popular weighting schema and many researches such as [14] show that it gives a good results.

4.3 Evaluation

There are different measures that we can use to measure classification accuracy. The basic measures that we can use are: accuracy, precision, recall, F-measure. Accuracy as a measure is the number of samples that are correctly classified.

Computation of precision and recall are based on computing confusion matrix [26] as shown in Table 4.6. A confusion matrix is computed by creating two categories, it is a matrix where test cases are distributed as follows:

1- **True positive (TP):** refers to positive instances that are correctly labeled.

2- **False Negative (FN):** are the positive instances that are incorrectly labeled.

3- **False Positive (FP):** are the negative instances that are incorrectly labeled.

4- **True negative (TN)**: refers to negative instances that are correctly labeled.

Table 4.6: Confusion matrix for two class classification problem.

True Positive (TP)	False Negative (FN)
False Positive (FP)	True Negative (TN)

We can compute classifier accuracy as:

$$Accuracy = \frac{\text{number of TP} + \text{number } of \; TN}{\text{number of } TP + FP + FN + TN} \qquad \text{……………………………………..} \; (4.5)$$

The precision is the percentage of predicted documents for the given topic that are correctly classified:

$$Precision = \frac{number\ of\ true\ positives}{number\ of\ true\ positives + false\ positives} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (4.6)$$

Also, we compute recall which is the percentage of the total documents for the given topic that are correctly classified as follows:

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.7)$$

The F measure combines precision and recall. We used the F-measure to evaluate the performance of text classifiers:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.8)$$

So using accuracy, precision, recall and F-measure we can evaluate our system and compare our results with other experiments.

4.4 Text Mining Tools

1- Weka Data Mining Software in Java [68]: Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

2- RapidMiner [69]: RapidMiner provides data mining and machine learning procedures including: data loading and transformation, data preprocessing and visualization, modeling, evaluation, and deployment. RapidMiner is written in the Java programming language. It uses learning schemes and attributes evaluators from the Weka machine learning environment.

CHAPTER 5
EXPERIMENTAL RESULTS

This chapter describes the results and analyses of combining different classification approaches on the selected Arabic datasets. In our experiments we implement four combination approaches. We select different classifiers to use in our experiments such as SVM, Naïve Bayes, C4.5, kNN, Decision Stump, RBFNN and LVQ2.1. These classifiers are selected because they are the most famous classifiers used in other researches to classify Arabic text documents. Also some classifiers such as Decision trees are selected in AdaBoost and Bagging because researchers recommended using these classifiers with such of these combining algorithms.

In our experiments we implement four combined models. The first model is built using fixed combination rules. We examine five fixed rules which mentioned in chapter 3. The second combination approach we build is combining classifiers using stacking. Different numbers of classifiers are used to build different stacked models.

The third model is built using AdaBoost algorithm. We use C4.5 classifier with this algorithm. The last model that we examine is build using Bagging algorithm. We use Decision stump classifier with this algorithm.

The experiments are implemented using three datasets, BBC Arabic, CNN Arabic, OSAC datasets.

Two stemming approaches were used with these datasets, light stemming and Khoja stemmer. We used only TF-IDF term weighting schema.

In the following sections we will examine and analyze the four combined models. Experimental results investigate building models time, precision, recall, F-measure and classifiers accuracy.

5.1 Implementation Environment

The combined models are implemented using two data mining tools, WEKA and RapidMiner. We use WEKA tool to build two models using fixed combining rules and stacking. The AdaBoost and Bagging models were built using RapidMiner.

In our implementation, we use a platform of Intel Core i3 of speed 2.2 GHz, 4 GB of Memory and 64 bit windows 7 operating system.

5.2 Arabic Text Documents classification using fixed combining rules

In this approach we use five different fixed combination rules as the following:

1- Average rule.
2- Product rule.
3- Majority Voting
4- Minimum rule.
5- Maximum rule.

We apply these five fixed rules using different number of classification algorithms; we combine three, five and seven classifiers using these rules.

In every case we use three datasets to confirm our results as we will show in the next sections.

5.2.1 Using three classifiers Model

In the first step we use three classifiers with each rule; the classifiers used at this stage are SVM, Naive Bayes and C4.5.

Table 5.1 shows the results of combining three classifiers using five fixed rules when Appling light stemming on the BBC Arabic data set and using TF-IDF term weighting.

Table 5.1 presents that applying majority voting rule achieved the highest accuracy (94.1%), recall (0.943), precision (0.943) and F-measure (0.943) of classification.

Table 5.1: Accuracy, F-measure and Time of combined classifiers using three classifiers and BBC dataset (TF-IDF and Light stemming).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.937	0.938	0.937	93.7	90.5
Product rule	0.940	0.940	0.941	90.4	105.2
Majority vote	0.943	0.943	0.943	94.1	107.1
Minimum rule	0.895	0.893	0.897	89.6	110.37
Maximum rule	0.880	0.879	0.882	88.2	111.67

Table 5.2 shows the result of combining three classifiers using with five fixed rules when applying Khoja stemmer on BBC Arabic data set and using TF-IDF term weighting.

Table 5.2: Accuracy, F-measure and Time of combined classifiers using three classifiers and BBC dataset (TF-IDF and Khoja stemmer).

Rule	F measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.929	0.929	0.929	92.9	135.18
Product rule	0.911	0.912	0.911	91.3	155.52
Majority vote	0.935	0.935	0.935	93.5	153.65
Minimum rule	0.894	0.891	0.898	90.7	156.50
Maximum rule	0.861	0.860	0.862	86.2	166.06

And from Table 5.2 we notice also that the majority voting rule outperforms (93.5%) over all other fixed combination rules.

To confirm our results we use other dataset as shown in Table 5.3 which shows the result of combining three classifiers using with five fixed rules when Appling light stemming on the CNN Arabic data set and using TF-IDF term weighting.

Table 5.3: Accuracy, F-measure and Time of combined classifiers using three classifiers and CNN dataset (TF-IDF and Light stemming).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.924	0.925	0.924	92.4	468.94
Product rule	0.923	0.923	0.923	92.3	459.58
Majority vote	0.959	0.959	0.959	93.4	457.44
Minimum rule	0.909	0.909	0.910	91.2	451.68
Maximum rule	0.884	0.890	0.879	87.9	453.90

Table 5.4 shows the result of combining three classifiers using with five fixed rules when Appling Khoja stemmer on CNN Arabic data set and using TF-IDF term weighting.

Table 5.4: Accuracy, F-measure and Time of combined classifiers using three classifiers and CNN dataset (TF-IDF and Khoja stemmer).

Rule	F measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.922	0.923	0.922	92.2	308.08
Product rule	0.910	0.911	0.910	91.8	291.54
Majority vote	0.924	0.925	0.924	92.4	299.88
Minimum rule	0.909	0.911	0.908	91.8	298.15
Maximum rule	0.877	0.888	0.867	86.7	298.38

Figure 5.1 summarizes the results in term of accuracy using two datasets and two different stemmers.

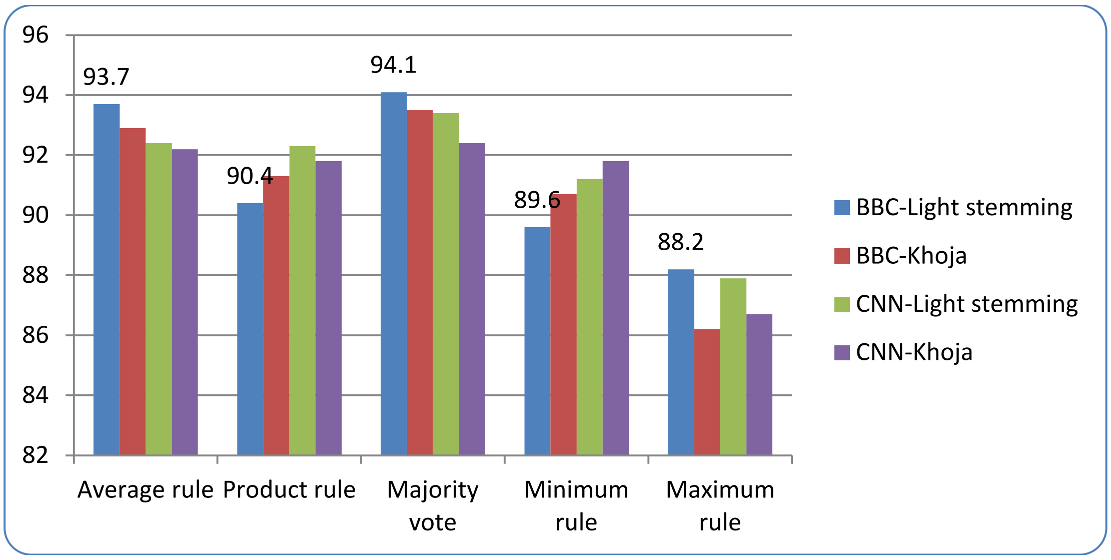

Figure 5.1: Accuracy summarization of three classifiers combined models

From Figure 5.1 we notice that applying majority voting combination rule on BBC Arabic dataset with Light stemming give the highest accuracy (94.1%) of classification.

Figure 5.2 summarizes the results in term of model building time using two datasets and two different stemmers.

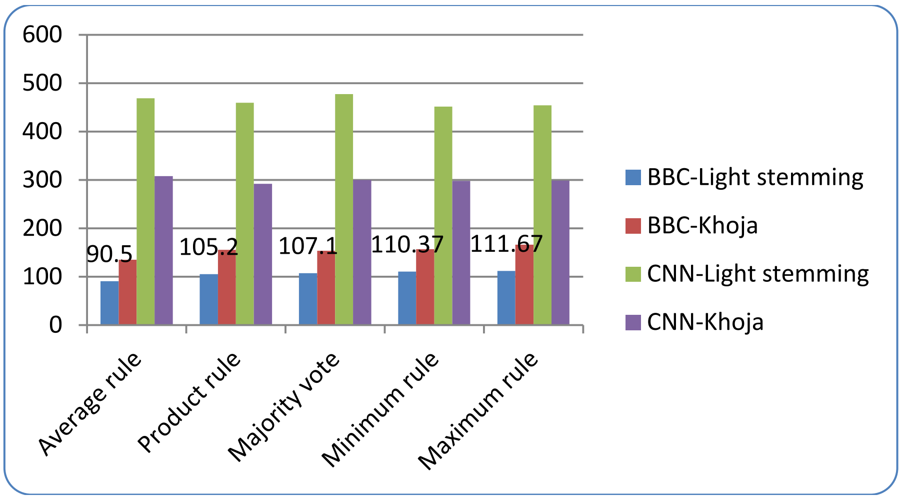

Figure 5.2: Summarization of three classifiers combined Model building time.

From Figure 5.2, building Average rule model using BBC Arabic dataset with Khoja stemmer need less time (90.5 s) than any other model, and this model gives a good accuracy (92.9%) as shown in Table 5.2.

5.2.2 Using five classifiers Model

At this model we combine five classifiers using fixed rules combination method; the classifiers used at this stage are SVM, Naive Bayes and C4.5, kNN and Decision Stump.

Table 5.5 shows the results of combining five classifiers using five fixed rules when applying light stemming on the BBC Arabic data set and using TF-IDF term weighting.

Table 5.5 presents that applying majority voting rule achieves the highest Accuracy (94.5%), Recall (0.945), Precision (0.945) and F-measure (0.943) of classification.

Table 5.5: Accuracy, F-measure and Time of combined classifiers using five classifiers and BBC dataset (TF-IDF and Light stemming).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.925	0.926	0.925	92.5	150.9
Product rule	0.911	0.912	0.911	91.3	166.5
Majority vote	0.945	0.945	0.945	94.5	158.78
Minimum rule	0.915	0.913	0.917	91.4	147.92
Maximum rule	0.861	0.860	0.862	86.2	165.73

And from Table 5.6 we notice also that the majority voting rule outperforms (94.3%) over all other fixed combination rules using Khoja stemmer.

Table 5.6: Accuracy, F-measure and Time of combined classifiers using five classifiers and BBC dataset (TF-IDF and Khoja stemmer).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.928	0.929	0.927	92.7	90.48
Product rule	0.912	0.913	0.912	91.1	90.59
Majority vote	0.943	0.944	0.943	94.3	96.19
Minimum rule	0.942	0.943	0.942	89.6	91.15
Maximum rule	0.880	0.879	0.882	88.2	93.98

Also to confirm our results using five classifiers model we use other dataset as shown in Table 5.7 which shows the result of combining five classifiers by five fixed rules when applying light stemming on the CNN Arabic data set and using TF-IDF term weighting.

Table 5.7: Accuracy, F-measure and Time of combined classifiers using five classifiers and CNN dataset (TF-IDF and Light stemming).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.930	0.930	0.930	92.9	489.34
Product rule	0.926	0.926	0.926	92.5	478.30
Majority vote	0.954	0.959	0.950	93.4	473.46
Minimum rule	0.905	0.909	0.901	90.4	463.49
Maximum rule	0.884	0.879	0.890	87.9	508.91

Table 5.8 shows the result of combining five classifiers using five fixed rules when Appling Khoja stemmer on CNN Arabic data set and using TF-IDF term weighting.

Table 5.8: Accuracy, F-measure and Time of combined classifiers using five classifiers and CNN dataset (TF-IDF and Khoja stemmer).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.921	0.922	0.921	92.1	299.47
Product rule	0.950	0.950	0.950	91.8	300.69
Majority vote	0.926	0.926	0.926	92.6	299.3
Minimum rule	0.902	0.903	0.902	90.7	300.17
Maximum rule	0.877	0.888	0.867	86.7	281.13

Figure 5.3 summarizes the accuracy results of applying 5 classifiers combined model using fixed rules, applied on two datasets with two different stemming algorithms.

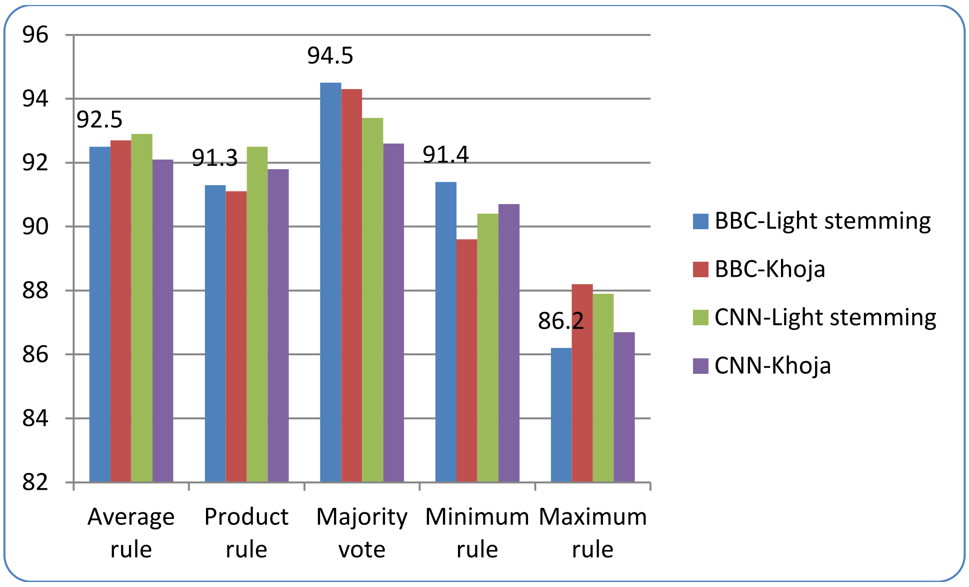

Figure 5.3: Accuracy summarization of five classifiers combined models

From Figure 5.3 we notice that the best results are when using majority vote rule applied on BBC Arabic dataset with light stemming.

Figure 5.4 summarizes the results of combining a five classifiers model in term of time using two datasets and two different stemmers, and we notice that building Average rule model using BBC Arabic dataset with Khoja stemmer need less time (90.48 s) than any other model, and this model gives a good accuracy (92.7%) as shown in Table 5.6.

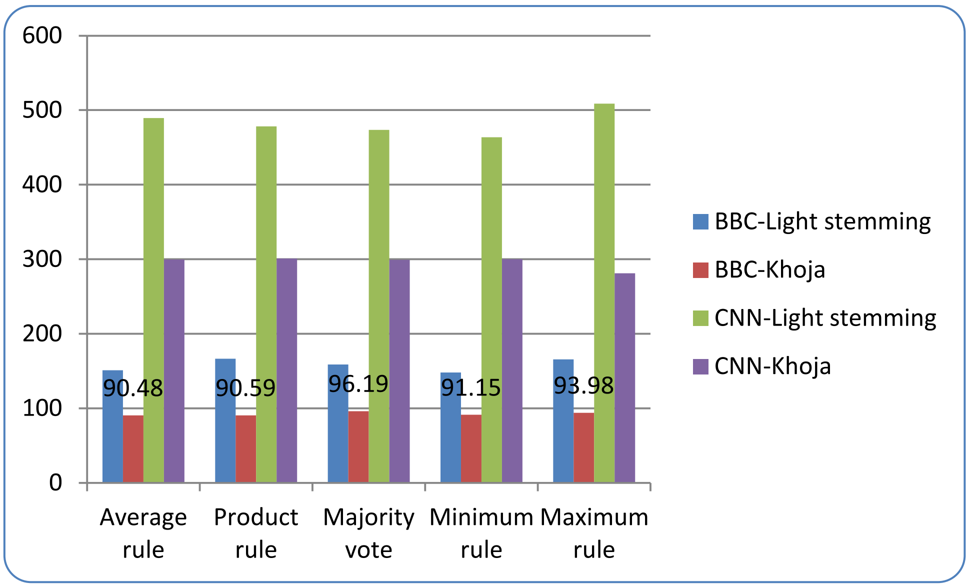

Figure 5.4: Comparison of five classifiers combined Models building time

5.2.3 Using seven classifiers Model

At this model we combine seven classifiers using fixed rules combination method; the classifiers used at this stage are SVM, Naive Bayes, C4.5, RBFN, kNN, Decision Stump and Nearest-neighbor-like

Table 5.9 shows the results of combining seven classifiers using five fixed rules when applying light stemming on the BBC Arabic data set and using TF-IDF term weighting.

Table 5.9: Accuracy, F-measure and Time of combined classifiers using seven classifiers and BBC dataset (TF-IDF and Light stemming).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.935	0.939	0.932	93.2	730.32
Product rule	0.926	0.926	0.926	92.9	787.54
Majority vote	0.954	0.955	0.953	95.3	835.94
Minimum rule	0.901	0.901	0.901	90.2	735.12
Maximum rule	0.876	0.883	0.870	87.0	741.05

We notice that from Table 5.9, the accuracy increases when applying majority voting rule on BBC Arabic dataset using seven classifiers, the highest accuracy equal to 95.3%.

Also we apply the model on BBC Arabic dataset with Khoja stemmer. Table 5.10 shows that the maximum accuracy when using fixed rules combination is given by using majority voting rule (94.6%).

Table 5.10: Accuracy, F-measure and Time of combined classifiers using seven classifiers and BBC dataset (TF-IDF and Khoja stemmer).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.920	0.920	0.921	92.0	360.57
Product rule	0.910	0.908	0.912	90.1	360.83
Majority vote	0.946	0.946	0.946	94.6	323.7
Minimum rule	0.901	0.901	0.902	89.2	360.97
Maximum rule	0.864	0.870	0.858	85.8	332.13

To confirm our results, we apply the model to CNN Arabic dataset as shown in Table 5.11 and Table 5.12.
Table 5.11 shows the result when we use CNN Arabic dataset with light stemming, and we notice that the majority voting accuracy about 93.3% which is the highest one over all other rules.

Table 5.11: Accuracy, F-measure and Time of combined classifiers using seven classifiers and CNN dataset (TF-IDF and Light stemming).

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.921	0. 922	0.920	92.0	5948.59
Product rule	0.917	0.916	0.918	91.8	6102.25
Majority vote	0.930	0.931	0.930	93.3	5945.64
Minimum rule	0.901	0.902	0.901	90.9	5891.67
Maximum rule	0.890	0.889	0.891	88.9	5982.17

And using CNN Arabic dataset with Khoja stemmer, Table 5.12 shows the results which confirm all previous experiments results in which the majority voting give 92.8% of accuracy.

Table 5.12: Accuracy , F-measure and Time of combined classifiers using seven classifiers and CNN dataset (TF-IDF and Khoja stemmer)

Rule	F-measure	Precision	Recall	Accuracy %	Time (s)
Average rule	0.919	0.919	0.919	91.9	3682.28
Product rule	0.901	0.902	0.901	90.0	3722.75
Majority vote	0.926	0.927	0.926	92.8	3602.66
Minimum rule	0.887	0.887	0.887	88.9	3589.54
Maximum rule	0.861	0.861	0.861	86.0	3584.02

Figure 5.5 summarizes the accuracy results of applying seven classifiers combined model using fixed rules, applied on two datasets with two different stemming algorithms.

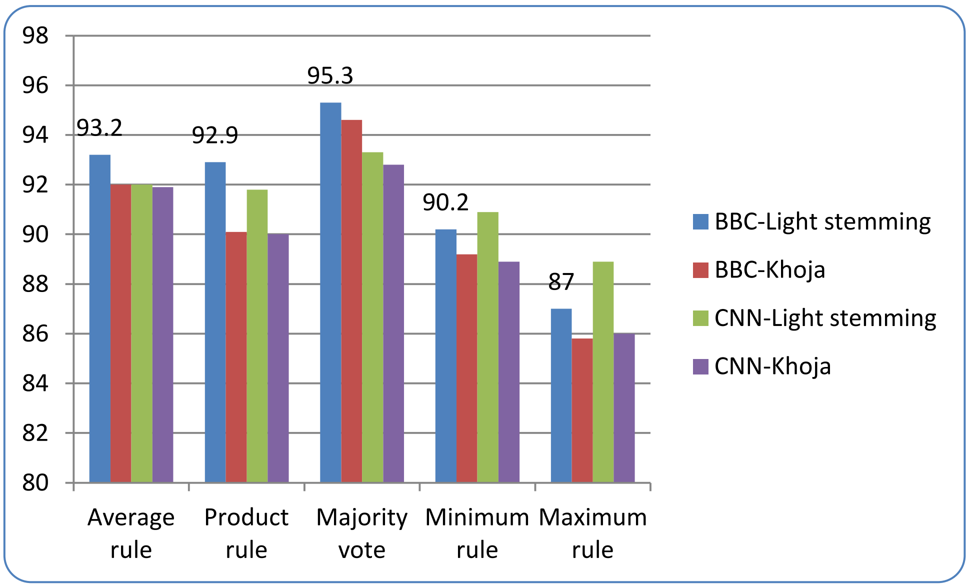

Figure 5.5: Accuracy summarization of seven classifiers combined models

From Figure 5.5 we notice that the best results are when using majority vote rule applied on BBC Arabic dataset with light stemming.

Figure 5.6 summarizes the results of combining seven classifiers model in term of time using two datasets and two different stemmers, and we notice that building majority voting model using BBC Arabic dataset with Khoja stemmer need less time

(323 s) than any other model, and this model gives a high accuracy (92.8%) as shown in Table 5.12.

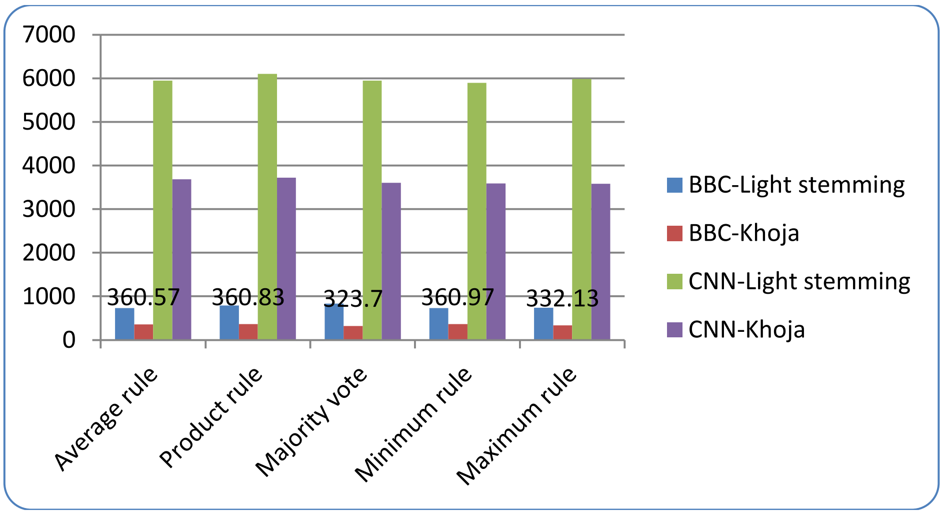

Figure 5.6: Comparison of seven classifiers combined Models building time.

5.2.3 Comparing all models

In this section we compare between the previous models, Figure 5.7 shows the accuracy of all models using three, five and seven classifiers.

From the previous tables and figures we notice that the best results are got using BBC dataset with light stemming and TF-IDF weighting, Figure 5.7 compare between these models using this dataset and stemming.

From Figure 5.7 we can conclude that the best accuracy achieved when using seven classifiers model using BBC dataset with Light stemming and TF-IDF weighting.

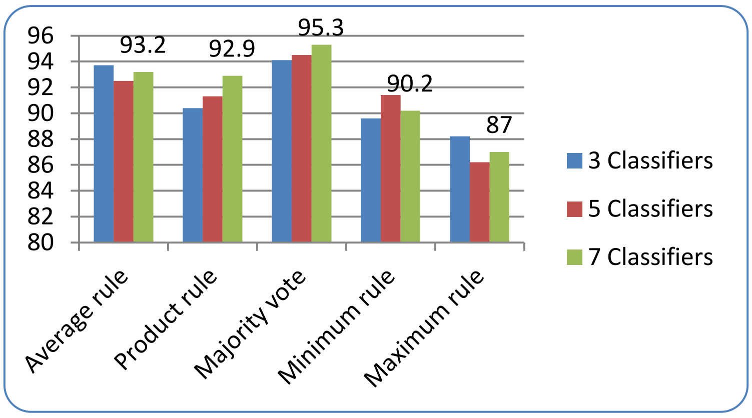

Figure 5.7: The accuracy of all models using three, five and seven classifiers.

5.3 Arabic Text Documents classification using Stacking

The second approach that we build in this thesis to improve Arabic text documents classification is stacking.

In this approach, we use two basic models, in which the difference between them is the Meta classifier. In the first model we use Naïve Bayes classifier as a Meta classifier, where in the second model we use Linear Regression prediction as a Meta classifier.

In each model, we use different number of base classifiers as we will show in the next sections. The base classifiers that we use in all models are:

1- Naïve Bayes
2- SVM
3- C4.5
4- Decision Stump
5- k-Nearest Neighbor (kNN)
6- A radial basis function network (RBFN)
7- Learning Vector Quantization (LVQ)

In each of the two models, we use different number of base classifiers. We implement the models using three and five base classifiers where as we cannot implement a stacking model with seven classifiers because it needs more high resources.
The models evaluated using two datasets BBC and CNN, with two different stemming algorithms (Light Stemming and Khoja stemmer).
The following sections show the evaluated models and the results of each model.

5.3.1 Stacking with Naïve Bayes as Meta classifier

In this section, we build combined models that based on Naïve Bayes classifier as a Meta classifier using three and five base classifiers.

The first model that we evaluate is a stacking model that consists of the following three base classifiers:

1- Naïve Bayes
2- SVM
3- C4.5

Table 5.13: Accuracy, F-measure and Time of stacked model of three (Naïve Bayes, SVM and C4.5) classifiers and Naïve Bayes Meta classifier.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.989	0.989	0.989	98.9	1480.35
BBC with Khoja stemmer	0.975	0.976	0.975	97.6	951.40
CNN with light stemming	0.924	0.924	0.924	92.4	3671.63
CNN with Khoja stemmer	0.906	0.907	0.906	90.8	3203.74
OSAC with light stemming	0.968	0.968	0.968	96.8	16433.78
OSAC with Khoja stemmer	0.956	0.956	0.957	95.7	15527.16

Three different datasets are used to confirm the results of combination as shown in Table 5.13, which shows the result of combining three classifiers using stacking; the highest accuracy is given by using BBC Arabic dataset with light stemming (98.9 %).
The second model that we evaluate is a stacking model that consists of the following three base classifiers:

1- LVQ
2- Naive Bayes
3- C4.5

Table 5.14: Accuracy, F-measure and Time of stacked model of three classifiers (LVQ, Naive Bayes and C4.5) and Naïve Bayes Meta classifier.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.982	0.983	0.982	98.3	1713.75
BBC with Khoja stemmer	0.977	0.979	0.976	97.7	1535.51
CNN with light stemming	0.960	0.962	0.959	96.0	3503.74
CNN with Khoja stemmer	0.951	0.952	0.950	95.1	3253.84

Table 5.14 shows the results of combining LVQ, Naive Bayes, and C4.5 using stacking with Naïve Bayes as a Meta classifier, the results show that stacking these classifiers gives high classification accuracy with the two datasets.
The third model that we evaluate is a stacking model that consists of the following five base classifiers:

1- SVM
2- Naive Bayes

3- C4.5
4- Decision Stump
5- kNN

Table 5.15: Accuracy, F-measure and Time of stacked model of five classifiers (SVM, Naive Bayes, C4.5, Decision Stump and kNN) and Naïve Bayes Meta classifier.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.992	0.993	0.992	99.2	1962.73
BBC with Khoja stemmer	0.983	0.985	0.981	98.9	1760.91
CNN with light stemming	0.977	0.978	0.977	97.8	3756.54
CNN with Khoja stemmer	0.965	0.966	0.964	96.4	3572.37

From Table 5.15 we notice that we get a very high accuracy using five classifiers, but also the time needed to build the model is increasing also compared to Table 5.13.

At this stage we use only two datasets because using stacking with five base classifiers needs high memory recourses, so we cannot use OSAC dataset with all stacked models that contain five classifiers.

The fourth model that we evaluate is a stacking model that consists of the following five base classifiers:

1- LVQ
2- Naive Bayes
3- C4.5
4- RBF networks
5- kNN

Table 5.16: Accuracy, F-measure and Time of stacked model of five classifiers (LVQ, Naive Bayes, C4.5, RBF networks and kNN) and Naïve Bayes Meta classifier.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.988	0.989	0.988	98.9	2163.43
BBC with Khoja stemmer	0.983	0.985	0.981	98.1	1822.65
CNN with light stemming	0.975	0.976	0.975	97.6	4196.71
CNN with Khoja stemmer	0.970	0.970	0.970	96.9	3714.42

Table 5.16 shows the results of combining these five classifiers with stacking and using Naïve Bayes as Meta classifier. Also we notice that the accuracy increases when using five bas classifiers compared to the model contains just three classifiers as in Table 5.14.

5.3.2 Stacking with Linear Regression as Meta classifier

In this section we build combined models that based on Linear Regression classifier as a Meta classifier using three and five base classifiers.

The first model that we evaluate is a stacking model that consists of the following three base classifiers with Linear Regression classifier as a Meta classifier:

1- Naïve Bayes
2- SVM
3- C4.5

Table 5.17 shows the results of combining these three classifiers with stacking and using Linear Regression as Meta classifier, and we notice this model achieves a high accuracy using BBC dataset.

Table 5.17: Accuracy, F-measure and Time of stacked model of three (Naïve Bayes, SVM and C4.5) classifiers and Linear Regression Meta classifier.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.994	0.994	0.994	99.4	1567.84
BBC with Khoja stemmer	0.977	0.977	0.977	97.7	1026.27
CNN with light stemming	0.938	0.939	0.938	93.9	3701.96
CNN with Khoja stemmer	0.922	0.922	0.922	92.2	3289.74
OSAC with light stemming	0.955	0.956	0.955	95.6	16964.25
OSAC with Khoja stemmer	0.942	0.942	0.942	94.2	15891.81

The second model that we evaluate is a stacking model that consists of the following five base classifiers with Linear Regression classifier as a Meta classifier:

1- Naïve Bayes
2- SVM
3- C4.5

4- Decision Stump
5- kNN

Table 5.18 shows the results of combining five classifiers with stacking and using Linear Regression as Meta classifier; we notice the accuracy increases when using five classifiers where as the time is also increases.

Table 5.18: Accuracy, F-measure and Time of stacked model of five classifiers (SVM, Naive Bayes, C4.5, Decision Stump and kNN) and Linear Regression Meta classifier.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.994	0.995	0.994	99.5	3718.07
BBC with Khoja stemmer	0.980	0.980	0.980	98.0	3291.12
CNN with light stemming	0.940	0.941	0.940	94.0	8721.59
CNN with Khoja stemmer	0.929	0.929	0.929	93.0	8036.61

5.4 Arabic Text Documents classification using boosting

At this experiment we build a model that use AdaBoost to classify Arabic text documents. The model build based on C4.5 classifier.

Table 5.19 shows the results of using AdaBoost with C4.5 with 5 iterations; we notice that we get the highest accuracy (95.3%) using BBC Arabic dataset with light stemming.

Table 5.19: Accuracy, F-measure and Time of using AdaBoost with C4.5 classifier using 5 iterations.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.952	0.953	0.952	95.3	1174.58
BBC with Khoja stemmer	0.941	0.941	0.942	94.2	922.37
CNN with light stemming	0.924	0.924	0.924	92.6	3543.64
CNN with Khoja stemmer	0.901	0.901	0.901	90.1	3328.93

From Table 5.19 and Table 5.20 we see that increasing the number of iteration produces a high classification accuracy using all datasets.

Table 5.20: Accuracy, F-measure and Time of using AdaBoost with C4.5 using 10 iterations.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.995	0.995	0.995	99.5	1965.72
BBC with Khoja stemmer	0.980	0.980	0.980	98.0	1585.29
CNN with light stemming	0.942	0.942	0.943	94.3	4877.76
CNN with Khoja stemmer	0.938	0.938	0.939	93.9	4398.46

5.5 Arabic Text Documents classification using Bagging

At this experiment we build a model that use bagging to classify Arabic text documents. The model build based on Decision Tree classifier.

At the first experiment we use bagging with decision Tree with 5 iterations as shown in Table 5.21, we notice that the highest accuracy is got when using BBC dataset with light stemming.

Table 5.21: Accuracy, F-measure and Time of using Bagging with Decision Tree using 5 iterations.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.936	0.936	0.936	93.7	295.85
BBC with Khoja stemmer	0.930	0.931	0.930	93.0	201.17
CNN with light stemming	0.911	0.912	0.911	91.1	922.16
CNN with Khoja stemmer	0.906	0.906	0.906	90.6	738.88

We use the same experiment but with 10 iterations as shown in Table 5.22, we notice that we get a higher accuracy when increasing the number of iterations.

Table 5.22: Accuracy, F-measure and Time of using Bagging with Decision Tree using 10 iterations.

Dataset	F-measure	Precision	Recall	Accuracy %	Time (s)
BBC with light stemming	0.993	0.993	0.993	99.3	470.99
BBC with Khoja stemmer	0.977	0.977	0.977	97.7	365.62
CNN with light stemming	0.928	0.929	0.928	92.9	1427.54
CNN with Khoja stemmer	0.913	0.913	0.913	91.3	1132.43

5.5 Comparing Combined models with single classifiers

In this section we compare the classification accuracy of all combined models that we build with the classification accuracy of single classifiers that other researchers used to classify Arabic text documents

5.5.1 Comparing combined classifiers using fixed combining rules with single classifiers

According to the results that we get from combining different classifiers using fixed combining rules in section 5.1 , we compare these results with each single classifier that used by other researchers to classify Arabic text document.

Figure 5.8 shows the accuracy of combining three classifiers using majority voting rule and BBC Arabic dataset with light stemming with other classifiers.

From Figure 5.8 we notice that the combined model of three classifiers (Naïve Bayes, SVM and C4.5) give a high accuracy (94.1%) compared to the results of single classifiers used in [22, 24, 27].

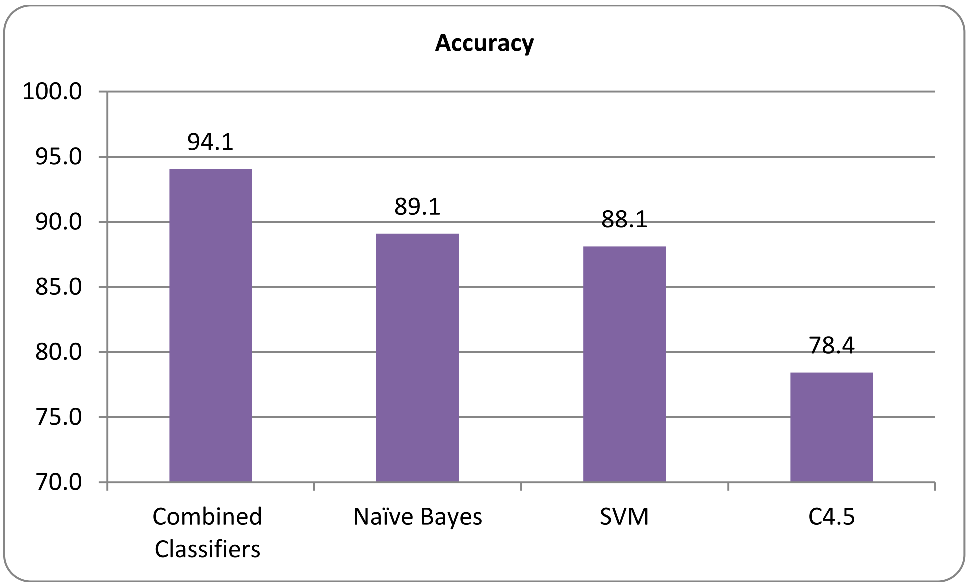

Figure 5.8: **A comparison between three combined classifiers using majority voting rule vs. single classifiers.**

Figure 5.9 shows the accuracy of combining five classifiers using majority voting rule and BBC Arabic dataset with light stemming with other classifiers.

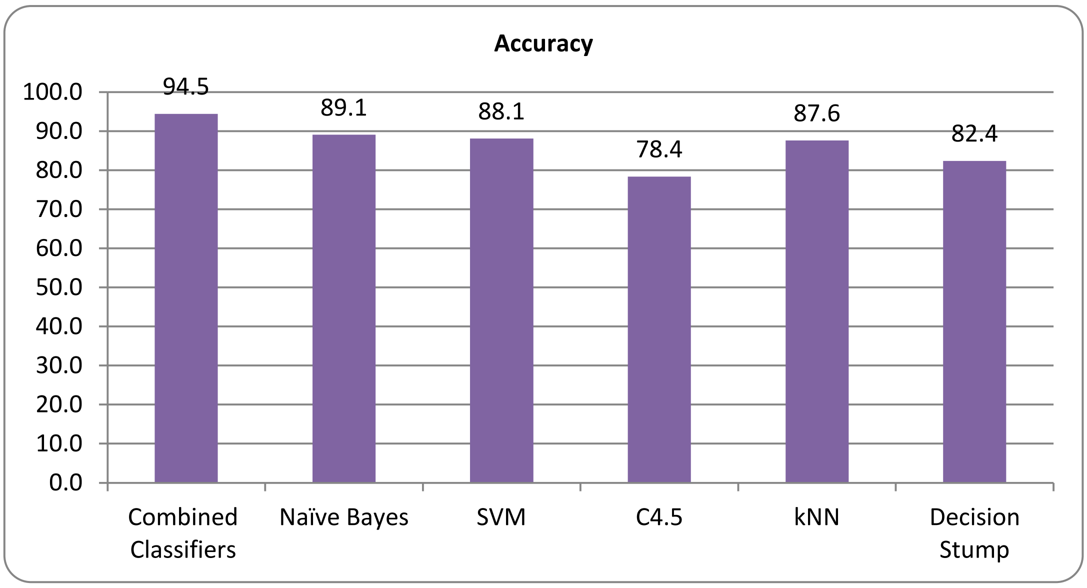

Figure 5.9: A comparison between five combined classifiers using majority voting rule vs. single classifiers.

From Figure 5.9 we notice that the combined model of five classifiers (Naïve Bayes, SVM, C4.5, kNN and Decision Stump) gives a high accuracy (94.5%) compared to the results of single classifiers used in [11, 22, 24, 27].

Figure 5.10 shows the accuracy of combining seven classifiers using majority voting rule and BBC Arabic dataset with light stemming with other classifiers.
The comparison shows that the combined model using seven classifiers (Naïve Bayes, SVM, C4.5, kNN, RBFN, Nearest-neighbor-like and Decision Stump) an majority voting rule give a accuracy higher than any other single classifier used in the model [11, 22, 24, 27].

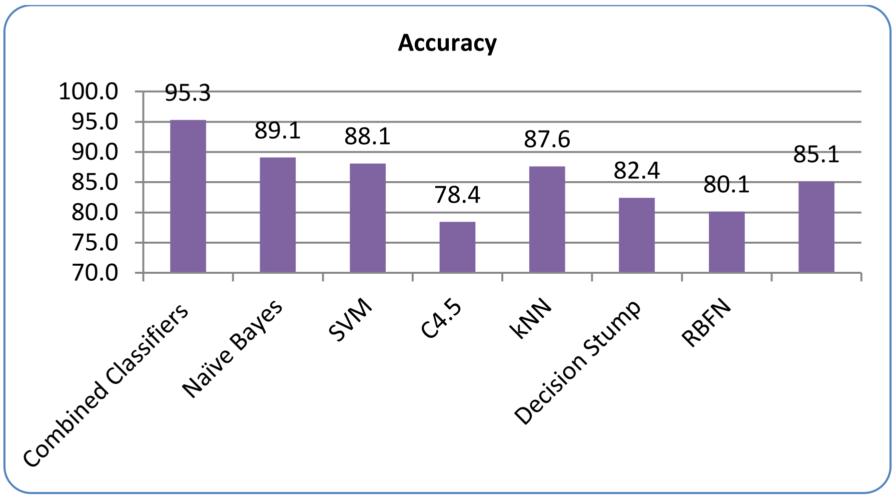

Figure 5.10: A comparison between seven combined classifiers using majority voting rule vs. single classifiers.

5.5.2 Comparing combined classifiers using stacking with single classifiers

In this section we compare the results of combined classifiers using stacking with other single classifiers that researchers use to classify Arabic text document.

First we will compare stacked model with three and five classifiers using naïve Bayes as a Meta classifier.

Figure 5.11 compare between stacked model using three classifiers (Naïve Bayes, SVM and C4.5) using Naïve Bayes as a Meta classifier and single classifiers , we see that the accuracy of stacked model (98.3%) is higher than any single classifier.

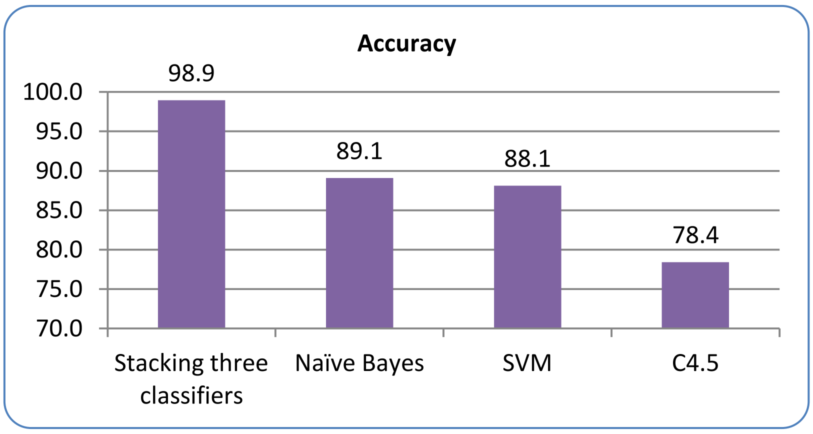

Figure 5.11: A comparison between stacking using three classifiers vs. single classifiers.

In Figure 5.12 we use other three classifiers (LVQ, Naïve Bayes and C4.5); the comparison shows that stacked model outperforms other single classifiers [22, 24, 40].

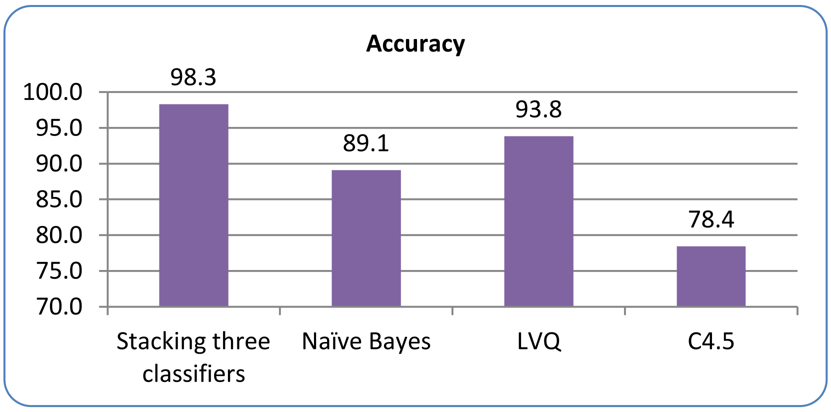

Figure 5.12: A comparison between stacking using three classifiers vs. single classifiers.

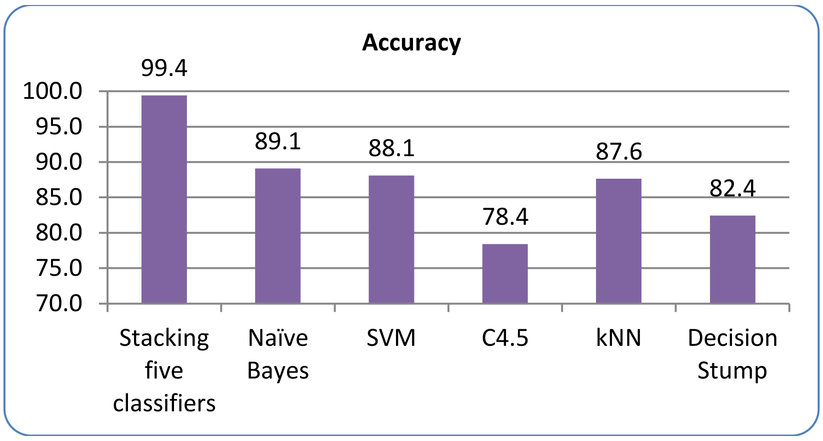

Figure 5.13: **A comparison between stacking using five classifiers vs. single classifiers.**

The other comparison is done between the stacked models built using five classifiers, the first model consists of Naïve Bayes, kNN , SVM, Decision Stump and C4.5 using Naïve Bayes as a Meta classifier and with each single classifier used to classify Arabic text documents , the result shows the high accuracy of stacked model as shown in Figure 5.13.

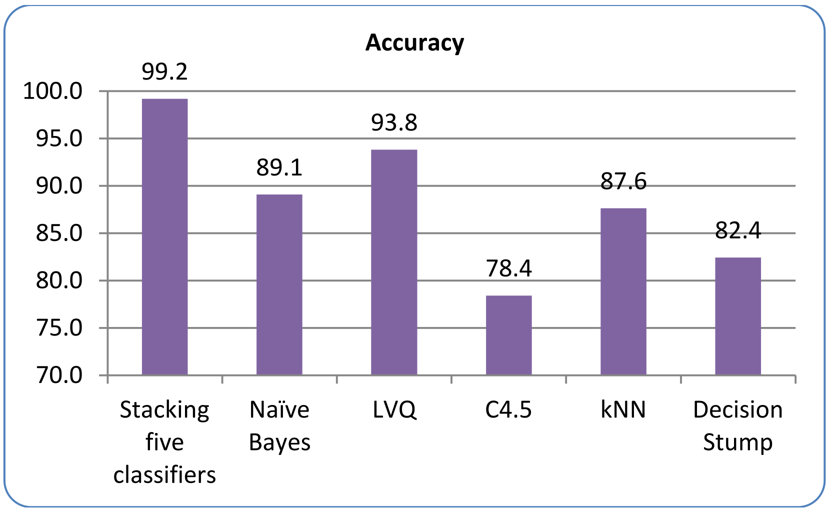

Figure 5.14: **A comparison between stacking using five classifiers vs. single classifiers**

Figure 5.14 shows other model that consists of Naïve Bayes, kNN, LVQ, Decision Stump and C4.5; comparing this model to the accuracy of single classifiers we get that stacking outperforms all single classifiers which used in [11, 22, 24, 27].

The last stacked model that we will compare is the same as previous model but when using Linear Regression as a Meta classifier.

Figure 5.15 show the comparison between stacked models using Linear Regression as a Meta classifier and other single classifiers, the result shows that using five stacked classifiers to classify Arabic text documents give a very high accuracy compared to single classifiers in [11, 22, 24, 27, 40].

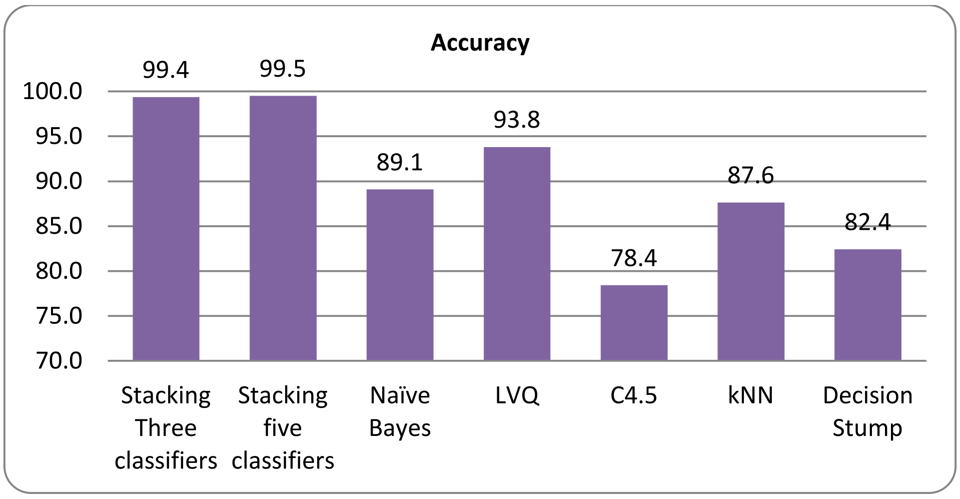

Figure 5.15: A comparison between stacking using three and five classifiers vs. single classifiers.

5.5.3 Comparing AdaBoost with single classifier

In this section we compare the results of using AdaBoost a single classifier that researchers use to classify Arabic text document.

We use C4.5 classifier with AdaBoost, first we implement boosting using 5 iterations and then using 10 iterations; Figure 5.16 shows the result of boosting C4.5.

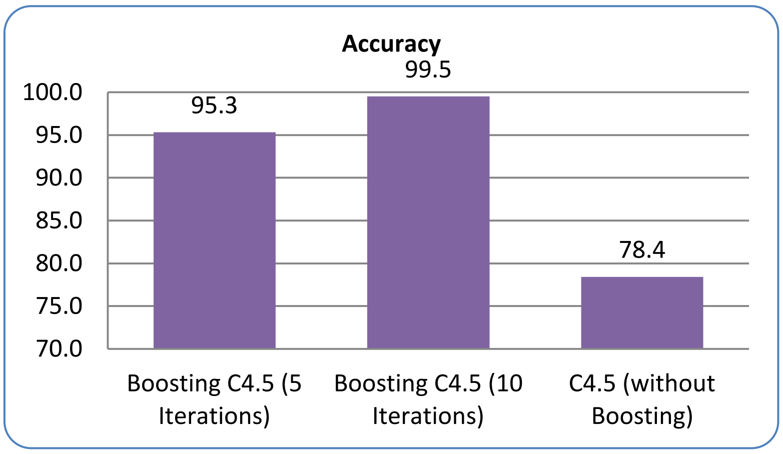

Figure 5.16: A comparison between AdaBoost vs. single classifier (C4.5) using 5 and 10 iterations.

From Figure 5.16 we see that using C4.5 as a single classifier used in [11] achieved a classification accuracy 78.42%, which is low compared to using the same classifier with boosting which improve the accuracy of classifying Arabic text documents to 99.5% using 10 iterations.

5.5.4 Comparing Bagging with single classifier

In this section we compare the results of using bagging a single classifier that researchers use to classify Arabic text document.

We use Decision Tree classifier with bagging, first we implement bagging using 5 iterations and then using 10 iterations; Figure 5.17 shows the result of bagging Decision Tree.

We notice that applying bagging on Decision Tree classifier improves effectively the accuracy of classifying Arabic text documents compared to using the Decision Tree as a single classifier such in [70].

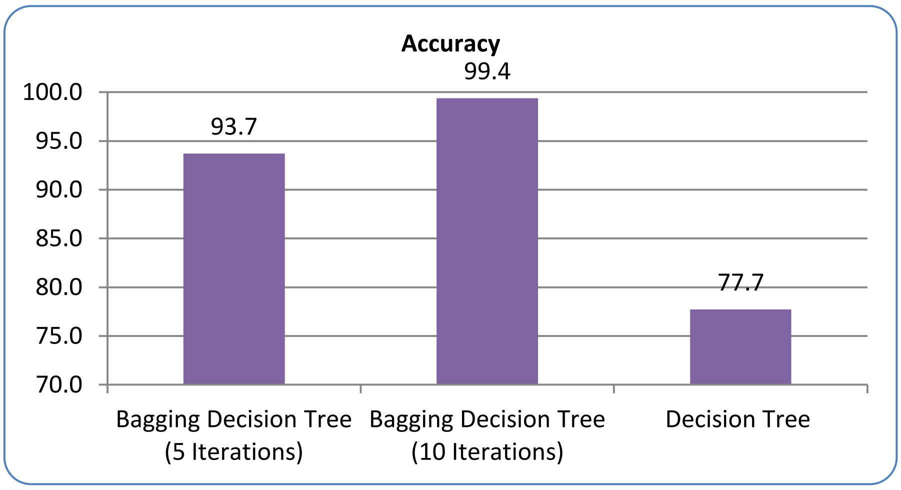

Figure 5.17: A comparison between Bagging vs. single classifier (Decision Tree) using 5 and 10 iterations.

5.6 Discussion

In this chapter we implement different approaches to combine classifiers, fixed combining rules, stacking, boosting and Bagging are used to improve the accuracy of classifying Arabic text document. Two stemmers are applied on our datasets, where we use only TF-IDF term weighting schema.

All combined approaches used in this thesis achieve a high accuracy in classifying Arabic text documents. The best results obtained using BBC Arabic dataset with light stemming and TF-IDF weighting schema. The time needed to build the models depends on the combination algorithm, we notice that stacking needs more time than any other approach. Also the number of classifiers used in the model increase the time needed to build the model.

The fixed combination rules need less time to build the model than any other combination approach, because this approach simply called non-trainable combiner, which means the combiners do not need to be trained. Each classifier in the model gives a decision and the combiner uses the selected rule to give the final decision.

The Stacking algorithm needs more time than any other combination approach because it consists of two levels of classifiers. The first level consists of base classifiers which are trained using the documents dataset, and then the outputs of the base classifiers are used with the original dataset to produce a new dataset. 70 % of the dataset is used to train the base classifiers, and 30% for testing. The second level or Meta classifier is trained using the new dataset by using 10 folds cross validation. The 10 folds cross validation training method needs more time than the method used with the base classifiers. All mentioned reasons makes the stacking algorithm need more time to be built.

The AdaBoost algorithm also achieves a very high accuracy in classification. The AdaBoost algorithm focuses on the unclassified documents or the misclassified during building the model. It assigns weights and focuses on these documents through the next iterations to improve the classification accuracy. The AdaBoost algorithm needs an acceptable time to build the model compared with stacking model.

The last algorithm is bagging which achieves a high accuracy in classifying Arabic text documents. We notice that bagging needs less time than AdaBoost algorithm to build the model because bagging trains different models in the same time and combines their decisions, see section (3.3.5).

Table 5.23 shows a comparison between fixed combination rules and stacking when using the BBC Arabic dataset.

From Figure 5.23 we notice that the accuracy of using three classifiers with fixed combining rules was 94.1%, and the needed time to build the approach was 107.1 seconds. When we increased the number of classifiers to seven, the accuracy increased to 95.3%, but in the other side the time needed to build the approach was increased to 835.94 seconds. That means increasing the accuracy of results by 1.2% needs additional 728.84 seconds.

The accuracy of the stacking algorithm was 98.9%, where the time needed to build the approach was 1480.35 seconds when using three classifiers. But when we used five classifiers, the accuracy was 99.2% and the time needed to build the approach was 1963.73 seconds. Comparing the results of stacked approach using three classifiers with the approach which was built by three classifiers using fixed combining rules, we notice that the accuracy of the stacking algorithm is higher by 4.8%, but the time cost was very high because the stacking algorithm needed 1373.25 seconds more than the fixed combining rules combiner.

The high cost of time that the stacking algorithm needed to build the approach was due to the two levels of learning and the 10 fold cross validation learning method used by level-1 Meta classifier, where in the other side the fixed combining rules did not need to learn the combiner.

Table 5.23: Comparing the Accuracy and Time between Fixed combining rules and Stacking.

Number of classifiers	**Fixed Combining Rules**		**Stacking**	
	Accuracy (%)	**Time(s)**	**Accuracy (%)**	**Time(s)**
3 classifiers	94.1	107.1	98.9	1480.35
5 classifiers	94.5	158.78	99.2	1962.73
7 classifiers	95.3	835.94	-	-

Table 5.24 shows a comparison between AdaBoost and Bagging algorithms. We notice that the AdaBoost algorithm needs more time than bagging to build the approach. For example using ten iterations, the accuracy of AdaBoost was 99.5% and it needed 1174.58 seconds to classify documents, where the accuracy of bagging algorithm was 99.3% and it needed 470.99 second to classify documents.

The high cost of time that the AdaBoost needed to build the approach was due to it works iteratively, while the bagging algorithm works in parallel, which means that in the bagging algorithm, each classifier learn using a sample of dataset at the same time.

Table 5.24: Comparing the Accuracy and Time between AdaBoost and Bagging.

Number of Iterations	**AdaBoost**		**Bagging**	
	Accuracy	**Time(s)**	**Accuracy**	**Time(s)**
5	95.3	1174.58	93.7	295.85
10	99.5	1965.72	99.3	470.99

Based upon the experimental results, we have demonstrated that combining classifiers using different approaches can effectively improve the accuracy of classifying Arabic Text documents.

CHAPTER 6

CONCLUSIONS

6.1 Summary and Concluding Remarks

Many different classifiers were used to classify Arabic text documents; some of these classifiers gave high classification accuracy. In this thesis, we combined different approaches to enhance Arabic text documents classification.

The advantages of combining classifiers motivated us to combine classifiers to improve the accuracy of classifying Arabic text documents.

In this thesis, we built four combined models; the first model used fixed combining rules to combine the results of different classifiers, where the second one is stacking which used two stages of classification, in the first stage it used base classifiers, where in the second one it learnt a meta classifier based on the results of base classifiers to give the final classification result.

The third and the fourth models that we built are AdaBoost and Bagging respectively, where we used different number of iterations for each one.

The results of combining classifiers using fixed combining rules showed that the majority voting rule outperformed all single classifiers that were used to build the model and it also outperformed all other fixed combining rules such as average of probability, median of probability and other fixed rules. The highest accuracy of combining classifiers using fixed rules was achieved by majority voting using BBC Arabic data set with light stemming and using TF-IDF term weighting schema, the accuracy of the model using seven classifiers is 95.3% which was high compared to using single classifier. The time required to build this model was 835.94 second which was relatively acceptable compared to some single classifiers such as Decision Tree.

We used different classifiers at this model, the results showed that the accuracy increased when we increased the number of classifiers, but at the same time, the required time to build the model increased also. The accuracy using a model with three classifiers was 94.1%, where it was 94.5% when using five classifiers, but the best accuracy achieved using seven classifiers and it was 95.3%.

The second model that we built was a stacking model; the accuracy of stacked model was very high compared to the accuracy of single classifiers, but it needed more time to build the model because stacking needed two stages to learn the model. The first stage was to learn the base classifiers; where the second stage was to learn the Meta classifier based on the original dataset and the classification results of the base classifiers.

We used Naïve Bayes and Linear Regression as Meta classifiers to build our stacked model, where we used three and five base classifiers for each model.

The best results achieved using Naïve Bayes Meta classifier when using five base classifiers was 99.2%; where the best accuracy when using Linear Regression with five base classifiers was 99.4%. All these results achieved using BBC Arabic dataset with light stemming and TF-IDF term weighting schema.

The third model we built using AdaBoost, the AdaBoost used in conjunction with C4.5 classifier, the AdaBoost improved the performance of C4.5 classifier to classify Arabic text documents, when boosting the C4.5 using 5 iterations it classified Arabic documents and achieved a 95.3% of classification accuracy, where when we used 10 iterations the accuracy was 99.5%.

The last model we built using bagging in conjunction with Decision Tree, the model achieved a high accuracy and improved the results of decision tree classifier , the results showed that using 5 iterations achieved an accuracy of 93.7% where when we used 10 iterations we achieved 99.4% of classification accuracy.

In all of previous models, the highest accuracy achieved using BBC Arabic dataset with light stemming and TF-IDF term weighting schema.

In our experiments we used three datasets BBC Arabic, CCN Arabic and OSAC datasets. We used just two stemming methods, the light stemming and Khoja stemmer and we used only TF-IDF term weighting schema.

The combined models were compared to other single classifiers that used by researchers to classify Arabic text documents, the comparison was done in term of accuracy, precision, recall and F-measure.

The first limitation in our experiments that we cannot use the OSAC data set with all models because the OSAC data set did not fit into memory specially with stacking , AdaBoost and Bagging because they needed a high memory resources.

The second limitation was that we cannot build a stacking model that consisted of more than five classifiers because of memory resources that were needed, and at the same time increasing the number of base classifiers produced a model that needed a very long time to be built.

The combined models that we built in our research improved the accuracy of classifying Arabic text documents, all models achieved a high classification accuracy compared to the single classifiers used by other researchers, although some models such as stacking needed more time to be built, but it achieved a very high accuracy.

Fixed combining rules, AdaBoost and Bagging achieved a high accuracy and needed an acceptable time to build the model compared to some classification algorithms.

6.2 Recommendations and Future Work

In this thesis, we have shown that combining classifiers improved the accuracy of classifying Arabic text documents. Different combination methods were evaluated and all of these approaches achieved high classification accuracy.

Using fixed combination rules achieved a high accuracy with an acceptable time compared to other classifiers, where stacking achieved a very high accuracy but it needed a relatively more time to train the system.

Using boosting we showed that weak classifiers such as C4.5 can achieve high classification accuracy. The same high accuracy was achieved using bagging with decision tree classifiers.

According to the results of experiments and the limitations that we faced in our thesis, the future work will be devoted to the following points:

1- Using other classifiers to build a combined model by fixed rules to enhance the accuracy more than the achieved results.
2- Reducing the time needed to build a combined model especially for stacked models.
3- Adopting our models to deal with large datasets specially when using a large number of classifiers.
4- Working with other data types such as images and voice.

REFERENCES

[1] T. David and D. Robert, "Experiments with Classifier Combining Rules," in *Proceedings of the First International Workshop on Multiple Classifier Systems,* Cagliari, Italy, 2000.

[2] T. Dietterich, "Ensemble Methods in Machine Learning," in *Proceedings of the First International Workshop on Multiple Classifier Systems*, London, UK, 2000.

[3] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 27, no. 6, p. 942–956, 2005.

[4] M. Ponti, "Combining Classifiers: From the Creation of Ensembles to the Decision Fusion," in *4th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials,* Sao Carlos, Brazil, 2011.

[5] K. L., B. J. and D. R., "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern Recognition,* vol. 24, no. 2, p. 299–314, 2001.

[6] R. Lior, Pattern Classification Using Ensemble Methods, New Jersey: World Scientific Publishing Co. Pte. Ltd., 2010.

[7] J. Kittler, M. Hatef, R. Duin and J. Matas, "On combining classifiers," *IEEE Trans Pattern Analysis and Machine Intelligence,* vol. 20, no. 3, p. 226–239, 1998.

[8] D. Wolpert, "Stacked Generalization," *Neural Networks,* vol. 5, no. 2, pp. 241-259, 1992.

[9] J. Quinlan, "Bagging, Boosting, and C4.5," in *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[10] B. Leo, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.

[11] S. Al-Harbi, A. Almuhareb, A. Al-Thubaity, M. Khorsheed and A. Al-Rajeh, "Automatic Arabic Text Classification," in *Proceedings of The 9th International Conference on the Statistical Analysis of Textual Data*, Lyon-, France, 2008.

[12] R. Duwairi, "Arabic text Categorization," *The International Arab Journal of Information Technology,* vol. 4, no. 2, pp. 125-132, 2007.

[13] M. Saad and W. Ashour, "Arabic Text Classification Using Decision Trees," in *Computer science and information technologies*, Moscow,Russia, 2010.

[14] M. Saad, "The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification," The Islamic University, Gaza, 2010.

[15] A. El-Halees, "A Comparative Study on Arabic Text Classification," *Egyptian Computer*

Science Journal, vol. 20, no. 2, 2008.

[16] G. Kanaan, R. Al-Shalabi, S. Ghwanmeh and H. Al-Ma'adeed, "A comparison of text classification techniques applied to Arabic text," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 9, p. 1836 – 1844, 2009.

[17] Y. Bi, D. Bell, H. Wang, G. Guo and J. Juan, "Combining Multiple Classifiers Using Dempster’s Rule of Combination for Text Categorization," *Applied Artificial Intelligence*, vol. 21, no. 3, pp. 211-239, 2007.

[18] M. Cory, "Classifier Ensembles: A Practical Overview," 20 April 2005. [Online]. [Accessed March 2013].

[19] A. El-Halees, "Arabic Opinion Mining Using Combined Classification Approach," in *Proceedings of The International Arab Conference On Information Technology*, Azrqa, Jordan, 2011.

[20] R. Miguel and S. Padmini, "Combining Machine Learning and Hierarchical Indexing Structures for Text Categorization," PhD. Thesis. December, The University of Iowa , Iowa City, 2001.

[21] D. B. Durga and G. Venu, "Text Categorization and Machine Learning Methods:Current State of the Art," *Global Journal of Computer Science and Technology*, vol. 12, no. 11, 2012.

[22] A. Mesleh, "Chi Square Feature Extraction Based Svms Arabic Language Text Categorization System," *Journal of Computer Science*, vol. 3, no. 6, pp. 430-435, 2007.

[23] F. Harrag and E. El-Qawasmeh, "Neural Network for Arabic text classification," in *The 2nd International Conference of Applications of Digital Information and Web Technologies*, London, 2009.

[24] M. El-Kourdi, A. Bensaid and T. Rachidi, "Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm," in *The 20th International Conference on Computational Linguistics*, Geneva, 2004.

[25] A. El-Halees, "Arabic Text Classification Using Maximum Entropy," *The Islamic University Journal*, vol. 15, no. 1, pp. 157-167, 2007.

[26] H. Sawaf, J. Zaplo and H. Ney, "Statistical Classification Methods for Arabic News Articles," in *In the Workshop on Arabic Natural Language Processing*, Toulouse, France, 2001.

[27] R. Al-Shalabi, G. Kanaan and M. Gharaibeh, "Arabic text categorization using kNN algorithm," in *Proceedings of the 4th International Multiconference on Computer Science and Information Technology*, Amman, Jordan, 2006.

[28] I. Hmeidi, B. Hawashin and E. El-Qawasmeh, "Performance of KNN and SVM classifiers on full word Arabic articles," *Advanced Engineering Informatics*, vol. 22, no.

1, p. 106–111, 2008.

[29] M. Abbas, K. Smaili and D. Berkani, "Comparing TR-Classifier and kNN by using Reduced Sizes of Vocabularies," in *The 3rd International Conference on Arabic Language Processing*, Rabat, Morroco, 2009.

[30] M. Bawaneh, M. Alkoffash and A. Al-Rabea, "Arabic Text Classification using K-NN and Naive Bayes," *Journal of Computer Science,* vol. 4, no. 7, pp. 600-605, 2008.

[31] A. Danesh, B. Moshiri and O. Fatemi, "Improve text classification accuracy based on classifier fusion methods," in *Proceedings of The 10th International Conference on Information Fusion*, Quebec, Canada, 2007.

[32] A. Fujino, H. Isozaki and J. Suzuki, "Multi-label Text Categorization with Model Combination based on F1-score Maximization," in *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, Kyoto, Japan, 2008.

[33] T. M. Mitchell, Machine learning, New York, US: McGraw Hill, 1996.

[34] T. Sergios and K. Konstantinos, Pattern Recognition, Burlington,USA: Elsevier Inc, 2009.

[35] S. Fabrizio, N. Consiglio and R. delle, "Machine learning in automated text categorization," *ACM Computing Surveys (CSUR) journal* , vol. 34, no. 1, pp. 1-47 , 2002 .

[36] T. Gharib, M. Habib and Z. Fayed, "Arabic Text Classification Using Support Vector Machines," *International Journal of Computers and Their Applications,* vol. 16, no. 4, pp. 192-199, 2009.

[37] A. Yottamine, "Analytics: The Machine Learning Advantage," Analytics: The Machine Learning Advantage, 2011. [Online]. Available: http://yottamine.com/machine-learning-svm. [Accessed September 2013].

[38] T. Kohonen, Self-organization and associative memory, New York: Springer-Verlag, 1995.

[39] M. Martín-Valdivia, L. Ureña-López and G.-V. M., "The learning vector quantization algorithm applied to automatic text classification tasks," *Neural Networks,* vol. 20, no. 6, pp. 748-756, 2007.

[40] M. Azara, T. Fatayer and A. El-Halees, "Arabie text classification using Learning Vector Quantization," in *8th International Conference on Informatics and Systems (INFOS)*, Giza, Egypt, 2012.

[41] K. Khalifa, M. Bedoui, M. Dogui and F. Alexandre, "Alertness States Classification By SOM and LVQ Neural Networks," in *Proceedings Of World Academy Of Science, Engineering And Technology*, 2005.

[42] M. Andrew and N. Kamal, "A comparison of event models for naive bayes text classification," in *In AAAI-98 Workshop on Learning for Text Categorization*, Madison, Wisconsin, 1998.

[43] J. Thorsten, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," in *In proceedings of the 14th International Conference on Machine Learning* , Nashville,USA, 1997.

[44] Y. Yiming and L. Xin, "A Re-Examination of Text Categorization Methods," in *In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1999.

[45] M. B. Habib, "An intelligent system for automated arabic text categorization," Ain shams University,Electrical Engineering, Mathematics and Computer Science, Cairo, Egypt, 2008.

[46] O. Yen-Jen, H. Shien-Ching, O. Yu-Yen and C. Chien-Yu, "Data classification with radial basis function networks based on a novel kernel density estimation algorithm," *IEEE Transactions on Neural Networks,* vol. 16, no. 1, 2005.

[47] Wikipedia, "Radial basis function network," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Radial_basis_function_network.

[48] L. Felföldi, "Classifier Combination Systems and their Application in Human Language Technology," Research Group on Artificial Intelligence, Szeged,Hungary, 2008.

[49] L. Kuncheva, "'Fuzzy' vs `Non-fuzzy' in combining classifiers designed by boosting," *IEEE Transactions on Fuzzy Systems,* vol. 11, no. 6, pp. 729-741, Dec 2003.

[50] D. Saso, Bernard and Ženko, "Is Combining Classifiers with Stacking Better than Selecting the Best One?," *Machine Learning Journal,* vol. 54, no. 3, pp. 255 - 273, 2004.

[51] Y. Freund, "Boosting a Weak Learning Algorithm by Majority," *Information and Computation,* vol. 121, no. 2, p. 256–285, 1995.

[52] M. Saad, "Arabic Computational Linguistics," 26 07 2010. [Online]. Available: http://sourceforge.net/projects/ar-text-mining/. [Accessed 23 04 2013].

[53] A. Aliwy, "Tokenization as Preprocessing for Arabic Tagging System," *International Journal of Information and Education Technology,* vol. 2, no. 4, pp. 348-353, 2012.

[54] A. Fahad, A. Ibrahim and F. Salah, "Processing Large Arabic Text Corpora: Preliminary Analysis and Results," in *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 2009.

[55] M. Attia, "Arabic Tokenization System," in *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, Prague, Czech Republic, 2007.

[56] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, New York: Cambridge University Press, 2008.

[57] M. R. Al-Maimani, A. Naamany and A. Z. A. Bakar, "Arabic information retrieval: techniques, tools and challenges," in *GCC Conference and Exhibition*, 2011.

[58] A. Hayder, A. Shaikha, A. Amna, A. Khadija, A. Naila, A. Noura and A. and Shaikha, "Arabic Light Stemmer: A new Enhanced Approach," in *The Second International Conference on Innovations in Information Technology (IIT'05)*, Dubai, 2005.

[59] C. Aitao, "“Building an Arabic Stemmer for Information Retrieval," in *Proceedings of the Eleventh Text Retrieval Conference*, Berkeley, 2003.

[60] M. Ababneh, R. Al-Shalabi, G. Kanaan and A. Al-Nobani, "Building an Effective Rule-Based Light Stemmer for Arabic Language to Improve Search Effectiveness," *The International Arab Journal of Information Technology*, vol. 9, no. 4, pp. 368-372, 2012.

[61] S. Khoja and R. Garside, "Stemming Arabic Text," in *Lancaster, UK, Computing Department, Lancaster University*, 1999.

[62] N. Abdusalam, S. Tahaghoghi and S. Falk, "Stemming Arabic Conjunctions and Prepositions," in *Proceedings of the 12th international conference on String Processing and Information Retrieval*, Heidelberg, 2005.

[63] L. Leah, B. Lisa and C. Margaret, "Light Stemming for Arabic Information Retrieval," *Arabic Computational Morphology Text, Speech and Language Technology*, vol. 38, pp. 221-243, 2007.

[64] Q. Zhengwei, G. Cathal, D. Aiden and S. Alan, "Term weighting approaches for mining significant locations from personal location logs," in *CIT 2010 - 10th IEEE International Conference on Computer and Information Technology*, Bradford, UK, 2010.

[65] L. Man, T. Chew-Lim, L. Hwee-Boon and S. Sam-Yuan, "A comprehensive comparative study on term weighting schemes for text categorization with support vector machines," in *WWW '05 Special interest tracks and posters of the 14th international conference on World Wide Web*, Chiba, Japan, 2005.

[66] N. Nanas, V. Uren, A. Roeck and J. Domingue, "A Comparative Study of Term Weighting Methods for Information Filtering," KMi-TR-128. Knowledge Media Institue, The Open University, 2003.

[67] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries," in *First International Conference on. Machine Learning*, New Brunswick:NJ, USA, 2003.

[68] "Weka 3: Data Mining Software in Java," Machine Learning Group at the University of Waikato, [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[69] "RapidMiner," [Online]. Available: http://rapid-i.com/.

[70] F. Harrag, E. El-Qawasmeh and P. Pichappan, "Improving arabic text categorization using decision trees," in *First International Conference on Networked Digital Technologies*, Ostrava, 2009.

[71] A. Charu and Z. ChengXiang, "A Survey of Text Classification Algorithms," in *Mining Text Data*, Springer US, 2012, pp. 163-222.

[72] R. Duin, "The combining classifier: to train or not to train?," in *Proceedings of 16th International Conference on Pattern Recognition*, Netherlands, 2002.

[73] L. Shoushan, Z. Chengqing and W. Xia, "Sentiment Classification through Combining Classifiers with Multiple Feature Sets," in *International Conference on Natural Language Processing and Knowledge Engineering*, Beijing, 2007.