The Islamic University - Gaza

Deanery of Higher Studies

Faculty of Engineering

Computer Engineering Department

The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification

By

Motaz K. Saad

Supervisor

Dr. Wesam Ashour

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

In

Computer Engineering

1431H (Sep, 2010)

بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

عمادة الدراسات العليا
هاتف داخلي: 1150

ج س غ/35/
2010/08/31
الرقم..... Ref
التاريخ..... Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **معتز خالد حسين سعد**، لنيل درجة الماجستير في **كلية الهندسة** قسم **هندسة الحاسوب** وموضوعها:

The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification

وبعد المناقشة المغلقة التي تمت اليوم السبت 25 رمضان 1431هـ، الموافق 2010/09/04م الساعة الثانية عشرة والنصف ظهراً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. وسام محمود عاشور	مشرفاً ورئيساً	.....
أ.د. إبراهيم سليمان أبو هيبه	مناقشاً داخلياً	.....
د. علاء مصطفى الهليس	مناقشاً داخلياً	.....

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،،،

عميد الدراسات العليا

د. زياد إبراهيم مقداد

P. O. Box 108, Rimal, Gaza, Palestine fax: +970 (8) 286 0800 فاكس Tel: +970 (8) 286 0700 ص.ب. 108 الرمال، غزة، فلسطين هاتف
public@iugaza.edu.ps www.iugaza.edu.ps

Deduction

To the memory of my father ...

To my mother ...

To my brothers and sisters ...

Acknowledgements

Thanks to Allah for giving me the power and help to accomplish this research. Without the grace of Allah, I was not able to accomplish this work.

I would like to thank my parents very much for their pray, patience, motivation, and continues support. I also extend my thanks to all my family members for their motivation and support.

I am grateful to my supervisor, Dr. Wesam Ashour, for his enormous support, valuable guide, and assistance throughout the work of this research.

Special thanks for Dr. Alaa El-Halees for his valuable guide and comments.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

TM	Text Mining
TC	Text Classification / Text Categorization
VSM	Vector Space Model
IR	Information Retrieval
BOT	Bag of Tokens
CA	Classical Arabic
MSA	Modern Standard Arabic
DA	Dialectic Arabic
NB	Naïve Bayes
CNB	Complement Naïve Bayes
NBM	Naïve Bayes Multinomial
DMNB	Discriminative Multinomial Naïve Bayes
SVMs	Support Vector Machines
KNN	K Nearest Neighbors
DT	Decision Trees
ME	Maximum Entropy
NN	Neural Networks
FE	Frequency Estimate
DFE	Discriminative Frequency Estimate
OSAC	Open Source Arabic Corpus

أثر معالجة النصوص و توزين الكلمات على تصنيف النصوص العربية

معتز خالد سعد

الملخص

هذا البحث يعرض ويقارن تأثير معالجة النصوص وتوزين الكلمات على تصنيف النصوص العربية باستخدام المصنفات الشائعة حيث ان الابحاث الموجودة لم تتعرض لتأثير معالجة النصوص العربية على تصنيفها. تتضمن معالجة النصوص المعالجة الصرفية (التجذير والتجذير الخفيف للكلمات العربية) وتوزين الكلمات. طبّقنا المصنفات المذكروة سابقاً على سبع مجموعات من البيانات النصية العربية. اظهرت النتائج ان التجذير الخفيف هو الاسلم لغويا وانه الافضل من ناحية السرعة والدقة. كما اظهرت النتائج تفوق $SVMs$ و NB المعدل على باقي المصنفات. واظهرت النتائج ان توزين الكلمات له تأثير كبير على المصنفات التي تعتمد على دالة المسافة.

الكلمات المفتاحية

تصنيف النصوص العربية، معالجة النصوص العربية، التحليل الصرفي للغة العربية.

The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification

Motaz K. Saad

Abstract

This research presents and compares the impact of text preprocessing, which has not been addressed before, on Arabic text classification using popular text classification algorithms; Decision Tree, *K* Nearest Neighbors, Support Vector Machines, Naïve Bayes and its variations. Text preprocessing includes applying different term weighting schemes, and Arabic morphological analysis (stemming and light stemming). We implemented and integrated Arabic morphological analysis tools within the leading open source machine learning tools: Weka, and RapidMiner. Text Classification algorithms are applied on seven Arabic corpora (3 in-house collected and 4 existing corpora). Experimental results show: (1) Light stemming with term pruning is best feature reduction technique. (2) *Support Vector Machines* and *Naïve Bayes* variations outperform other algorithms. (3) Weighting schemes impact the performance of distance based classifier.

Keywords

Arabic Text Mining, Arabic text preprocessing / classification, Term weighting, Arabic morphological analysis (Arabic stemming / light stemming), Vector Space Mode (VSM), TFIDF, probabilistic text classification.

Chapter 1 : Introduction

This chapter introduces text mining (*TM*) and text classification (*TC*), describes Arabic Language, discusses the complexity of Arabic Language, and finally states the research motivation.

1.1 Text Mining (*TM*)

Data mining is the process of extracting patterns from data. Data mining is becoming an increasingly important tool to transform the data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery [30, 46, 86].

Data mining can be applied on a variety of data types. Data types include structured data (relational), multimedia data, free text, and hypertext as shown in Figure 1.1. We can strip hypertext from XML/XHTML tags to get free text [43, 49].

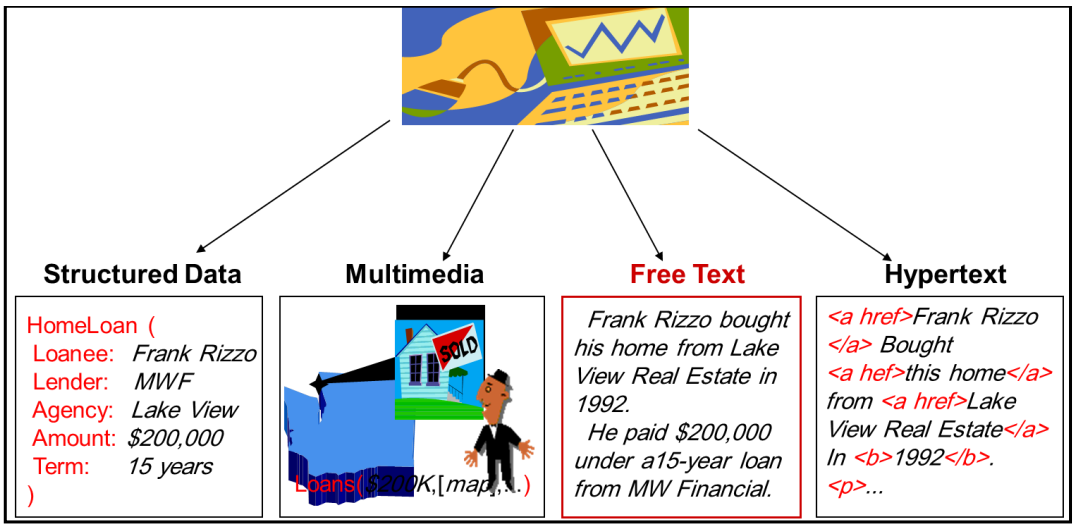

Figure 1.1: Data mining over verity of data [46]

Text mining, sometimes alternately referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the divining of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output as shown in Figure 1.2. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. Typical text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities) [43, 49].

The purpose of Text Mining is to process unstructured (textual) information, extract meaningful numeric indices from the text, and make the information contained in the text accessible to the various data mining algorithms. Information can be extracted to derive summaries for the words contained in the documents or to compute summaries for the documents based on the words contained in them. Hence, we can analyze words, clusters of words used in documents, etc., or we could analyze documents and determine similarities between them or how they are related to other variables of interest in data mining. In the most general terms, text mining will "turn text into numbers" (meaningful indices), which can then be incorporated in other analyses such as predictive/descriptive data mining [43, 49].

Text mining is well motivated, due to the fact that much of the world’s data can be found in text form (newspaper articles, emails, literature, web pages, etc.). Text mining tasks include

text categorization, clustering, document summarization, and extracting useful knowledge/trends [43, 49].

Figure 1.2: Text Mining Process

1.2 Text Classification (*TC*)

Text classification (*TC* – also known as text categorization, or topic spotting) is the task of automatically sorting a set of documents into categories (or classes, or topics) from a predefined set [43, 49]. This task, that falls at the crossroads of information retrieval (*IR*) and machine learning (*ML*), has witnessed a booming interest in the last ten years from researchers and developers alike [43, 49].

TC can provide conceptual views of document collections and has important applications in the real world. For example, news stories are typically organized by subject categories (topics) or geographical codes; academic papers are often classified by technical domains and sub-domains; patient reports in health-care organizations are often indexed from multiple aspects, 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 [30, 46, 87, 49]. Another widespread application of text categorization is spam filtering, where email messages are classified into the two categories spam and non-spam [43, 49].

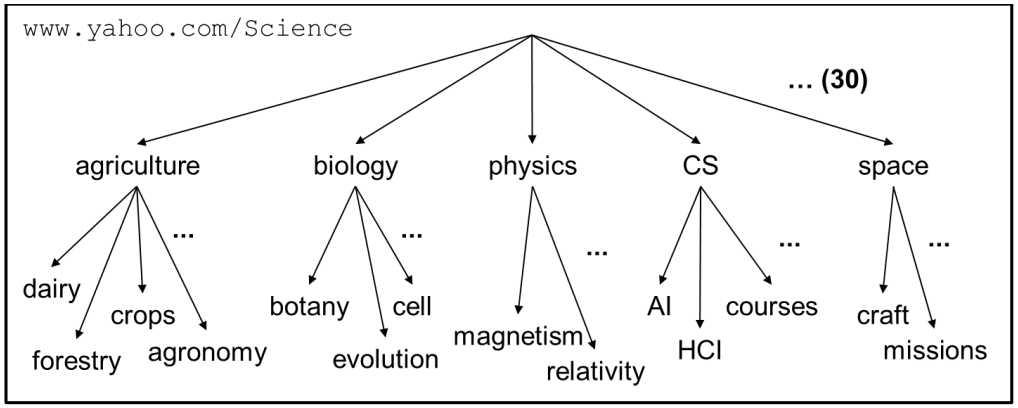

Figure 1.3: Yahoo.com Science directory

Automatic text categorization can significantly reduce the cost of manual categorization, for example MEDLINE (National Library of Medicine) uses $2 million/year for manual indexing of journal articles [13, 24], another example is Yahoo site which uses more than 200 expert people to manually label or categorize its web site pages where it receives hundreds of pages daily [13, 24]. Figure 1.3 shows topic hierarchy (topic classification) in Yahoo Science directory. Figure 1.4 and 1.5 show Google Arabic directory[1] and Yahoo Arabic Directory (Maktoob)[2] respectively. Note that the sport category in Google Arabic directory has 172 sites while it has 2 sites in Yahoo Arabic directory.

Figure 1.4: Google Arabic Directory

[1] http://www.google.com/Top/World/Arabic/
[2] http://www.maktoob.com

The web continues to grow at staggering rates. Automated search engines are increasingly unable to turn up useful results to search queries. The small paid editorial staffs at commercial directory sites can't keep up with submissions, and the quality and comprehensiveness of their directories has suffered. Instead of fighting the explosive growth of the Internet, the Open Directory provides the means for the Internet to organize itself. As the Internet grows, so do the number of net-citizens. These citizens can each organize a small portion of the web and present it back to the rest of the population, culling out the bad and useless and keeping only the best content. The Open Directory Project [51] is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors [51]. Google Arabic Directory depends on Open directory for websites classification.

Figure 1.5: Yahoo Arabic Directory (Maktoob)

Making the text at human level understanding to machines is not trivial task. The process includes deriving linguistic features from text to be at human like interpretation to be mined. Text mining must overcome a major difficulty that there is no explicit structure [43, 49]. Machines can reason relational data well since schemas are explicitly available. However, text encodes all semantic information within natural language. Text mining algorithms, then, must make some sense out of this natural language representation. Humans are great at doing this, but this has proved to be a problem for machines [43, 49].

The text classification problem is composed of several sub problems, which have been studied intensively in the literature such as the document indexing, the weighting assignment, document clustering, dimensionality reduction, threshold determination and the type of classifiers [30, 43, 46, 49, 86,]. Several methods have been used for text classification such as: Support Vector Machines (*SVMs*) [9, 38, 65, 66, 84, 102], *K* Nearest Neighbor (*KNN*) [13, 38, 54, 85], Neural Networks (*NN*) [10, 11, 38, 48], Naïve Bayes (*NB*) [38, 41, 54, 63, 72, 101], Decision Trees (*DT*) [9, 74], Maximum Entropy (*ME*) [39, 76], *N-Grams* [57, 69],and Association Rules [17, 40].

Text processing includes tokenizing string to words, normalizing tokenized words, remove predefined set of words (stopwords), morphological analysis, and finally term weighting [43, 49]. More details about text preprocessing in chapter 4.

Term indexing and weighting aim to represent high quality text. High quality in text mining usually refers to some combination of relevance, novelty, and interestingness. Several approaches have been used to index and weight terms but all of them share the following characteristics: The more the number of times a term occurs in documents that belong to some category, the more it is relative to that category [43, 49]. The more the term appears in different

documents representing different categories, the less the term is useful for discriminating between documents as belonging to different categories. The most commonly used weighting approach is the Term Frequency Inverse Document Frequency *tf-idf* [43, 49] which will be described in details in chapter 4.

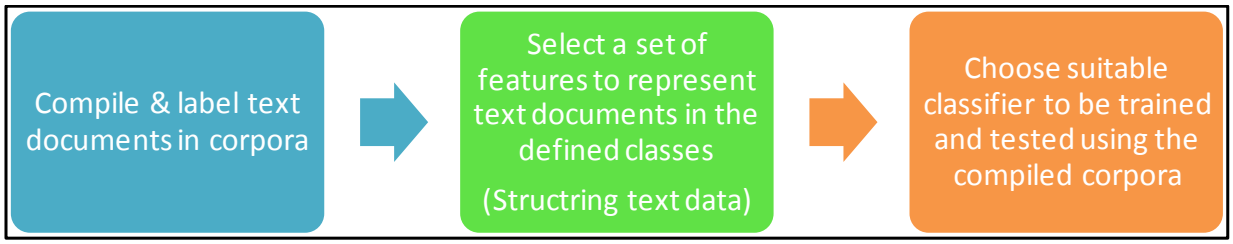

Figure 1.6: Building Text Classification System Process

The main consecutive phases of building a text classification system which involve compiling and labeling text documents in corpus, selecting a set of features to represent text documents in a defined set classes or categories (structuring text data), and finally choosing a suitable classifier to be trained and tested using the compiled corpus (Figure 1.6). The constructed classifier system then can be used to classify new (unlabeled) text documents as shown in Figure 1.7.

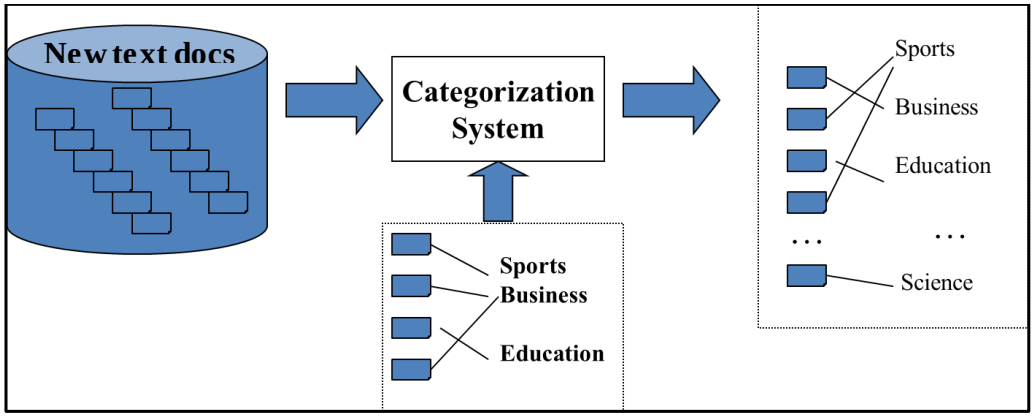

Figure 1.7: Classifying new text documents using text classification system

Structuring text data is a process to view text as a bag-of-tokens (words). This is the same approach as Information Retrieval (*IR*). Under that model we can already summarize, classify, cluster, and compute co-occurrence statistics over free text. These are quite useful for mining and managing large volumes of free text. However, the *BOT* approach loses a lot of information contained in text, such as word order, sentence structure, and context; these are precisely the features that humans use to interpret text. Natural Language Processing (*NLP)* attempts to understand document completely (at the level of a human reader). General *NLP* has proven to be too difficult because text is highly ambiguous. Natural Language is meant for human consumption and often contains ambiguities under the assumption that humans will be able to develop context and interpret the intended meaning [4, 13, 14, 52]. Figure 1.8 shows the process of structuring text data as Vector Space Model (*VSM)*.

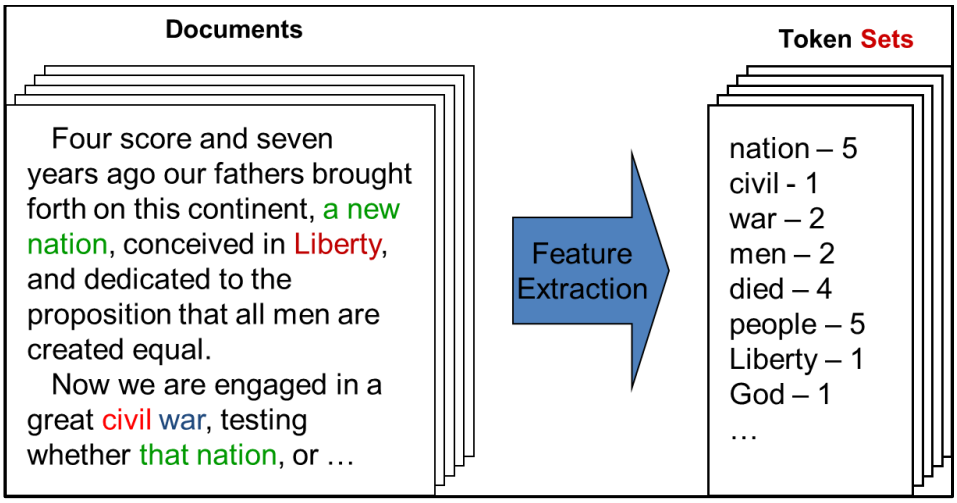

Figure 1.8: Structuring text data as *VSM*

1.2 Arabic Language

Arabic Language is the 5th widely used languages in the world. It is spoken by more than 422 million people as a first language and by 250 million as a second language [19]. Arabic Language belongs to the Semitic language family. Semitic languages are commonly written

without the vowel marks which would indicate the short vowels. Semitic languages can get away with this because they all have a predictable root pattern system [56]. Arabic alphabet consists of the following 28 letters (أ ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي) in addition, the Hamza (ء). There is no upper or lower case for Arabic letters like English letters. The letters (أ و ي) are vowels, the rest are constants. Unlike Latin-based alphabets, the orientation of writing in Arabic is from right to left [19, 31, 32, 33, 34, 35, 56].

The Arabic script has numerous diacritics, including **iʿjam** (**إعجام**), consonant pointing, and **tashkīl** (**تشكيل**), supplementary diacritics. The latter include the **ḥarakāt** (**حركات**, singular ḥaraka **حركة**), vowel marks. The literal meaning of *taškīl* is "forming". As the normal Arabic text does not provide enough information about the correct pronunciation, the main purpose of *tashkīl* (and *ḥarakāt*) is to provide a phonetic guide or a phonetic aid; i.e. show the correct pronunciation (double the word in pronunciation or to act as short vowels). The *ḥarakāt,* which literally means "motions", are the short vowel marks. There is some ambiguity as to which *tashkīl* are also *ḥarakāt;* the *tanwīn,* for example, are markers for both vowels and consonants [18].

Arabic diacritics include: Fatha, Kasra, Damma, Sukūn, Shadda, and Tanwin. The pronunciations of aforementioned diacritics for the Arabic letter (ب) are presented in Table 1.1. Arabic words may also have Tatweel or kasheeda as shown in figure 1.9.

Table 1.1: Diacritics

Double Constant	**No Vowel**	**Nunation**			**Vowel**		
بّ **/bb/**	بْ /b/	بٍ /bin/	بٌ /bun/	بً /ban/	بِ /bi/	بُ /bu/	بَ /ba/

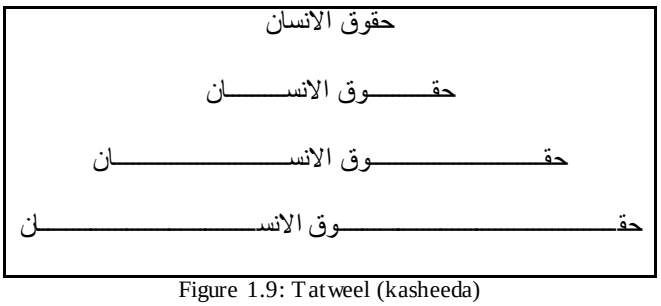

Figure 1.9: Tatweel (kasheeda)

Arabic words have two genders, masculine (مذكر) and feminine (مؤنث); three numbers, singular (مفرد), dual (مثنى), and plural (جمع); and three grammatical cases, nominative (الرفع), accusative (النصب), and genitive (الجر). A noun has the nominative case when it is subject (فاعل); accusative when it is the object of a verb (مفعول); and the genitive when it is the object of a preposition (مجرور بحرف جر). Words are classified into three main parts of speech, nouns (اسماء) (including adjectives (صفات) and adverbs (ظروف)), verbs (افعال), and particles (ادوات).

Arabic has 3 forms; Classical Arabic (*CA*), Modern Standard Arabic (*MSA*), and Dialectal Arabic (*DA*). *CA* includes classical historical liturgical text, *MSA* includes news media and formal speech, and *DA* includes predominantly spoken vernaculars and has no written standards.

1.2.1 Complexity of Arabic Language

Arabic is a challenging language for a number of reasons [4, 5, 6, 7, 8, 9, 10, 11, 13, 33, 38, 44, 54, 56, 69, 73, 83, 98, 101]:

- Orthographic (الاملاء) with diacritics is less ambiguous and more phonetic in Arabic, certain combinations of characters can be written in different ways.
- Arabic language has short vowels which give different pronunciation. Grammatically they are required but omitted in written Arabic texts.
- Arabic has a very complex morphology as compare to English language.

- Synonyms are widespread. Arabic is a highly inflectional and derivational language.
- Automatic *TC* depends on the contents of documents, a huge number of features or keywords can be found in Arabic text such as morphemes that may generated from one root which may lead to a poor performance in terms of both accuracy and time.
- Lack of publically freely accessible Arabic Corpora.

In the following, we shall discuss these points in details.

1.2.2 Examples from Arabic show the complex nature of Arabic Language

1.2.2.1 Word meanings

It is possible to identify the different meanings associated with a word, due to one word may have more than one meaning in different contexts, by using corpus this kind of ambiguity can be authentically detected. Table 1.2 shows the Arabic word (قلب) which has 3 meaning as a noun.

Table 1.2: The meaning of word (قلب) as a noun

Word meaning	Sentence
core	في **قلب** الاحداث
heart	اجرى عملية **قلب** مفتوح
center, middle	الكرة في **قلب** الملعب

1.2.2.2 Variations in lexical category

One word may have more than lexical category (noun, verb, adjective, etc.) in different contexts as shown in Table 1.3. Morphological analysis of a given corpus includes investigating word frequency of a word as a lexical category.

Table 1.3: The Lexical Category of word (عين)

Word meaning	Word Category	Sentence
Ain	Proper-Noun	**عين** جالوت
wellspring	Noun	**عين** الماء
eye	Noun	**عين** الانسان
delimitate/be delimitate	Verb/passive Verb	**عين** وزيرا للخارجية

1.2.2.3 Synonyms

Languages have many words that are considered synonymous. Through a given corpus, the researchers can use morphological analysis tools to know synonyms of a word, the frequency of each word of those synonyms and which one of them is more common. Examples of synonyms in Arabic are (بذل منح اعطى وهب) which means (give), (اسرة عائلة) which means (family), and (فصل صف) which means (classroom).

1.2.2.4 The word form according to its case

The form of some Arabic words may change according to their case modes (nominative, accusative or genitive). For instance the plural of word (مسافر) which means (traveler) may be the form (مسافرون) in the case of nominative (مرفوعة) and the form (مسافرين) in the case of accusative/genitive (منصوبة/مجرورة). Arabic light stemming can handle these cases. More details in chapter 4.

1.2.2.5 Morphological characteristics

An Arabic word may be composed of a stem plus affixes and clitics. The stem consists of a consonantal root (جذر صحيح) and a pattern morpheme (اصغر كلمة ذات معنى). The affixes include inflectional markers (علامات او حركات اعرابية) for tense, gender, and/or numbers. The clitics include some prepositions (حروف جر), conjunctions (حروف العطف), determiners (محددات), possessive pronouns (ضمائر الملكية) and pronouns (ضمائر). The clitics attached to the beginning of a stem are

called proclitic and the ones attached to the end of it are called enclitics. Most Arabic morphemes are defined by three consonants, to which various affixes can be attached to create a word. For example, from the tri-consonant "ktb" (كتب), we can inflect (يصرف) several different words concerning the idea of writing as (wrote كَتب), (book كِتاب), (the book الكِتاب), (books كُتُب), (he writes يكْتُب), (author كاتِب), (library مكْتَبَة). Moreover an Arabic word may correspond to several English words. Because of the variability of prefixes and suffixes, the morphological analysis is an important step in Arabic text processing. For example, the Arabic word (وبنفوذها) and its equivalence in English "and with her influences". This makes segmentation of Arabic textual data different and more difficult than Latin languages.

Table 1.4: Affix set in Arabic Language

Affixes in Arabic	**Examples**
Prefixes of length three	ولل ، وال ، كال ، بال
Length tow prefixes	ال ، لل
Length one prefixes	ل ، ب ، ف ، س ، و ، ى ، ت ، ن ، ا
Length three suffixes	تمل ، همل ، تان ، تين ، كمل
Length two suffixes	ون ، ات ، ان ، ين ، تن ، كم ، هن ، نا ، يا ، ها ، تم ، كن ، ني ، وا ، ما ، هم
Length one suffixes	ة ، ه ، ي ، ك ، ت ، ا ، ن

Table 1.5: Arabic Patterns and Roots

Arabic Pattern and roots (الأوزان)	**Examples**
Length four pattern	فاعل فاعول فعلة فعال مفعل
Length five pattern and length three roots	تفاعل افتعل افعال فعالة فعلان فعولة تفعلة تفعيل مفعلة مفعول فاعول فواعل مفاعل مفعيل افعلة فعائل منفعل مفتعل فاعلة مفاعل فملاع يفتعل تفتعل فعلالي انفعال
Length five pattern and length four roots	تفعلل افعلل مفعلل فعللة فعلان فعالل
Length six pattern and length three roots	استفعل مفاعلة افتعال افعوعل انفعل مستفعل
Length six pattern and length four roots	افنلل افعلال متفعلل

Affixes set in Arabic are shown in Table 1.4, and Arabic patterns (الأوزان) and roots are shown in Table 1.5. The word (علم) may give various meanings by adding different affixes (prefixes, infixes, or suffixes) as shown in Table 1.6. Other morphological variations example is the word (يذهب) which means (go) are pretested in Table 1.7.

Table 1.6: Versions of the word (علم) and its meaning when adding affixes

Meaning	Suffix	Infix	Prefix	Word
Scientific	ية	***	***	علمية
Learned us	تنا	***	***	علمتنا
His science	ه	***	***	علمه
Scientists	اء	***	***	علماء
Teaching	***	ي	ت	تعليم
Sciences	***	و	***	علوم
Informative	يه	ا	است	استعلامية

Table 1.7: Morphological variation of word (ذهب)

verb	time	Number of participants	Gender of subjects
ذَهب	Past	1	Male
ذَهبت	Past	1	Female
ذَهبا	Past	2	Male
ذَهبتا	Past	3	Female
ذَهبوا	Past	3 or more	Male
ذَهبن	Past	3 or more	Female
يذْهب	Present	1	Male
تذْهب	Present	1	Female
سيذْهب	Future	1	Male
ستذْهب	Future	1	Female
سيذْهبوا	Future	3 or more	Male
سيذْهبن	Future	3 or more	Female

Table 1.8: Different meaning of morphology of the same root in Arabic

Meaning	Root	Word
Class room **Apartheid**	فصل	**الفصل** الدراسي **الفصل** العنصري
Goes out of house **Graduate** from university	خرج	**يخْرج** من البيت **تخّرج** من الجامعة
The fisherman **twist** the cord The student **argued** with the teacher	جدل	**جدل** الصياد الحبل **جادل** الطالب المدرس
He **focuses** the arrow The man lost **his mind**	صوب	انه **يصوب** السهم فقد الرجل **صوابه**

Stemming usually used to convert words to root form, it dramatically reduces the complexity of Arabic language morphology by reducing the number of feature / keywords in corpora. The reason for using stemming as feature reduction technique is that all morphology of words mostly has the same context meaning, but the case is not always true. Table 1.8 shows

some of these cases. There is another approach for morphology reduction that just removes affixes and does not convert the word to bas/root form. This approach is called light stemming. More details in chapter 4.

1.2.2.6 Encoding Problem

Arabic Language has display Problems (encoding issues) because it has different encoding according to machine platform. Figure 1.10 shows encoding problem where all shaded cells are displayed correctly while the other cells are not displayed correctly. Text preprocessing and classification with incorrect encoding may lead to incorrect results. Table 1.9 presents the characteristics of two common Arabic encoding systems; Unicode and *CP-1256* code page 1256 Arabic windows.

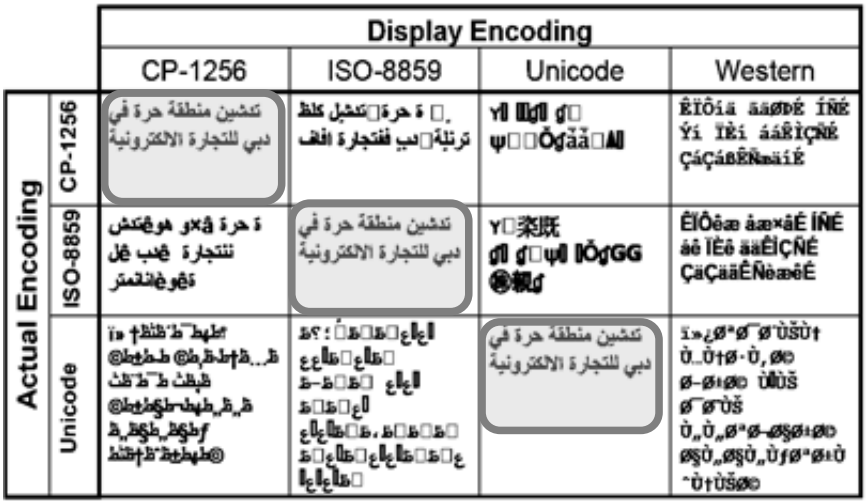

Figure 1.10: Arabic Encoding Problem

Table 1.9: Unicode vs. cp-1256 Arabic windows encoding

Unicode	CP-1256 code page 1256 Arabic windows
Becoming the standard more and more	Commonly used
2-byte characters	1-byte characters
Widely supported input/display	Widely supported input/display
Supports extended Arabic characters	Minimal support for extended Arabic characters
Multi-script representation	bi-script support (Roman/Arabic)
Supports presentation forms (shapes and ligatures)	Tri-lingual support: Arabic, French, English (ala ANSI)

1.2.3 Arabic Corpus Problem

Text data mining is a multidisciplinary field involving information retrieval, text analysis, information extraction, clustering, categorization and linguistics. Text mining is becoming of more significance, and efforts have been multiplied in studies to provide for fetching the increasingly available information efficiently [6, 7]. Due to the Arabic language lacking of corpora, it is difficult to represent textual content and quantitative data of Arabic [6, 7].

Corpus-based approaches to language have introduced new dimensions to linguistic description and various applications by permitting some degree of automatic analysis of text. The identification, counting and sorting of words, collocations and grammatical structures which occur in a corpus can be carried out quickly and accurately by computer, thus greatly reducing some of the human drudgery sometimes associated with linguistic description and vastly expanding the empirical basis [6, 7]. Linguistic research has become heavily reliant on text corpora over the past ten years. Due to the increasing need of an Arabic corpus to represent the Arabic language and because of the trials to build an Arabic corpus in the last few years were not enough to consider that the Arabic language has a real, representative and reliable corpus, it was necessary to build such an Arabic corpus to support various linguistic research on Arabic [6, 7].

One of the difficulties that encountered this work and other researches in the field of Arabic linguistics was the lack of publicly available Arabic corpus for evaluating text categorization algorithms [6, 7, 15, 16]. Arabic corpus problem was posed by [6, 7, 15, 16]. A survey by [6, 7] confirms that existing corpora are too narrowly limited in source-type and genre, and that there is a need for a freely-accessible Corpus of Contemporary Arabic (*CCA*) covering a broad range of text-types. Chapter 5 lists the available free and none-free Arabic corpora.

Al-Nasray et. al. [6, 7] discussed three axes in their paper; the first axes is a survey of the importance of corpora in language studies e.g. lexicography, grammar, semantics, Natural Language Processing and other areas. The second axis demonstrates how the Arabic language lacks textual resources, such as corpora and tools for corpus analysis and the effected of this lack on the quality of Arabic language applications. There are rarely successful trials in compiling Arabic corpora, therefore, the third axis presents the technical design of the International Corpus of Arabic (*ICA*), a newly established representative corpus of Arabic that is intended to cover the Arabic language as being used all over the Arab world. The corpus is planned to support various Arabic studies that depends on authentic (اصيلة) data, in addition to building Arabic Natural Language Processing Applications.

International Corpus of Arabic (*ICA*) is a big project initiated by Bibliotheca Alexandrina (*BA*). *BA* is one of the international Egyptian organizations that play a noticeable role in disseminating culture and knowledge, and in supporting scientific research. *ICA* is a real trial to build a representative Arabic corpus as being used all over the Arab world to support research on Arabic [6, 7]. *ICA* corpus has been analyzed by Al-Nasry et. al. in [7], they shed light on the levels of corpus analysis e.g. morphological analysis, lexical analysis, syntactic analysis and semantic analysis. Al-Nasry also demonstrates different available tools for Arabic morphological analysis (*Xerox, Tim Buckwalter, Sakhr and RDI*). The morphological analysis of *ICA* includes: selecting and describing the model of analysis, pre-analysis stage and full text analysis stages. *ICA* is not publically available now and it expected to be released soon.[3]

[3] http://www.bibalex.org/unl/Frontend/Project.aspx?id=9

1.3 Research Motivation

The majority of works have been done in automatic text classification for documents written in English. Despite Arabic is used widely, the work on the retrieval/mining of Arabic text documents is fairly limited in the literature [4, 5, 6, 7, 8, 9, 10, 11, 13, 33, 38, 44, 54, 56, 69, 73, 83, 98, 101]. This is due to the unique nature of Arabic language morphological principles as mentioned in section 1.2.

There has been a debate among researchers about the benefits of using morphological tools in *TC* [27, 73, 79, 80, 102]. Studies in the English language illustrated that performing stemming during the preprocessing step degrades the performance slightly [79, 80]. However, they have a great impact on reducing the memory requirement and storage resources needed. The experiment conducted by [27] illustrates that selecting 10% of features exhibits the same classification performance as when using all the features when using *SVMs* in classification. This may indicate that using preprocessing tools and dimensionality reduction techniques is not necessary, for the English language, from the performance view point (accuracy and time) when using a robust classifier such as *SVMs*. However, preprocessing tools are essential for decreasing the training time and storage required as indicated by [102]. The effect of the preprocessing tools on Arabic text categorization is an area of research [73].

1.3.1 Research Problems

The following points describe the research problems:

- Debate among researchers about the benefits of using English morphological tools in *TC.* To the best of our knowledge, the benefits of using Arabic morphological tools (stemming

and light stemming) is not address for Arabic Language; only [31, 32] applied on single corpus belong to only 3 categories.

- To the best of our knowledge, the impact of text preprocessing and different term weighting schemes combinations on Arabic text classification using popular text classification algorithms has not been studied in the literature. Only [31, 73] have addressed the impact of morphological analysis tools on Arabic text classification. Their work is not comprehensive regarding Arabic corpora, classifiers, and term weighting schemes. Furthermore, our results are different from their results. More details and explanations are reported in chapter 6.
- To the best of our knowledge, the following question is not posed in the literature: how much the time and storage saved using preprocessing (morphological analysis feature reduction and term weighting) to get accurate classification model? Is the time feasible? Maybe we can get accurate classification model when we work on raw text with feasible time. i.e., preprocessing is not necessary. Formally speaking, we need to make a trade-off between preprocessing time, classification time, and required memory storage to run the process.
- The lack of availability of publically free accessible Arabic Corpora.
- The lack of standard Arabic morphological analysis tools.
- Most of related works in the literature used small in-house collected corpus.
- Most of related works in the literature applied one or two classifiers to classify one corpus. This is not enough to evaluate Arabic *TC*.

- There are contradictions between results of researches in the literature because of using different corpora and different preprocessing techniques.
- Probabilistic classifiers (*NB* and its variant) that depend on Language model have been not addressed for Arabic *TC* in the literature.

In the following, we shall state the research objectives briefly and describe them in research contributions.

1.3.2 Research Objectives

- Build the largest publically free accessible Arabic Corpora.
- Implement and integrate Arabic morphological analysis tools.
- Conduct a comprehensive study about the impact of text preprocessing on Arabic text classification, and resolve the contradiction in the literature.
- Provide comprehensive guidelines to help in making trade-off between accuracy and time storage requirements.

1.3.3 Research Contributions

- One of the aims of this research is to compile representative Arabic corpora that cover different text genres which will be used in this research and can be used in this research and in the future as a benchmark. Therefore, three different corpora were compiled covering different genres and subject domains. The corpora were collected from different sources and various domains. The corpora is available publically accessible freely at [68]. The first corpus was collected from *BBC* Arabic website, the second was collected from *CNN* Arabic website, and the last one was collected from multiple websites. The corpus is

the largest Arabic text dataset; it contains 18M words, and has 0.5M distinct keywords after removing stopwords. The corpora can be used for computation linguistics researches including text mining, information retrieval. Compiling freely and publically available corpora is advancement step on the field of computational linguistics.

- Implement and integrate Arabic morphological analysis tools (stemming and light stemming) into leading open source machine learning tools (Weka and RapidMiner). The tools are available publically accessible freely at [68]. The implemented Arabic morphological analysis tools were applied on Arabic corpora.
- Apply 7 *TC* algorithms on seven Arabic corpora (3 in-house collected and 4 existing corpora). *TC* algorithms include: *C4.5* Decision trees (*C4.5 DT*), *K* Nearest Neighbors (*KNN*), Support Vector Machines (*SVMs*), Naïve Bayes (*NB*), and *NB* variants (Naïve Bayes Multinomial (*NBM*), Complement Naïve Bayes (*CNB*), and Discriminative Multinomial *NB* (*DMNB*)). Applying 7 TC algorithms on 7 corpora resolves contradictions in the literature.
- Apply different term weighting schemes (Boolean, word count, word count normalization, term frequency, term frequency inverse document frequency, and term pruning) on Arabic corpora and investigate it impact on Arabic *TC*. Different weighting schemes have not been address in the literature for Arabic Language.
- This research is a comprehensive study for Arabic text classification. We investigate the impact and the benefits of using different Arabic morphological techniques with different weighting schemes applied on seven corpora and using seven classifiers. The total number of carried experiments is 1617. We have 33 different representations for Arabic text, 7 classifiers and 7 Arabic Corpora.

- Provide comprehensive analysis about the trade-off between preprocessing time, classification time, and required memory storage to get accurate classification model.

1.4 Thesis Structure

The rest of the report is organized as follows: chapter 2 review related work; chapter 3 summarizes text classification algorithms; Chapter 4 describes text preprocessing steps and stages; Chapter 5 presents the used and compiled Arabic corpora; Experimental results are presented in chapter 6, and finally, we draw the conclusion.

Chapter 2 : Related Work

Many researchers have been worked on text classification in English and other European languages such as French, German, Spanish [1, 26], and in Asian languages such as Chinese and Japanese [70]. However, researches on text classification for Arabic language are fairly limited [4, 5, 6, 7, 8, 9, 10, 11, 13, 33, 38, 44, 54, 56, 69, 73, 83, 98, 101].

Researches on the field of Arabic *TC* fall into four categories: applying classification algorithms on Arabic text, comparing classification algorithms applied on Arabic text, proposing new classification methods, and investigates the impact of preprocessing.

2.1 Applying Classification Algorithms on Arabic Text

El-Kourdi et. al. [41] 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 (*ME*) used by **El-Halees** [39] for Arabic text classification, and by **Sawaf et. al**. [76] (2001) to classify and cluster News articles. The best classification accuracy reported by El-Halees was 80.4% and 62.7% by Sawaf.

Association Rules used by **El-Halees** [40], and by **Al-Zoghby** [17] to classify Arabic documents. The classification accuracy reported by El-Halees was 74.41%. Al-Zoghby used *CHARM* algorithm and showed the excellence of soft-matching over hard big *O* exact matching.

Al-Zoghby used a corpus consisting of 5524 records. Each record is a snippet of emails having the subject "nuclear". The vocabulary size after removing stopwords and punctuations is 103,253 words. The average size of text document is 18 words. The words of text documents were converted into the root form.

Mesleh applied *SVMs* to classify Arabic articles with Chi Square feature selection in [65], the reported F-measure by Mesleh is 88.11%. **Mesleh** also compared 6 feature selection methods with *SVMs* in [66], 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).

Harrag et. al. [47] improved Arabic text classification by feature selection based on hybrid approach. Harrag used decision tree algorithm and reported classification accuracy of 93% for scientific corpus, and 91% for literary corpus. Harrag collected 2 corpora; the first one is from the scientific encyclopedia *"Do You Know*" (هل تعلم). It contains 373 documents belonging to 1 of 8 categories (innovations, geography, sport, famous men, religious, history, human body, and cosmology), each category has 35 documents. The second corpus is collected from *Hadith encyclopedia* (موسوعة الحديث الشريف) from "*the nine books*" (الكتب التسعة). It contains 435 documents belonging to 14 categories.

KNN has been applied by **Al-Shalabi et. al**. [13] 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 1of 6 categories (politics 111, economic 179, sport 96, health and medicine 114, health and cancer 27, agriculture 100). They preprocessed the corpus by applying stopwords removal and light stemming.

Laila Kheirsat [57] used *N-grams* frequency statistics to classify Arabic text, she addressed high dimensional text data by mapping text documents to set of real numbers representing *tri-grams* frequency profile. The *N-gram* method is language independent and works well in the case of noisy-text. The *tri-grams* for the word (المسافر) are (الم ، لمس ، مسا ، ساف ، افر). Kheirsat classifies a test text document by computing Manhattan/Dice distance similarity measure to all training documents and assign the class of the training document with smallest/largest computed distance to the test text document. Kheirsat reported that *Dice* outperforms *Manhattan* distance measure. Although the *Manhattan* measure has provided good classification results for English text documents, it does not seem to be suitable for Arabic text documents. Kheirsat collected her corpus from Jordanian newspapers (*Al-Arab, Al-Ghad, Al-Ra'I, Ad-Dostor*). The corpus belongs to 1 of 4 categories (sport, economic, weather, and technology). She applied stopwords removal and used 40% for training and 60% for testing.

Harrag and El-Qawasmah [48] 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* (موسوعة الحديث الشريف) from "*the nine books*" (الكتب

التسعة). It contains 435 documents belonging to 14 categories. He applied light stemming and stopwords removal on his corpus. *tf-idf* is used as a weighting scheme.

2.2 Comparing Classification Algorithms Applied on Arabic Text

There are several studies compare classification algorithms on Arabic text. **Hmeidi et. al.** [50] 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 et. al. [3] 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 stopwords. After removing stopwords and infrequent words the vocabulary size became 7M words. *tf-idf* was used as weighting schemes.

In [34], **Duwairi** 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 stopwords removal and stemming. She used 50% for training and 50% for testing.

Kannan et. al. [55] 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 authors 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. 4-folds cross-validation was performed for evaluation.

Al-Harbi et. al. [9] 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 stopwords. The corpora are not available publically.

Bawaneh et. al. [22] applied *KNN* and *NB* on Arabic text and conclude that *KNN* has better performance than *NB*, they also conclude 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 [38] 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 term 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 (*www.aljazeera.net*). The documents categorized into six domains: politics, sports, culture and arts, science and technology, economy and health. The author applied stopwords removal and normalization and used 10-folds cross-validation for testing.

2.3 Proposing New Classification Methods

Duwairi [33, 35] proposed a distance-based classifier for categorizing Arabic text. Each category is represented as a vector of words in an m-dimensional space, and documents are classified on the basis of their closeness to feature vectors of categories. The classifier, in its learning phase, scans the set of training documents to extract features of categories that capture inherent category specific properties; in its testing phase the classifier uses previously determined category-specific features to categorize unclassified documents. The average accuracy reported was 0.62 for the recall and 0.74 for the precision. Duwairi collected 1000 text documents belonging to 10 categories (sport, economic, internet, art, animals, technology, plants, religious, politics, and medicine). Each category contains 100 documents. She used 50% for training and 50% for testing. Duwairi applied stemming for feature reduction.

Alruily et. al. [12] introduced initial prototype for identifying types from Arabic text, they explored 2 approaches to perform identification task; using gazetteers, and using rule-based system.

Abbas et. al. [2] proposed Triggered (*TR*) classifier. Triggers of a word W_k are ensemble of words which highly correlated with it. The main idea of *TR*-Classifier is computing the average mutual information (*AMI*) for each couple of words from the training documents and

testing document, and then assigns the topic that highest *AMT* to the test document. The best recall achieved is 0.9.

Ayadi et. al. [31] applied intertextual distance theory to classify any anonymous Arabic text according to criteria of lexical statistic, this requires integration of a metric for classification task using a database of lemmatized corpus.

Syiam et. al. [82] experimental results show that the suggested hybrid method of statistics and light stemmers is the most suitable stemming algorithm for Arabic language and gives general accuracy of about 98%.

2.4 Investigating The Impact of Preprocessing

Duwairi et. al [31, 32] compared three dimensionality reduction techniques; stemming, light stemming, and word cluster. Duwairi used *KNN* to perform the comparison. Performance metrics are: time, accuracy, and the size of vector. She showed that light stemming is the best in term classification accuracy. Duwairi collected 1,500 documents belonging to one of three categories (sport, economic, education). Each category has 5,000 documents. She split the corpus; 9,000 documents for training and 6,000 documents for testing.

Thabtah et. al. [85] investigates different variations of *VSM* and term weighting approaches using *KNN* algorithm. Her experimental results showed that *Dice* distance function with *tf-idf* achieved the highest average score. The authors used the corpus collected by [13].

Said et. al [73] provided an evaluation study of several morphological tools for Arabic Text Categorization using *SVMs*. Their study includes using the raw text, the stemmed text, and the root text. The stemmed and root text are obtained using two different preprocessing tools. The results revealed that using light stemmer combined with a good performing feature selection

method such as mutual information or information gain enhances the performance of Arabic Text Classification for small sized datasets and small threshold values for large datasets. Additionally, using the raw text leads to the worst performance in small datasets while its performance was among the best tools in large datasets. This may explain the contradiction in the results obtained previously in the literature of the Arabic text categorization since the performance of the preprocessing tools is affected by the characteristics of the dataset used.

From previous discussion, most of related work in the literature used small in-house collected corpus, and applied one or two classifiers to classify one corpus which is not enough to evaluate Arabic *TC*. Thus, there are contradictions between results of researches in the literature because of using different corpora and different preprocessing techniques. In addition, the impact of text preprocessing and different term weighting schemes combinations on Arabic text classification using popular text classification algorithms has not been studied in the literature. Also, there is a debate among researchers about the benefits of using morphological tools in *TC*.

In this research, we provide a comprehensive study for Arabic text classification. We investigate the impact and the benefits of using different Arabic morphological techniques with different weighting schemes applied on seven corpora and using seven classifiers. We also provide comprehensive analysis about the trade-off between preprocessing time, classification time, and required memory storage to get accurate classification model.

In the next chapter, we shall provide description about the text classifiers that used in this research.

Chapter 3 : Text Classifiers

This chapter describes famous *TC* algorithms: Support Vector Machines (*SVMs*), *K* Nearest Neighbors (*KNN*), Decision Trees (*DT*), Naïve Bayes (*NB*), and Naïve Bayes variants (Multinomial Naïve Bayes (*MNB*), Complement (*CNB*), and Discriminative Multinomial Naïve Bayes (*DMNB*)). The followings are brief overview on the classification algorithms mentioned above.

The goal of classification is to build a set of models that can correctly predict the class of the different objects. The input to these methods is a set of objects (i.e., training data), the classes which these objects belong to (i.e., dependent variables), and a set of variables describing different characteristics of the objects (i.e., independent variables). Once such a predictive model is built, it can be used to predict the class of the objects for which class information is not known a priori. The key advantage of supervised learning methods over unsupervised methods is having an explicit knowledge of the classes [30, 28, 46, 86].

3.1 Naïve Bayes

A Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naïve) independence assumptions [30, 28, 46, 86]. In simple terms, a naïve Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naïve Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple. [30, 28, 46, 86]

Derivation of Naïve Bayes Classifier

Depending on the precise nature of the probability model, naïve Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naïve Bayes models uses the method of maximum likelihood [30, 28, 46, 86]. The following provide description for *NB* derivation.

Let *D* be a training set of tuples and their associated class labels, and each tuple is represented by an *n-D* attribute vector $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$ and there are *m* classes $C_1, C_2, \ldots, C_m$. The maximum posteriori, i.e., the maximal $P(C_i|\boldsymbol{X})$ then can be derived from Bayes' theorem [30, 28, 46, 86]

$$P(C_{i|}|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \qquad 3.1$$

Since *P(X)* is constant for all classes, only eq. 3.2 needs to be maximized

$$P(C_i|X) = P(X|C_i)P(C_i) \qquad 3.2$$

Based on the assumption is that attributes are conditionally independent (i.e., no dependence relation between attributes), we can compute $P(X|C_i)$ using eq. 3.3

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \qquad 3.3$$

Eq. 3.3 greatly reduces the computation cost, only counts the class distribution. If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_i, D|$ (# of tuples of C_i in *D*). And if A_k is continuous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ and $P(x_k|C_i)$ is

$$P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \qquad 3.4$$

$$g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad 3.5$$

Where μ is the mean and σ is the variance. If an attribute value doesn't occur with every class value, the probability will be zero, and a posteriori probability will also be zero (no matter how likely the other values are). We can avoid zero probability by adding 1 to the count for every attribute value class combination (Laplace estimator or Laplace correction) [46, 86].

The Naïve Bayesian (*NB*) algorithm has been widely used for document classification, and has been shown to produce very good performance. For each document, the naïve Bayesian algorithm computes the posterior probability that the document belongs to different classes, and assigns it to the class with the highest posterior probability [59, 61, 60, 64].

Documents can be characterized by the words that appear in them, and one way to apply machine learning to document classification is to treat the presence or absence of each word as a Boolean attribute. The naïve part of *NB* algorithm is the assumption of word independence that the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category. There are two versions of *NB* algorithm. One is the multi-variate Bernoulli event model that only takes into account the presence or absence of a particular term, so it doesn't capture the number of occurrence of each word. The other model is the multinomial model that captures the word frequency information in documents. [30, 28, 46, 86]. *NBM* is described in more details in section 3.2.

An advantage of the naïve Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification [25, 28, 30, 46, 61, 86, 101]. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance

matrix [30, 28, 46, 86]. Naïve Bayes works surprisingly well (even if independence assumption is clearly violated) because classification doesn't require accurate probability estimates as long as maximum probability is assigned to correct class. Various empirical studies of this classifier in comparison to decision tree and neural network classifiers have found it to be comparable in some domains. In theory, Bayesian classifiers have the minimum error rate in comparison to all other classifiers. However, in practice this is not always the case, owing to inaccuracies in the assumptions made for its use, such as class conditional independence, and the lack of available probability data [25, 28, 30, 46, 61, 86, 101].

The normal-distribution assumption for numeric attributes is a restriction on naïve Bayes. Many features simply aren't normally distributed. However, there is nothing to prevent us from using other distributions for the numeric attributes. If we know that a particular attribute is likely to follow some other distribution, standard estimation procedures for that distribution can be used instead. If we suspect it isn't normal but don't know the actual distribution, there are procedures for "kernel density estimation" that do not assume any particular distribution for the attribute values. Another possibility is simply to discretize the data first [30, 28, 46, 49, 86].

Disadvantages of *NB* is the class conditional independence assumption, therefore *NB* losses accuracy, practically, when dependencies exist among variables dependencies among variables cannot be modeled by naïve Bayesian Classifier while Bayesian Belief Networks can deals with these dependencies.

3.2 Naïve Bayes Multinomial (*NBM*)

The multinomial model of naïve Bayesian classification algorithm captures the word frequency information in document. So it requires the word frequency that is not weighted and normalized [46, 63, 86].

Using multinomial probabilistic model as a Bayesian assumption tries to overcome the drawback of using multivariate Bernoulli model which represent a text document as a vector of binary attributes indicating which words occur and do not occur in the document. In Bernoulli model, the number of times a word occurs in a document is not captured. When calculating the probability of a document, one multiplies the probability of all attribute values, including the probability of non-occurrence for words that do not occur in the document [63, 86].

On the other hand, multinomial probabilistic model is a uni-grams language model with integer word counts. The document is represented by a set of word occurrences from the document. The number of occurrences of each word in the document is captured. When calculating the probability of a document, one multiplies the probabilities of each word that occur. The individual word occurrences can be understood as "event" and the document to be the collection of word events. This model is called multinomial event model. This approach is more traditional in statistical language modeling for speech recognition, where it would be called a "uni-grams language model" [63, 86].

Naïve Bayes is a popular technique for this application because it is very fast and quite accurate. However, this does not take into account the number of occurrences of each word, which is potentially useful information when determining the category of a document. Instead, a document can be viewed as a bag of words—a set that contains all the words in the document, with multiple occurrences of a word appearing multiple times (technically, a set includes each of its members just once, whereas a bag can have repeated elements). Word frequencies can be accommodated by applying a modified form of naïve Bayes that is sometimes described as multinomial Naïve Bayes. [86].

In Multi-Bernoulli probabilistic model, event is word presence or absence, let $D = (x_1, \ldots, x_{|V|})$, where $x_i = 1$ for presence of word w_i; and $x_i = 0$ for absence. We can compute $P(D|C)$ using the eq. 3.6

$$p(D = (x_1, \ldots, x_{|v|})|C) = \prod_{i=1}^{|v|} p(w_i = x_i|C) = \left[\prod_{i=1, x_i=1}^{|v|} p(w_i = 1|C)\right] \times \left[\prod_{i=1, x_i=0}^{|v|} p(w_i = 0|C)\right]$$

3.6

On the other hand, Multinomial (Language Model), event is word selection/sampling where $D = (n_1, \ldots, n_{|V|})$, n_i: frequency of word w_i $n = n_1, + \ldots + n_{|V}$. $P(D|C)$ is computed using the eq. 3.7.

$$p(D = (n_1, \ldots, n_{|v|})|C) = p(n|C)\binom{n}{n_1 \ldots n_{|V|}}\prod_{i=1}^{|V|} p(w_i|C)^{n_i} \qquad 3.7$$

Where $p(w_1|C) + \ldots p(w_{|v|}|C) = 1$

Parameter estimation for the Vocabulary: $V = \{w_1, \ldots, w_{|V|}\}$ as follows:

Category prior $$p(C_i) = \frac{|E(C_i)|}{\sum_{j=1}^{k} |E(C_j)|} \qquad 3.8$$

Where $E(C_i)$ is the expectation of the category C_i. Equations 3.9 and 3.10 presents Multi-Bernoulli document model and Multinomial doc model respectively

$$p(w_j = 1|C_i) = \frac{\sum_{d \in E(C_i)} \delta(w_j, d) + 0.5}{|E(C_i)| + 1} \quad \delta(w_j, d) = \begin{cases} 1 \text{ if } w_j \text{ occurs in } d \\ 0 \text{ otherwise} \end{cases} \qquad 3.9$$

$$p(w_j|C_i) = \frac{\sum_{d \in E(C_i)} c(w_j, d) + 1}{\sum_{m=1}^{|V|} \sum_{d \in E(C_i)} c(w_m, d) + |V|} \quad c(w_j, d) = \text{counts of } w_j \text{ in } d \qquad 3.10$$

Table 3.1 describes classification procedures for Multi-Bernoulli and Multinomial probabilistic models.

Table 3.1: Multi-Bernoulli and Multinomial probabilistic models classification procedures.

Multi-Bernoulli	**Multinomial**
$d = (x_1, ..., x_{\|V\|}) \quad x \in \{0,1\}$ $C^* = \arg\max_C P(D \mid C)P(C)$ $= \arg\max_C \prod_{i=1}^{\|V\|} p(w_i = x_i \mid C)P(C)$ $= \arg\max_C \log p(C) + \sum_{i=1}^{\|V\|} \log p(w_i = x_i \mid C)$	$d = (n_1, ..., n_{\|V\|}) \quad \| d \| = n = n_1 + ... + n_{\|V\|}$ $C^* = \arg\max_C P(D \mid C)P(C)$ $= \arg\max_C p(n \mid C) \prod_{i=1}^{\|V\|} p(w_i \mid C)^{n_i} P(C)$ $= \arg\max_C \log p(n \mid C) + \log p(C) + \sum_{i=1}^{\|V\|} n_i \log p(w_i \mid C)$ $\approx \arg\max_C \log p(C) + \sum_{i=1}^{\|V\|} n_i \log p(w_i \mid C)$

Suppose $n_1, n_2, \ldots, n_k$ is the number of times word i occurs in the document, and $P_1, P_2, \ldots, P_k$ is the probability of obtaining word i when sampling from all the documents in category H. Assume that the probability is independent of the word's context and position in the document. These assumptions lead to a multinomial distribution for document probabilities. For this distribution, the probability of a document D given its class C—in other words, the formula for computing the probability $P(D|C)$ in Bayes' rule is shown in eq. 3.11

$$P(D|C) \cong \mathrm{N!} \prod_{i=1}^{k} \frac{P_i^{n_i}}{n_i!} \qquad 3.11$$

Where $N = n_1 + n_2 + \ldots + n_k$ is the number of words in the document. The reason for the factorials is to account for the fact that the ordering of the occurrences of each word is immaterial according to the bag-of-words model. P_i is estimated by computing the relative frequency of word i in the text of all training documents pertaining to category C. In reality, there should be a further term that gives the probability that the model for category C generates a document whose length is the same as the length of D (that is why we use the symbol ≅ instead

of =), but it is common to assume that this is the same for all classes and hence can be dropped. [14, 88].

For example, suppose there are only the two words, "*yellow*" and "*blue*", in the vocabulary, and a particular document class *C* has *P(yellow|C)* = 75% and *P(blue|C)* = 25% (you might call *C* the class of yellowish green documents). Suppose *d* is the document "*blue yellow blue*" with a length of *N* = 3 words. There are four possible bags of three words. One is {blue yellow blue}, and its probability according to the preceding formula is

$$P(\{blue\ yellow\ blue\}|C) \approx 3! \times 0.75^1/1! \times 0.25^2/2! = 9/64 \approx 0.14$$

Thus its probability of being generated by the yellowish green document model is 9/64, or 14%. Suppose another class, very bluish green documents (call it C`) has *P(yellow|C`)*= 10%, *P(blue|C`)* = 90%. The probability that the evidence is generated by this model is 24%. [14].

$$P(\{blue\ yellow\ blue\}|C`) \approx 3! \times 0.1^1/1! \times 0.9^2/2! = 0.24$$

If these are the only two classes, does that mean that the evidence is in the very bluish green document class? Not necessarily. Bayes' rule, given earlier, says that you have to take into account the prior probability of each hypothesis. If we know that in fact very bluish green documents are twice as rare as yellowish green ones, this would be just sufficient to outweigh the preceding 14% to 24% disparity and tip the balance in favor of the yellowish green class. [63, 86].

The factorials in the preceding probability formula don't actually need to be computed because—being the same for every class—they drop out in the normalization process anyway. However, the formula still involves multiplying together many small probabilities, which yields

extremely small numbers that cause underflow on large documents. The problem can be avoided by using logarithms of the probabilities instead of the probabilities themselves. [63, 86].

In the multinomial maïve Bayes formulation a document's class is determined not just by the words that occur in it but also by the number of times they occur. In general it performs better than the ordinary Naïve Bayes model for document classification, particularly for large dictionary sizes. [63, 86]

3.3 Complement Naïve Bayes (*CNB*)

This approach use simple, heuristic solutions to some of the problems with naïve Bayes classifiers. The approach addresses both systemic issues as well as problems that arise because text is not actually generated according to a multinomial model [72].

One systemic problem is that when one class has more training examples than another, naïve Bayes selects poor weights for the decision boundary. This is due to an under-studied bias effect that shrinks weights for classes with few training examples. To balance the amount of training examples used per estimate, a "complement class" formulation of naïve Bayes was introduced by Rennie et. al. [72].

Another systemic problem with naïve Bayes is that features are assumed to be independent. As a result, even when words are dependent, each word contributes evidence individually. Thus the magnitude of the weights for classes with strong word dependencies is larger than for classes with weak word dependencies. To keep classes with more dependencies from dominating, the approach normalizes the classification weights [72].

In addition to systemic problems, multinomial naïve Bayes does not model text well. Presenting a simple transform enables naïve Bayes to instead emulate a power law distribution

that matches real term frequency distributions more closely. Rennie et. al. [72] discussed two other pre-processing steps, common for information retrieval but not for naïve Bayes classification, that incorporate real world knowledge of text documents (*TF* transformation, *IDF* transformation and normalization). They significantly boost classification accuracy. The improved classification accuracy is worthwhile. Complement Naïve Bayes (*CNB*) classifier made simple corrections to *NB* and it approaches the accuracy of the Support Vector Machines (*SVMs*) while being faster and easier to implement than the *SVMs* and most modern-day classifiers [72]. Figure 3.1 shows the *CNB* algorithm steps [72].

- Let $(\vec{d_1}, \ldots, \vec{d_n})$ be a set of documents; d_{ij} is the count of word *i* in document j.
- Let $\vec{y} = (y_1, \ldots, y_n)$ be the labels.
- $CNB(\vec{d}, \vec{y})$
  1. $d_{ij} = log\,(d_{ij} + 1)$ (*TF* transform)
  2. $d_{ij} = d_{ij}\, log \frac{\sum_k 1}{\sum_k \delta ik}$ (*IDF* transform)
  3. $d_{ij} = \frac{d_{ij}}{\sqrt{\sum_k (d_{kj})^2}}$ (length norm)
  4. $\hat{\theta}_{ci} = \frac{\sum_{j:yj \neq c} d_{ij} + \propto_i}{\sum_{i:yj \neq c} \sum_k d_{kj} + \propto}$ (complement)
  5. $\omega_{ci} = \log \hat{\theta}_{ci}$
  6. $\omega_{ci} = \frac{\omega_{ci}}{\sum_i \omega_{ci}}$ (weight normalization)
  7. Let $t = (t_1, \ldots, t_n)$ be a test document; let t_i be the document according to $l\,(t) = arg\ min_c \sum_i t_i \omega_{ci}$

Figure 3.1: Complement Naïve Bayes Algorithm [72]

3.4 Discriminative Multinomial Naïve Bayes (*DMNB*)

Learning Bayesian networks from data has two elements: structure learning and parameter learning. Given a fixed Bayesian network structure, parameters learning can take two different approaches: generative and discriminative learning. While generative parameter learning is more efficient, discriminative parameter learning is more effective. Discriminative Frequency Estimate *DFE* provides simple, efficient, and effective discriminative parameter learning method which learns parameters by discriminatively computing frequencies from data

[81]. Empirical studies of [81] show that the *DFE* algorithm integrates the advantages of both generative and discriminative learning. *DFE* performs as well as the state-of-the-art discriminative parameter learning method, gradient descent based parameter learning [90], in accuracy, but is significantly more efficient. The motivation is to turn the generative parameter learning method, Frequency Estimate *FE*, into a discriminative one by injecting a discriminative element into it. *DFE* discriminatively computes frequencies from data, and then estimates parameters based on the appropriate frequencies. The empirical studies show that *DFE* inherits the advantages of both generative and discriminative learning [81]. In the following, we shall describe frequency estimate, and discriminative frequency estimate.

3.4.1 Frequency Estimate

Let the capital letters X be a discrete random variable. The lower-case letters x is used for the value taken by variable X, and x_{ij} refers to the variable X_i taking on its j^{th} value. Let the boldface capital letters $\boldsymbol{X}$ be a set of variables, and the boldface lower case letters $\boldsymbol{x}$ for the values of variables in $\boldsymbol{X}$. The training data D consists of a set of finite number of training instances, and an instance e is represented by a vector (x, c), where c is the class label. In general, the symbol "hat" to indicate parameter estimates.

A Bayesian network encodes a joint probability distribution $P(X,C)$ by a set of local distributions P for each variable. By forcing the class variable C to be the parent of each variable X_i, we can compute the posterior probability $P(C_j|X)$ from eq. 3.12

$$P(C|X) = \propto P(C)\prod_{i=1}^{n} P(X_i|U_i) \qquad 3.12$$

Where α is a normalization factor, and U_i denotes the set of parents of variable X_i. Note that the class variable C is always one parent of X_i. In naïve Bayes, U_i only contains the class

variable C. $P(C)$ is called the prior probability and $P(X_i|U_i)$ is called the local probability distribution of X_i.

The local distribution $P(X_i|U_i)$ is usually represented by a conditional probability table (*CPT*), which enumerates all the conditional probabilities for each assignment of values to X_i and its parents U_i. Each conditional probability $P(x_{ij}|u_{ik})$ in a *CPT* is often estimated using the corresponding frequencies obtained from the training data as follows.

$$\hat{P}(x_{ij}|u_{ik}) = \frac{n_{ijk}}{n_{ik}} \qquad 3.13$$

Where n_{ijk} denotes the number of training instances in which variable X_i takes on the value x_{ij} and its parents U_i take on the values u_{ik}. n_{ik} is equal to the sum of n_{ijk} over all j. The prior probability $P(C)$ is also estimated in the same way. For the convenience in implementation, an entry θ_{ijk} in a *CPT* is the frequency n_{ijk}, instead of $P(x_{ij}|u_{ik})$, which can be easily converted to $P(x_i|u_i)$. To compute the frequencies from a given training data set, we go through each training instance, and increase the corresponding entries θ_{ijk} in *CPTs* by 1. By scanning the training data set once, we can obtain all the required frequencies and then compute the corresponding conditional probabilities. This parameter learning method is called Frequency Estimate (*FE*) [81].

3.4.2 Discriminative Frequency Estimate

Discriminative Frequency Estimate (*DFE*) is a discriminative parameter learning algorithm for Bayesian network classifiers. When counting a training instance in *FE*, simply increase the corresponding frequencies by 1. Consequently, we do not directly take the effect on classification into account in computing frequencies. In fact, at any step in this process, we

actually have a classifier on hand: the classifier whose local probabilities are computed by eq. 3.9 using the current entries (frequencies) in *CPTs* [81].

Thus, when counting an instance, we can apply the current classifier to it, and then update the corresponding entries based on how well (bad) the current classifier predicts on the instance. Intuitively, if the instance can be classified perfectly, there is no need to change any entries. In general, given an instance e, we can compute the difference between the true probability *P(c|e)* and the predicted probability $\hat{P}(c|e)$ generated by the current parameters, where *c* is the true class of *e*, and then update the corresponding entries based on the difference. Furthermore, the *FE* process can be generalized such that we can count each instance more than once (as many as needed) until a convergence occurs. This is the basic idea of *DFE*. More precisely, the *DFE* parameter learning algorithm iterates through the training instances. For each instance *e*, *DFE* firstly computes the predicted probability $\hat{P}(c|e)$, and then updates the frequencies in corresponding *CPTs* using the difference between the true *P(c|e)* and the predicted $\hat{P}(c|e)$. The detail of the algorithm is described in Figure 3.2. Here *M* is a pre-defined maximum number of steps [81]. *L(e)* is the prediction loss for training instance *e* based on the current parameters θ^t, defined as follows.

$$L(e) = P(c|e) - \hat{P}(c|e) \qquad 3.14$$

In general, *P(c|e)* are difficult to know in classification task, because the information we have for *c* is only the class label. Thus, we assume that *P(c|e)* = 1 when *e* is in class *c* in the implementations. Note this assumption may not be held if data cannot be separated completely, and thus may introduce bias to our probability estimation.

In summary, *DFE* learns parameters by considering the likelihood information $P(x_{ij}|u_{ik})$ and the prediction error $P(c|e) - \hat{P}(c|e)$, and thus can be considered as a combination of generative and discriminative learning. Moreover, the likelihood information $P(x_{ij}|u_{ik})$ seems to be more important than $P(c|e) - \hat{P}(c|e)$. For example, a *DFE* algorithm without eq. 3.13 performs significantly worse than naïve Bayes, while a *DFE* algorithm without eq. 3.14 can still learn a traditional naïve Bayes [81].

1. Initialize each *CPT* entry θ_{ijk} to 0
2. **For** *t* from 1 to *M* **Do**
   - Randomly draw a training instance *e* from the training data set *D*.
   - Compute the posterior probability $\hat{P}(c|e)$ using the current parameters θ^t And Equation 3.9.
   - Compute the loss *L(e)* using Equation 3.10.
   - **For** each corresponding frequency θ_{ijk} in *CPTs*

$$- Let\ \theta_{ijk}^{t+1} = \theta_{ijk}^{t} + L(e)$$

Figure 3.2: Discriminative Frequency Estimate [81]

3.5 *K*-Nearest Neighbors (*KNN*)

K Nearest Neighbors algorithm (*KNN*) is a method for classifying objects based on closest training examples in the feature space. *KNN* is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. *KNN* is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbor [30, 28, 46, 86].

Nearest neighbor rules in effect compute the decision boundary in an implicit manner as shown in Figure 3.3. *KNN* Directly estimates the a posteriori probabilities *P(C|X)*, i.e. bypass probability estimation and go directly to decision functions. *KNN* can center a cell about *x* and let it grows until it captures k_n samples. A potential remedy for the problem of the unknown "best"

window function in Parzen Window technique is to let the cell volume be a function of the training data [28].

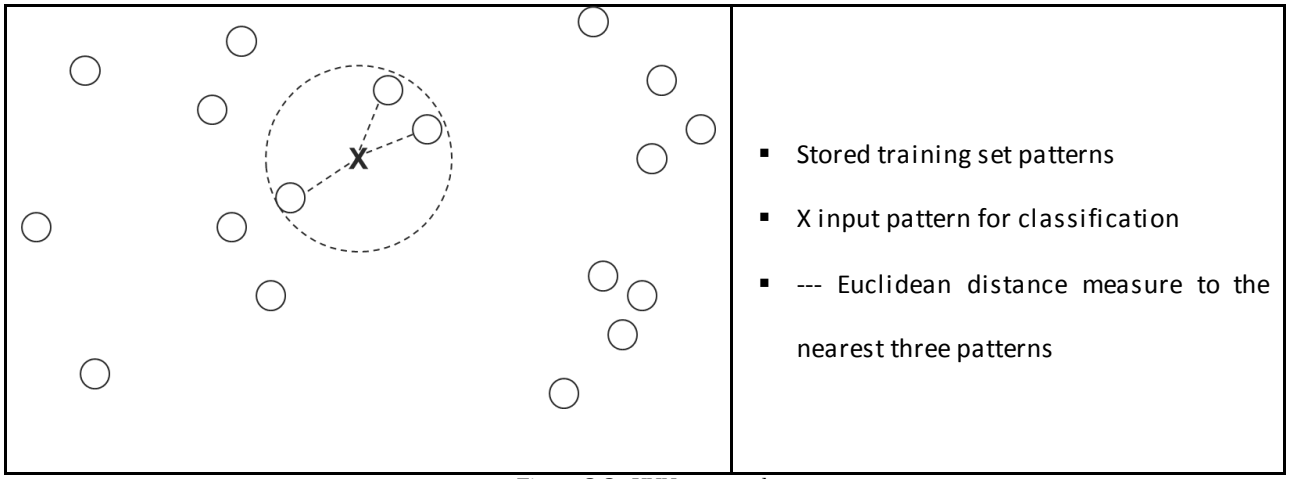

Figure 3.3: *KNN* approach

In the classification phase, an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point [30, 28, 46, 86]. *KNN* classifier is based on a distance function for pairs of observations, such as the *Euclidean* or *Cosine* distance functions [30, 28, 46, 86].

The best choice of k depends upon the data; generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. Figure 3.4 shows that the test sample (circle) should be classified either to the first class of squares or to the second class of triangles. If $k = 3$ it is classified to the second class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$ it is classified to first class (3 squares vs. 2 triangles inside the outer circle). A good k can be selected by various heuristic techniques, for example, cross-validation. The special case where the class is predicted to be the class of the closest training sample (i.e. when $k = 1$) is called the nearest neighbor algorithm [30, 28, 46, 86]. *KNN* algorithm steps are presented in Figure 3.5.

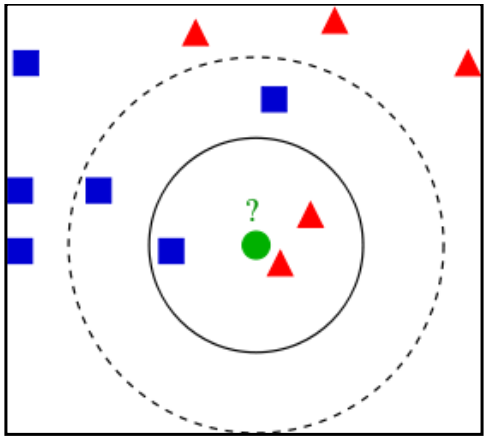

Figure 3.4: Example of *KNN* classification.

One of the advantages of *KNN* is that it is well suited for multi-modal classes as its classification decision is based on a small neighborhood of similar objects (i.e., the major class). So, even if the target class is multi-modal (i.e., consists of objects whose independent variables have different characteristics for different subsets), it can still lead to good accuracy.

A major drawback of the similarity measure used in *KNN* is that it uses all features equally in computing similarities. This can lead to poor similarity measures and classification errors, when only a small subset of the features is useful for classification [30, 28, 46, 86]. Another drawback to "majority voting" of *KNN* is that the classes with the more frequent examples tend to dominate the prediction of the new vector, as they tend to come up in the *k* nearest neighbors when the neighbors are computed due to their large number [30, 28, 46, 86].

KNN becomes a standard within the field of text categorization and is included in numerous experiments as a basis for comparison. It has been in use since the early stages of *TC* research, and is one of the best performing methods within the field [62, 78].

```
Input:
        D  // training data
        K  // number of neighbors
        t  // input tuple to classify
output:
```

```
        c  // class to which t is assigned
KNN algorithm:
        // algorithm to classify a tuple using KNN
        N = φ
        foreach d ∈ D do
        if |N| ≤ k
                N = N ∪ d
        Else
                If ∃ u ∈ N such that sim(t,u) ≥ sim(t,d) then
                Begin
                        N = N – u;
                        N = N ∪ d
                End
C = class to which the most u ∈ N are classified
```

Figure 3.5: *KNN* algorithm

3.6 Support vector machines (*SVMs*)

A support vector machine (*SVMs*) is a set of related supervised learning methods used for classification and regression. In simple words, given a set of training examples, each marked as belonging to one of two categories, *SVMs* training algorithm builds a model that predicts whether a new example falls into one category or the other. Intuitively, *SVMs* model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on [28, 46, 86].

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier as shown in Figure 17 [28, 46, 86].

SVMs was derived from statistical learning theory by Vapnik, et al. in 1992 [28, 46, 86]. *SVMs* became famous when, using images as input, it gave accuracy comparable to neural-

network with hand-designed features in a handwriting recognition task. Currently, *SVMs* is widely used in object detection and recognition, content-based image retrieval, text recognition, biometrics, speech recognition, speaker identification, benchmarking time-series prediction tests. Using *SVMs* in text classification is proposed by [53], and subsequently used in [29, 84].

Eq. 3.15 is dot product formula and used for the output of linear *SVMs*, where x is a feature vector of classification documents composed of words. w is the weight of corresponding x. b is a bias parameter determined by training process.

$$y = w \cdot x - b \qquad 3.15$$

The following summarizes *SVMs* steps:

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

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 hyperplane (i.e., "decision boundary"). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. *SVMs* finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors). Figure 3.6 shows support vectors and how margins are maximized [28, 46, 86].

*SVM*s is effective on high dimensional data because the complexity of trained classifier is characterized by the number of support vectors rather than the dimensionality of the data, the support vectors are the essential or critical training examples, they lie closest to the decision boundary, If all other training examples are removed and the training is repeated, the same separating hyperplane would be found. The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the *SVM*s classifier, which is independent of the data dimensionality. Thus, an *SVM*s with a small number of support vectors can have good generalization, even when the dimensionality of the data is high [28, 42, 46, 86].

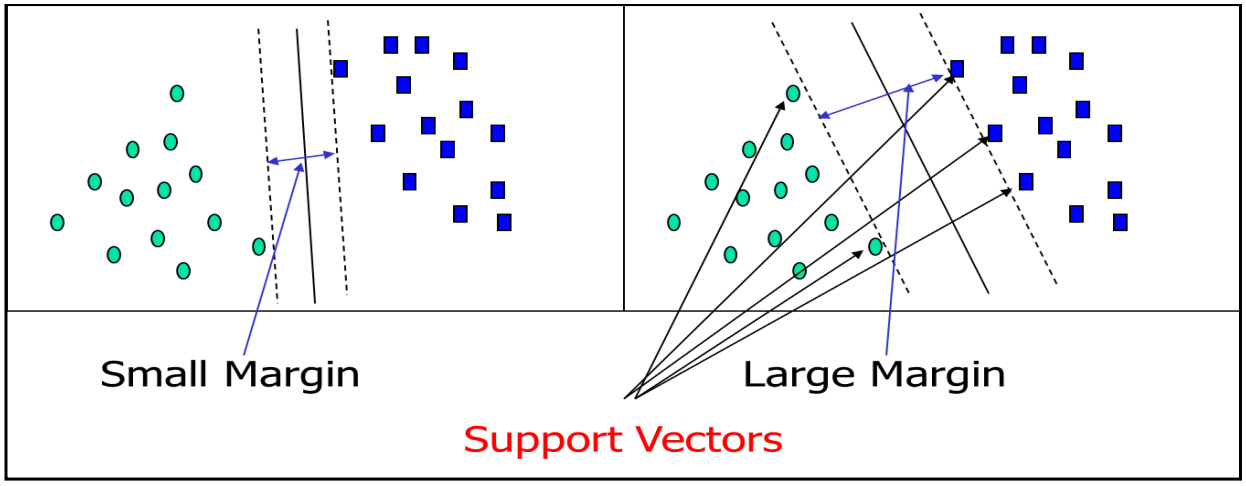

Figure 3.6: Support Vectors

3.7 *C4.5* Decision Tree

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan [71]. *C4.5* is an extension of Quinlan's earlier *ID3* algorithm. The decision trees generated by *C4.5* can be used for classification, and for this reason, *C4.5* is often referred to as a statistical classifier.

C4.5 builds decision trees from a set of training data in the same way as *ID3*, using the concept of information entropy. At each node of the tree, *C4.5* chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its

criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The *C4.5* algorithm then recurs on the smaller sub lists [28, 46, 71, 86].

Algorithm for decision tree induction constructs the tree in a top-down recursive divide-and-conquer manner. Below, summarizes algorithm steps [28, 46, 71, 86]:

- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- The algorithm stop partitioning in one of the following conditions:
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Information Gain is attribute selection measure for *ID3/C4.5*, it selects the attribute with the highest information gain. Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_i, D|/|D|$. Then, the expected information (entropy) needed to classify a tuple in D is [28, 46, 71, 86]:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \qquad 3.16$$

And the information needed (after using A to split D into v partitions) to classify D is:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j) \qquad 3.17$$

Information gained by branching on attribute A can be computed from eq. 3.18

$$Gain(A) = Info(D) - Info_A(D) \qquad 3.18$$

If the attribute A is a continuous-valued attribute, then we must determine the best split point for A by sorting the value A in increasing order. The midpoint between each pair of adjacent values is considered as a possible split point. D_1 is the set of tuples in D satisfying A ≤ split-point, and D_2 is the set of tuples in D satisfying A > split-point [28, 46, 71, 86].

Information gain measure is biased towards attributes with a large number of values. *C4.5* (a successor of *ID3*) uses gain ratio to overcome the problem (normalization to information gain). The gain ration of an attribute A can be computed from eq. 3.19 [28, 46, 71, 86].

$$\text{GainRatio(A)} = \text{Gain(A)/SplitInfo(A)} \qquad 3.19$$

where,

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right) \qquad 3.20$$

3.8 Summary

This chapter has described popular text classification algorithms that have been used in this research. The chose various algorithms: probabilistic algorithms (*NB*, *MNB*, *CNB*, *DMNB*), lazy or instance based algorithm *KNN*, partition based algorithms or decision trees (*C4.5*), and optimization method which tries to find a solution to a discriminant function (*SVMs*). These algorithms can be considered as de facto standard algorithms for text classifications.

We have not considered neural networks (*NN*) because it is scalable in high dimensional data [10, 11, 48]. *NN* has many parameters that must be determined by the user, it also takes very long time for training [10, 11, 48]. In addition, there is no guarantee for accurate results [10, 11, 48]. An association rules algorithm requires large memory space to find frequent patterns and it is not scalable for high dimensional datasets [17, 40].

Text preprocessing techniques will be discussed in the next chapter.

Chapter 4 : Text Preprocessing

This chapter describes text preprocessing, the important stage in *TC*. Text preprocessing includes many steps including feature reduction using morphological analysis techniques, and term weighting.

To reiterate, text mining can be summarized as a process of "numericizing" text. At the simplest level, all words found in the input documents will be indexed and counted in order to compute a table of documents and words, i.e., a matrix of frequencies that enumerates the number of times that each word occurs in each document [43, 49]. This basic process can be further refined to exclude certain common words such as "the" and "a" (stop word lists) and to combine different grammatical forms of the same words such as "traveling," "traveled," "travel," etc. [43, 49]. For Arabic, example of stopwords are (من الى فوق), and examples of different grammatical forms of same word in Arabic is (المسافرون المسافرين).

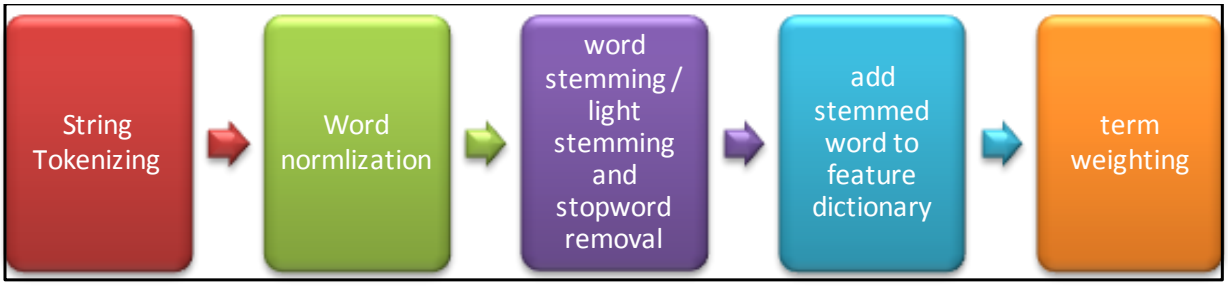

Figure 4.1: Structuring text data process

However, once a table of (unique) words (terms) by documents has been derived, all standard statistical and data mining techniques can be applied to derive dimensions or clusters of words or documents, or to identify "important" words or terms that best predict another outcome variable of interest [43, 49]. The process of structuring text data is depicted in Figure 4.1, the process includes tokenizing string to words, normalizing tokenized words, applying stopwords

removal, stemming / light stemming, and finally term weighting [43, 49]. Stemming / light stemming is optional step, and the resulting text data without stemming or light stemming is raw text [43, 49].

4.1 Issues and Considerations for "Numericizing" Text

4.1.1 Large numbers of small documents vs. small numbers of large documents

An example of using large numbers of small or moderate sized documents is a large number of short news or an active mailing list containing a large number of small posts. On the other hand, if we need to extract "concepts" from only a few documents that are very large (e.g., two lengthy books), then statistical analyses are generally less powerful because the "number of cases" (documents) in this case is very small while the "number of variables" (extracted words) is very large [43, 49].

Small number of large documents relatively generates more features compared as the features generated by large number of small documents. The reason is that words usually have frequent occurrences at documents level. Large number of small documents usually better in term of classification accuracy than small number of large documents. The reason is that large number of small documents has fewer features with more training example.

4.1.2 Excluding certain characters, short words, numbers

Excluding numbers, certain characters, or sequences of characters, or words that are shorter or longer than a certain number of letters can be done before the indexing of the input documents starts. The benefit of excluding them is that these characters, words, and numbers do not help determining the document topic. We may also want to exclude "rare words" which be defined as words that occur in a low percentage of the processed documents [43, 49].

4.1.3 Exclude lists of stop-words

Stop words is the name given to words which are filtered out prior to, or after, processing of natural language data (text). There is no definite list of stop words which all Natural language processing (*NLP*) tools incorporate. Not all *NLP* tools use a stoplist. Some tools specifically avoid using them to support phrase searching [43, 49].

"stop-words," i.e., terms that are to be excluded from the indexing can be defined. Typically, a default list of English stop words includes "the", "a", "of", "since", etc., i.e., words that are used in the respective language very frequently, but communicate very little unique information about the contents of the document.

For Arabic, stopwords list includes punctuations (? ! …), pronouns (... هما التي الذي هي هو), adverbs (... بين تحت فوق), days of week (... الاثنين الاحد السبت), month of year (... مارس فبراير يناير). Stopwords list are removed because they do not help determining document topic and to reduce features.

4.1.4 Stemming algorithms

In linguistic morphology, stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Word stemming is an important pre-processing step before indexing of input documents begins. For example, different grammatical forms or declinations of verbs are identified and indexed (counted) as the same word. For example, stemming will ensure that both "المسافرون", "المسافرين", and "مسافر" will be recognized by the text mining program as the same word [43, 49].

Synonyms and phrases must be considered, synonyms such as "ليث" or "اسد", or words that are used in particular phrases where they denote unique meaning can be combined for indexing [43, 49].

Stemming, synonyms, the letters that are permitted in words, etc. are highly language dependent operations. Therefore, support for different languages is very important [43, 49].

4.2 Vector Space Model (*VSM*) and Term Weighting Schemes

We described in section 1.1 the process of structuring text data to view text as a bag-of-tokens (words) and compute co-occurrence stats over free text. The aim of term weighting is to enhance text document representation as feature vector or vector space model (*VSM*). Popular term weighting schemes are Boolean model (which indicates absence or presence of a word with Booleans 0 or 1 respectively), word count (*wc*), normalized word count, term pruning, Term Frequency (*tf*), and Term Frequency-Inverse Document Frequency (*tf-idf*). Term frequency *tf(t, d)* is the number that the term *t* occurred in the document *d*. Document frequency *df(t)* is number of documents in which the term *t* occur at least once. The inverse document frequency can be calculated from document frequency using the formula *log(num of Docs/num of Docs with word i)*. The inverse document frequency of a term is low if it occurs in many documents and high if the term occurs in only few documents. Term discrimination consideration suggests that the best terms for document content identification are those able to distinguish certain individual documents from the collection. This implies that the best terms should have high term frequencies but low overall collection frequencies (*num of Docs with word i*). A reasonable measure of term importance may then be obtained by using the product of the term frequency and the inverse document frequency (*tf* * *idf*) [52, 73, 74].

In many situations, short documents tend to be represented by short-term vectors, whereas much larger-term sets are assigned to the longer documents. Normally, all text documents should have the same importance for text mining purposes. This suggests that a normalization factor to be incorporated into the term-weighting to equalize the length of the document vectors [52, 73, 74].

4.2.1 Term weighting equations

The term count in a given document is simply the number of times a given term appears in that document. This count is usually normalized to prevent a bias towards longer documents (which may have a higher term count regardless of the actual importance of that term in the document) to give a measure of the importance of the term t_i within the particular document d_j. Thus we have the term frequency, defined as follows [43, 49]

$$tf_{i,j} = \frac{n_{i,j}}{\sum_n n_{k,j}} \qquad 4.1$$

Where $n_{i,j}$ is the number of occurrences of the considered term (t_i) in document d_j, and the denominator is the sum of number of occurrences of all terms in document d_j. A variation of *tf* is to apply *log* transformation to term frequency (eq. 4.2) [43, 49].

$$Term\ Frequency\ Transformation = Log(1 + tf_{i,j}) \qquad 4.2$$

The inverse document frequency is a measure of the general importance of the term (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient).

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \qquad 4.3$$

Where | D | is the total number of documents in the corpus and $|\{d: t_i \in d\}|$ is number of documents where the term t_i appears (that is $n_{ij} \neq 1$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to use $1+|\{d: t_i \in d\}|$ [43, 49]

Then

$$(tf - idf)_{i,j} = tf_{i,j} \times idf_i \qquad 4.4$$

A high weight in *tf-idf* is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. The *tf-idf* value for a term will always be greater than or equal to zero [43, 49].

The *tf–idf* weight (term frequency–inverse document frequency) is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the *tf–idf* weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query [43, 49]. One of the simplest ranking functions is computed by summing the *tf-idf* for each query term; many more sophisticated ranking functions are variants of this simple model [43, 49].

Suppose we have a set of Arabic text documents and wish to determine which document is most relevant to the query "قال انها بقرة صفراء" ("he said it is yellow cow") A simple way to start out is by eliminating documents that do not contain all four words "قال" و"انها", "بقرة", and "صفراء" but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document and sum them all together; the number of

times a term occurs in a document is called term frequency. However, because the terms "قال" and "انها" are so common, this will tend to incorrectly emphasize documents which happen to use the words "قال" and "انها" more, without giving enough weight to the more meaningful terms "بقرة" and "صفراء". The terms "قال" and "انها" are not a good keyword to distinguish relevant and non-relevant documents and terms like "بقرة" and "صفراء" that occur rarely are good keywords to distinguish relevant documents from the non-relevant documents. Hence an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the collection and increases the weight of terms that occur rarely [43, 49].

tf-idf Example

Consider a document containing 100 words wherein the word "بقرة" appears 3 times. Following the previously defined formulas, the term frequency (*tf*) for cow is then 0.03 (3 / 100). Now, assume we have 10 million documents and "بقرة" appears in one thousands of these. Then, the inverse document frequency is calculated as $log(10\ 000\ 000 / 1\ 000) = 4$. The *tf-idf* score is the product of these quantities: $0.03 \times 4 = 0.12$.

The *tf-idf* weighting scheme is often used in the vector space model together with *cosine* similarity to determine the similarity between two documents.

4.3 Morphological Analysis (Stemming and light stemming)

In linguistics, morphology is the identification, analysis and description of the structure of morphemes and other units of meaning in a language like words, affixes, and parts of speech and intonation/stress, implied context (words in a lexicon are the subject matter of lexicology). Morphological typology represents a way of classifying languages according to the ways by which morphemes are used in a language —from the analytic that use only isolated morphemes, through the agglutinative ("stuck-together") and fusional languages that use bound morphemes

(affixes), up to the polysynthetic, which compress lots of separate morphemes into single words [43, 49].

While words are generally accepted as being (with clitics) the smallest units of syntax, it is clear that in most (if not all) languages, words can be related to other words by rules (grammars). For example, English speakers recognize that the words dog and dogs are closely related — differentiated only by the plurality morpheme "-s," which is only found bound to nouns, and is never separate. Speakers of English (a fusional language) recognize these relations from their tacit knowledge of the rules of word formation in English. They infer intuitively that dog is to dogs as cat is to cats; similarly, dog is to dog catcher as dish is to dishwasher (in one sense). The rules understood by the speaker reflect specific patterns (or regularities) in the way words are formed from smaller units and how those smaller units interact in speech. In this way, morphology is the branch of linguistics that studies patterns of word formation within and across languages, and attempts to formulate rules that model the knowledge of the speakers of those languages [43, 49].

Terms have many morphological variants (as described in section 1.2.2.5) that will not be recognized by term matching algorithm without additional text processing. Stemming algorithms are needed in many applications such as natural language processing, compression of data, and information retrieval systems. In most cases, these variants have similar semantic interpretation and can be treated as equivalence in text mining. Stemming algorithm can be employed to perform term reduction to a root form [43, 49].

In general, most of Arabic morphological tools face a problem with diacritics because most of them remove (normalize) diacritics. For example, the Arabic word (ذَهَبَ) which means (went) has identical form (without diacritics) to word (ذَهَبْ) which means gold. Diacritics

distinguish between them, but unfortunately, most of Arabic morphological tools remove them as a first step [43, 49].

For Arabic Language, there are two different morphological analysis techniques; stemming and light stemming. Stemming reduces words to their stems [56]. Light stemming, in contrast, removes common affixes from words without reducing them to their stems. Stemming would reduce the Arabic words (الكتاب الكاتب المكتبة) which mean (the library), (the writer), and (the book) respectively, to one stem (كتب), which means (write).

The main idea for using light stemming [31, 32] is that many word variants do not have similar meanings or semantics. However, these word variants are generated from the same root. Thus, root extraction algorithms affect the meanings of words. Light stemming aims to enhance the classification performance while retaining the words 'meanings. It removes some defined prefixes and suffixes from the word instead of extracting the original root [31, 32]. Formally speaking, the aforementioned Arabic words (الكتاب الكاتب المكتبة) which mean (the library), (the writer), and (the book) respectively, belong to one stem (كتب) despite they have different meanings. Thus, the stemming approach reduces their semantics. The light stemming approach, on the other hand, maps the word (الكتاب) which means (the book) to (كتاب) which means (book), and stems the word (الكاتب) which means (the writers) to (كاتب) which means (writer). Another example for light stemming is the words (المسافرون المسافرين) which mapped to word (مسافر). Light stemming keeps the words' meanings unaffected. We previously described in section 1.3 that there are many words morphology have different meaning despite they have the same root. Figure 4.2 shows the steps of Arabic light stemming. Arabic light stemmer from Apache Lucene [58] is standard Arabic light stemmer.

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

Figure 4.2: Arabic Light Stemming Algorithm Steps

Stemming algorithm by Khoja [56] one is of well know Arabic Stemmers. Khoja's stemmer removes the longest suffix and the longest prefix. It then matches the remaining word with verbal and noun patterns, to extract the root. The stemmer makes use of several linguistic data files such as a list of all diacritic characters, punctuation characters, definite articles, and stopwords.

However, the Khoja stemmer has several weaknesses [83]. First, the root dictionary requires maintenance to guarantee newly discovered words are correctly stemmed. Second, the Khoja stemmer replaces a weak letter with (و) which occasionally produces a root that is not related to the original word. For example, the word (منظمات) which mean (organizations) is stemmed to (ظمآ) which means (he was thirsty) instead of (نظم). Here the Khoja stemmer removed a part of the root when it removed the prefix and then added a *hamza* at the end. Third, by following a certain order of affixes, the Khoja stemmer will in some cases fail to remove all of them. For example, the terms (تستغرق) and (ركبتيه) are not stemmed although they are respectively derived from the two regular roots (غرق) and (ركب). Algorithm steps of Khoja Arabic stemmer is described in Figure 4.3.

Al-Shalabi, Kanaan and Al-Serhan [14] developed a root extraction algorithm (tri-literal root extraction) which does not use any dictionary. It depends on assigning weights for a word's

letters multiplied by the letter's position, Consonants were assigned a weight of zero and different weights were assigned to the letters grouped in the word (سألتمونيها) where all affixes are formed by combinations of these letters. The algorithm selects the letters with the lowest weights as root letters.

1. Remove diacritics
2. Remove stopwords, punctuation, and numbers.
3. Remove definite article (ال)
4. Remove inseparable conjunction (و)
5. Remove suffixes
6. Remove prefixes
7. Match result against a list of patterns.
   - If a match is found, extract the characters in the pattern representing the root.
   - Match the extracted root against a list known "valid" roots
8. Replace weak latters واي with و
9. Replace all occurrences of *Hamza* ئ ء ؤ with ا
10. Two letter roots are checked to see if they should contain a double character. If so, the character is added to the root.

Figure 4.3: Arabic Stemming Algorithm Steps

Sawalhi and Atwell [77] evaluated Arabic Language Morphological Analyzers and Stemmers. Authors reported Khoja stemmer achieved the highest accuracy then the tri-literal root extraction algorithm. The majorities of words have a tri-lateral root, in fact between 80 and 85% of words in Arabic are derived from tri-lateral roots [8, 36]. The rest have a quad-letter root, penta-letter root or hexa-letter root. Khoja stemmer works accurately for tri-literal roots, this why it achieved the highest accuracy. Sawalhi and Atwell also reported that most stemming algorithms are designed for information retrieval systems where accuracy of the stemmers is not important issue. On the other hand, accuracy is vital for natural language processing. The accuracy rates show that the best algorithm failed to achieve accuracy rate of more than 75%. This proves that more research is required. We cannot rely on such stemming algorithms for doing further research as Part-of-Speech tagging and then Parsing because errors from the stemming algorithms will propagate to such systems [77].

4.4 Text Preprocessing tools

We use WEKA (Waikato Environment for Knowledge Analysis) for text preprocessing and classification. WEKA [45] is a popular suite of machine learning software written in Java, developed at the University of Waikato. It is free software available under the *GNU* General Public License. WEKA provides a large collection of machine learning algorithms for data pre-processing, classification, clustering, association rules, and visualization, which can be invoked through a common Graphical User Interface. Using WEKA *StringToWordVector* tool options with different combinations, we setup the term weighting combinations presented in Table 4.1 to structure text data. Major combinations include Boolean, word count, *tf*, *tdf*, *tf-idf*, term pruning, and word count normalization options, these combinations have not been applied in the literature on Arabic text before. The resulting combinations are listed in Table 4.2.

Table 4.1: Weka String to Word Vector options

TF Transform	*log(1+fij),* where *fij* is the frequency of word *i* in document *dj*.
IDF Transform	*fij*log(num of Docs/num of Docs with word i)*, where *fij* is the frequency of word *i* in document *dj*.
TFIDF Transformation	*log(1 + fij) * log (num of Docs/num of Docs with word i)*, where *fij* is the frequency of word *i* in document *dj*.
minTermFreq	Sets the minimum term frequency (apply term pruning)
normalizeDocLength	Sets whether if the word frequencies for a document should be normalized or not.
outputWordCounts	Output word counts rather than Boolean 0 or 1(indicating absence or presence of a word).
Stemmer	The stemming algorithm to be use on the words (Khoja Arabic Stemmer Algorithm).

Seven text classification algorithms (*C4.5* Decision Tree (*C4.5 DT*), *K* Nearest Neighbors (*KNN*), Support Vector Machine (*SVMs*), Naïve Bayes (*NB*), Naïve Bayes Multinomial (*NBM*), Complement Naïve Bayes (*CNB*), and Discriminative Multinomial Naïve Bayes classifier (*DMNBtext))* are applied to classify text documents. Experimental results presented in chapter 6.

We implement and integrate Arabic stemming and light stemming algorithms, described in Figures 17, and 18 respectively, into Weka. We adopt Arabic stopwords list from [20] for stopwords removal. The complete package of integration is available publically at [68]. A

screenshot of Arabic stemmer / light stemmer integrated to Weka is depicted in Figure 4.4. We also developed a Java program that uses Weka libraries to preprocess text documents to produce different weight combinations mentioned in Table 4.2.

Table 4.2: Symbols used in experiment setup preprocessing combinations

Boolean	Indicating presence (1) or absence (0) of a word.
wc	Output word counts
wc-tf	Apply *TF* transformation on word count
wc-tf-idf	Apply *TFIDF* transformation on word count
wc-norm	Apply document normalization on word count
wc-minFreq3	Apply term pruning on word count that less than 3
wc-norm-minFreq3	Apply normalization and term pruning on word count that less than 3
wc-tfidf-norm-minFreq3	Apply *TFIDF* and normalization on word count that less than 3
wc-norm-minFreq5	Apply normalization and term pruning on word count that less than 5
wc-tfidf-norm-minFreq5	Apply *TFIDF* and normalization on word count that less than 5

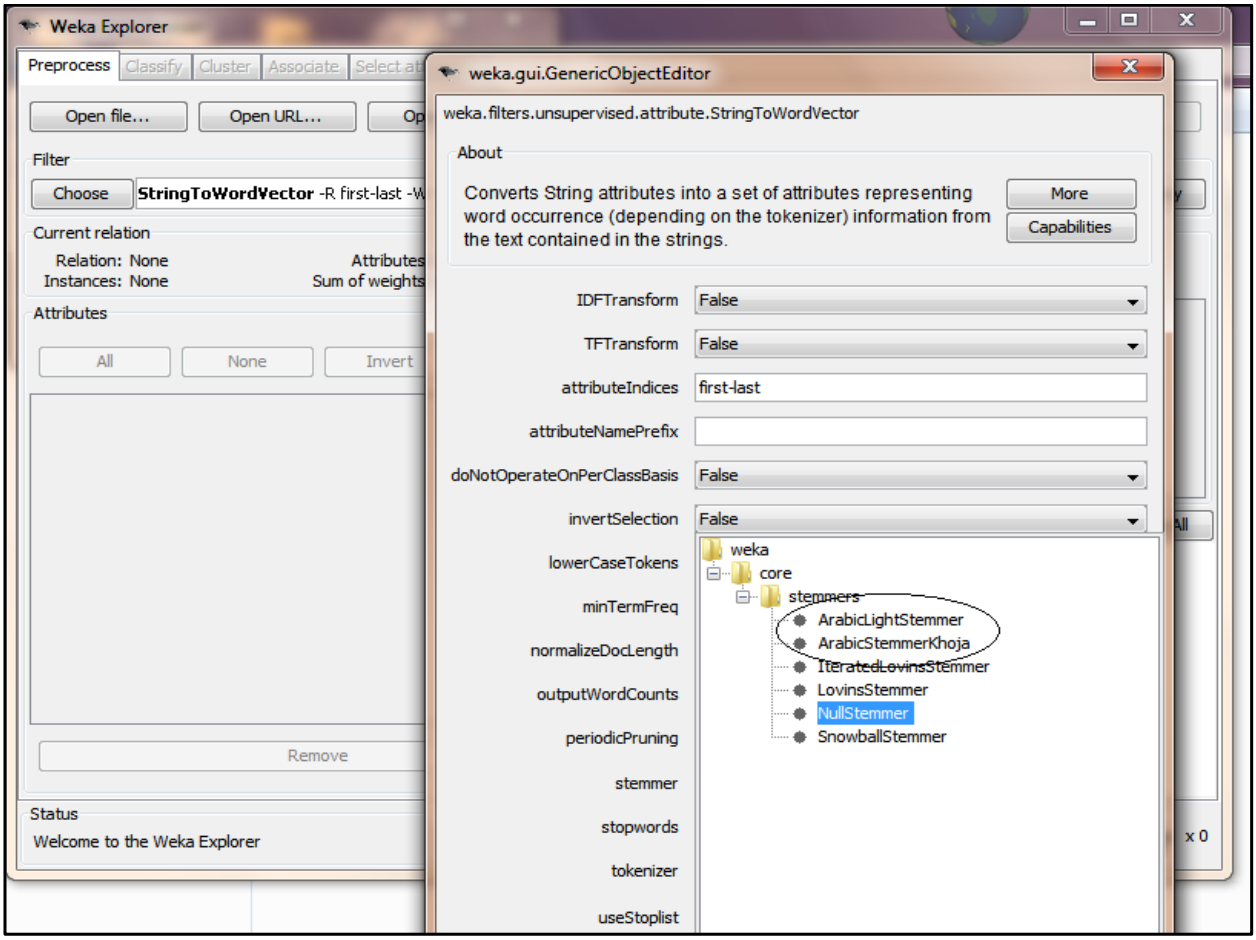

Figure 4.4: Weka Arabic Stemmers

RapidMiner (formerly YALE (Yet Another Learning Environment)) is an environment for machine learning and data mining experiments. It allows experiments to be made up of a large number of arbitrarily nestable operators. Operators are described in XML files which are created with RapidMiner' s graphical user interface. RapidMiner is used for both research and real-world data mining tasks [67]. RapidMiner provides more than 1,000 operators for all main machine learning procedures, including input and output, and data preprocessing and visualization. It is written in the Java programming language and therefore can work on all popular operating systems. It also integrates learning schemes and attributes evaluators of the Weka learning environment [67]. "Process Documents from files" is an RapidMiner operator that Generates word vectors from a text collection stored in multiple files. It also provides different term weighting schemes, and term pruning options as presented in Table 4.3.

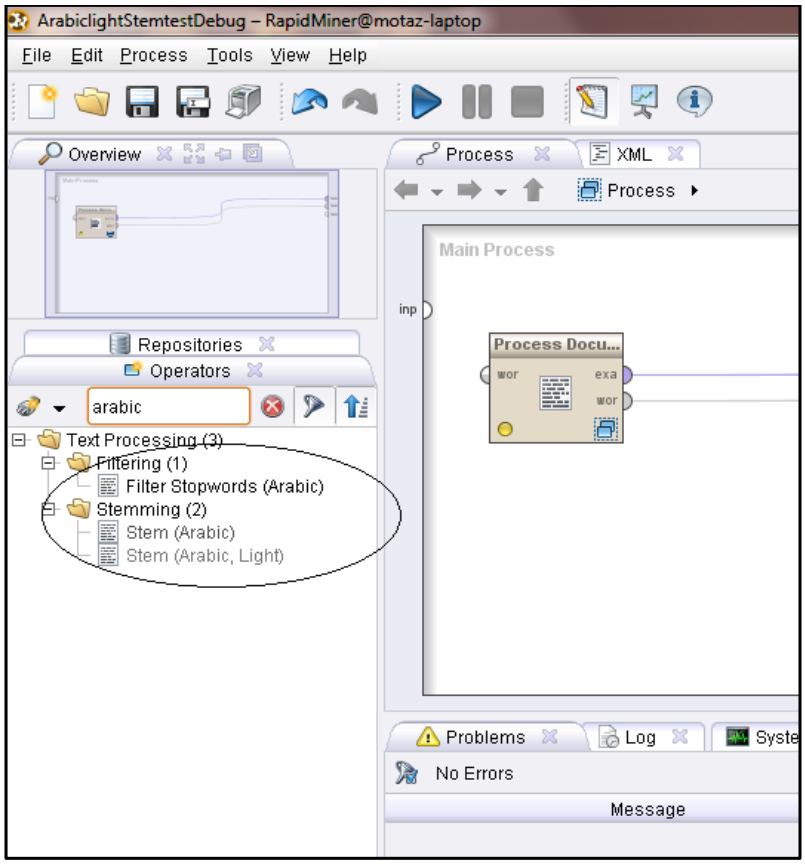

Figure 4.5: RapidMiner Arabic Stemmer Operators

Table 4.3: RapidMiner "Process document from files" operator options

Weight schemes	Equation	Description
TFIDF	$\frac{f_{ij}}{f_{d_j}} \log(\frac{\|D\|}{ft_i})$	Where |D| is the total number of documents. The resulting vector for each document is normalized.
TermFrequency	$\frac{f_{ij}}{f_{d_j}}$	The relative frequency of a term in a document. The resulting vector for each document is normalized.
TermOccurrences	f_{ij}	The absolute number of occurrences of a term. The resulting vector for each document is normalized.
BinaryOccurrences	1 if $f_{ij} > 0$ 0 otherwise	Count Occurrences as a binary value. The resulting vector for each document is not normalized.

We implement and contribute 3 operators to RapidMiner text plugin; Arabic Stemmer, Arabic Light Stemmer, and Arabic stopwords removal operator. The contribution is available publically within text processing RapidMiner plugin. Figure 4.5 shows a screenshot of the three operators. Figure 4.6 shows the process of transforming text documents to record using RapidMiner. Figure 4.7 shows the resulting wordlist (dictionary).

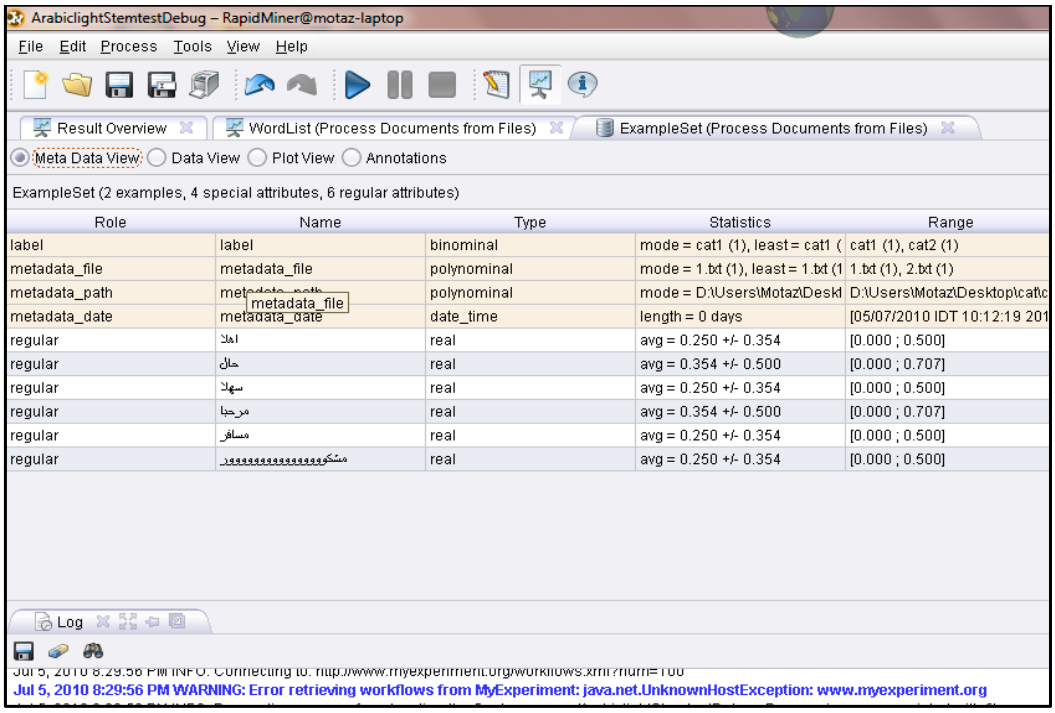

Figure 4.6: Transforming text documents to Example Set using RapidMiner

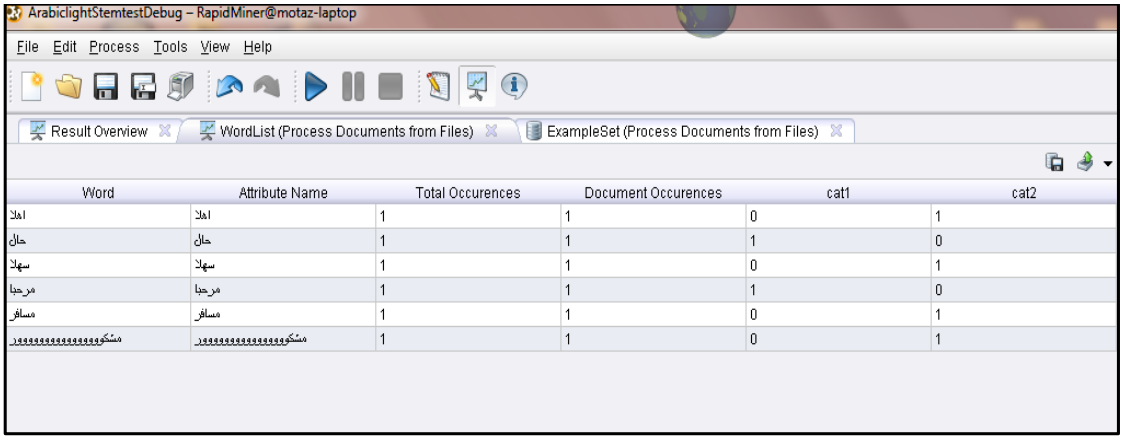

Figure 4.7: Transforming text documents to word list using RapidMiner

We implemented and integrated Arabic stemmers and light stemmers to WEKA and RapidMiner because both of them are the leading open source machine learning and data mining tools and to provide Arabic support in commonly used data mining / machine learning tools. Furthermore, there are some differences in preprocessing options in the tools. For example, RapidMiner provides various distance functions and term pruning methods while WEKA provides more detailed term weighting options.

Boyed this chapter, we shall cover the corpora that we used in this research in the next chapter.

Chapter 5 : Corpora

As mention in section 1.2.2, one of the difficulties that encounter this work and other researches in the field of Arabic linguistics was the lack of publicly available Arabic corpus for evaluating text categorization algorithms. Different training data sets are available for text classification in English. Reuter's collections[4] of news stories are popular and typical example. The Linguistic Data Consortium[5] (*LDC*) provides two non-free Arabic corpora, the Arabic NEWSWIRE and Arabic Gigaword corpus. Both corpora contain newswire stories. One of the aims of this research is to compile representative training datasets for Arabic text classification that cover different text genres which can be used in this research and in the future as a benchmark. Therefore, three different datasets were compiled covering different genres and subject domains.

There is a need for a freely-accessible corpus of Arabic. There are no standard or benchmark corpora. All researchers conduct their researches on their own compiled corpus. Arabic language is highly inflectional and derivational language which makes text mining a complex task. In Arabic *TC* research field, there are some published experimental results, but these results came from different datasets, it is hard to compare classifiers because each research used different datasets for training and testing [4, 5, 6, 7, 8, 9, 10, 11, 13, 33, 38, 44, 54, 56, 69, 73, 83, 98, 101]. Sebastiani stated at [78] "We have to bear in mind that comparisons are reliable only when based on experiments performed by the same author under carefully controlled

[4] *http://www.daviddlewis.com/resources/testcollections/*
[5] *http://www.ldc.upenn.ed/*

conditions". Tables 5.1, 5.2, and 5.3 present the existing non-free, free, and under development Arabic Corpora, respectively[6].

Table 5.1: Non-free Arabic corpora

Name of Corpus	*Source*	*Medium*	*Size*	*Purpose*	*Material*
Buckwalter Arabic Corpus 1986-2003 [32]	Tim Buckwalter	Written	2.5 to 3 billion words	Lexicography	Public resources on the Web
Leuven Corpus (1990-2004) [33]	Catholic University Leuven, Belgium	Written and spoken	3M words (spoken: 700,000)	Arabic-Dutch /Dutch-Arabic learner's dictionary	Internet sources, radio & TV, primary school books
Arabic Newswire Corpus (1994) [34]	University of Pennsylvania LDC	Written	80M words	Education and the development of technology	Agence France Presse, Xinhua News Agency, and Umma Press
CALLFRIEND Corpus (1995) [35]	University of Pennsylvania LDC	Conversational	60 telephone conversations	Development of language identification technology	Egyptian native speakers
NijmegenCorpus (1996) [36]	Nijmegen University	Written	Over 2M words	Arabic-Dutch / Dutch-Arabic dictionary	Magazines and fiction
CALLHOME Corpus (1997) [37]	University of Pennsylvania LDC	Conversational	120 telephone conversations	Speech recognition produced from telephone lines	Egyptian native speakers
CLARA (1997)	Charles University, Prague	Written	50M words	Lexicographic purposes	Periodicals, books, internet sources from 1975-present
Egypt (1999) [38]	John Hopkins University	Written	Unknown	MT	A parallel corpus of the Qur'an in English and Arabic
Broadcast News Speech (2000)	University of Pennsylvania LDC	Spoken	More than 110 broadcasts	Speech recognition	News broadcast from the radio of voice of America.
DINAR Corpus (2000) [39]	Nijmegen Univ.,SOTETEL-IT, co-ordination of Lyon2 Univ	Written	10M words	Lexicography, general research, NLP	Unknown
An-Nahar Corpus (2001) [40]	ELRA	Written	140M words	General research	An-Nahar newspaper (Lebanon)
Al-Hayat Corpus (2002) [41]	ELRA	Written	18.6M words	Language Engineering and Information Retrieval	Al-Hayat newspaper (Lebanon)
Arabic Gigaword (2003) [42]	University of Pennsylvania LDC	Written	Around 400M	Natual language processing, information retrieval, language modelling	Agence France Presse, Al-Hayat news agency, An-Nahar news agency, Xinhua news agency
E-A Parallel Corpus (2003)	University of Kuwait	Written	3M words	Teaching translation & lexicography	Publications from Kuwait National Council
General Scientific Arabic Corpus (2004)	UMIST, UK	Written	1.6M words	Investigating Arabic compounds	http://www.kisr.edu.kw/science/
Classical Arabic Corpus (CAC) (2004)	UMIST, UK	Written	5M words	Lexical analysis research	www.muhaddith.org and www.alwaraq.com
Multilingual Corpus 2004	UMIST, UK	Written	10.7M words (Arabic 1M)	Translation	IT-specialized websites
SOTETEL Corpus	SOTETEL-IT, Tunisia	Written	8M words	Lexicography	Literature, academic and journalistic material

[6] *http://www.comp.leeds.ac.uk/eric/latifa/arabic_corpora.htm*

Table 5.2: Free Arabic corpora

Name of Corpus	*Source*	*Medium*	*Size*	*Purpose*	*Material*
Corpus of Contemporary Arabic (CCA) 2004	University of Leeds, UK	Written and spoken	Around 1M words	TAFL	Websites and online magazines
Arabic Blogs (2009)and a corpus builder application [43]	Shereen Khoja, Pacific University, Oregon, USA	Written	131,836 words	Investigating the use of colloquial Arabic and gender issues	37 blogs around the death of a Saudi female journalist and blogger, Hadeel Alhodaif
Essex Arabic Summaries Corpus (EASC 1.0) [44]	Mahmoud El-Haj, University of Essex, UK	Written	-	-	153 Arabic articles and 765 human generated extractive summaries of the article.

Table 5.3: Under development Arabic Corpora

Name of Corpus	*Source*	*Medium*	*Size*	*Purpose*	*Material*
International Corpus of Arabic (ICA) 2008-2009	Bibliotheca Alexandrina (BA)	Written	100M words	General linguistic research	A wide range of sources from the Internet representing different Arabic regions

Corpus sizes for the same topics written in Arabic and other different languages are not the same. In fact, the size of the corpus extracted from the French newspaper “Le monde” from the period of 4 years, is 80 million words [23]. Moreover, the size of corpus extracted from the period of almost 7 years of Associated French Press (*AFP)* Arabic Newswire, and released in 2001 by *LDC* is 76 million tokens [4, 5]. This gap between the two sizes is justified by the compact form of the Arabic words. Formally speaking, the English word “write” is equivalent to one Arabic word “كتب”. But the group “He writes”, made up of two words, and also corresponds to one Arabic word “يكتب“. And the Arabic equivalent of the sentence “He will write” is the only one word “سيكتب”. Moreover, the word “سيكتبه” amounts to the group of words “He will write it”. Another example is the Arabic word (وبنفوذها) and its equivalence in English (4 words) “and with her influences”. This makes segmentation of Arabic textual data different and more difficult than Latin languages. This gives an explanation of the gap between the two corpuses size, if we make into consideration the difference of data extraction period [2, 3]. On the other hand, the required amount of storage (disk or RAM) for Arabic corpus is twice of English corpus for the same size of characters because Arabic characters require 2 bytes to be saved in Unicode format. This implies that feature reduction for Arabic text is necessary to consider storage limit.

5.1 Corpora Building Steps

The main consecutive phases of building a text classification system (presented in figure 1.6) has been described in section 1.2. The first phase in construction process is to build a text dataset which involves compiling and labeling text documents into corpus. We collect web documents from internet using the open source offline explorer, httrack[7]. The process also includes converting corpus html/xml files into *UTF-8* encoding using "Text Encoding Converter" by WebKeySoft[8]. The final step is to strip/remove html/xml tags as shown in Figure 5.1. We developed a Java program that strip / remove html/xml tags. The program is available publically at [68].

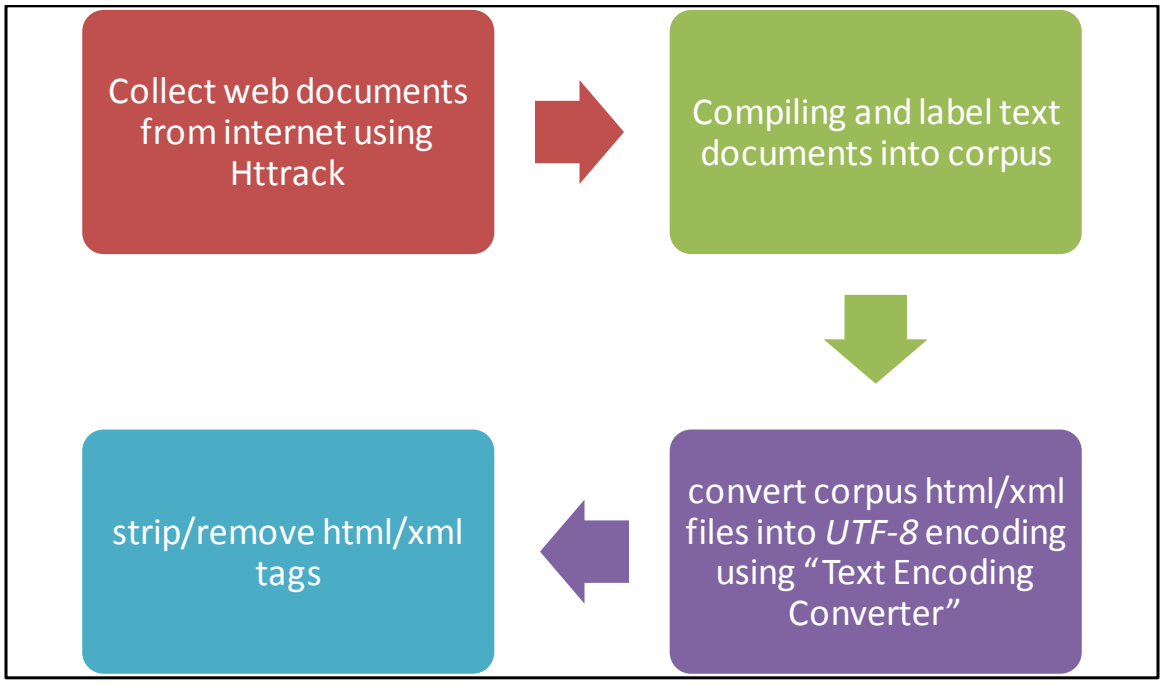

Figure 5.1: Corpus Building Steps

[7] www.httrack.com
[8] www.webkeysoft.com

5.2 Corpora Summary

We use various corpora to perform our experimentations, the corpora variations include small/large size corpus, with few and more categories. The used corpora have been collected by us and by other researchers. We collected three corpora, we collect them from: *BBC* Arabic, *CNN* Arabic, and the third corpus was collected from multiple websites, we shall call the third corpus as "*Open Source Arabic Corpus*" (*OSAC*). The three corpora are available publically at [68]. The corpora that collected by other researchers includes Contemporary Corpus of Arabic (*CCA*) [15, 16], *Aljazeera* corpus [10], *khaleej-2004* corpus [2, 3], and a corpus collected by [85] (we shall call the corpus as "*W"* corpus).

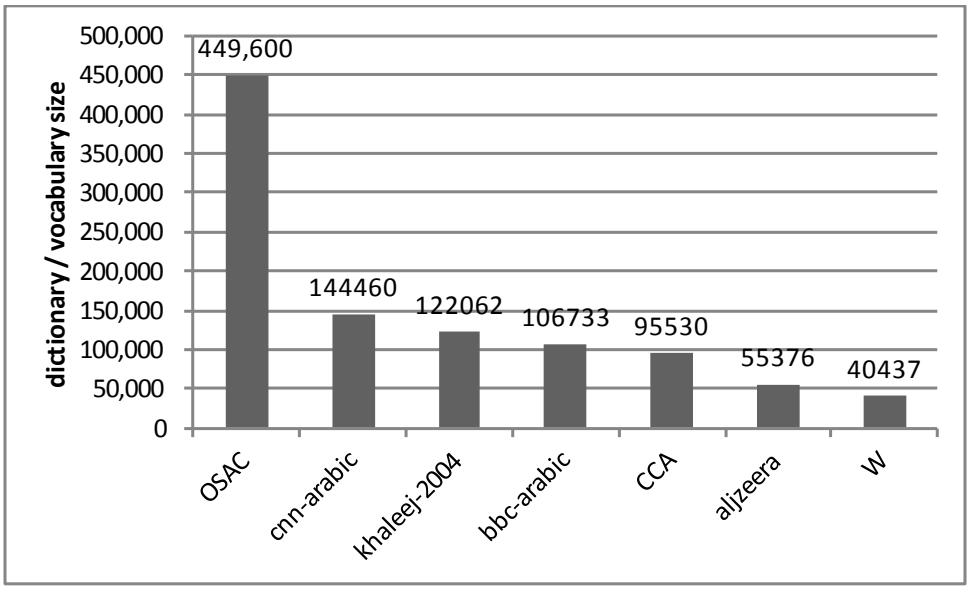

Figure 5.2: Dictionary (# of keywords) size for each corpus

Figures 5.2 and 5.3 present the corpora we used in this research, the used corpora have various keywords size and various number of documents. Figure 5.2 presents the district keywords (dictionary/vocabulary size) for each corpus, and Figure 5.3 shows the number of text documents for each corpus. Despite *CCA* corpus has a small number of text documents (293 text documents) but it has relatively a large number of keywords (95,350 keywords), the reason is

that *CCA* corpus has large size text documents (long documents), also, the corpus covers broad range of text genre. *OSAC* corpus has the largest number of text documents and largest vocabulary. In the following, we shall describe each corpus in details.

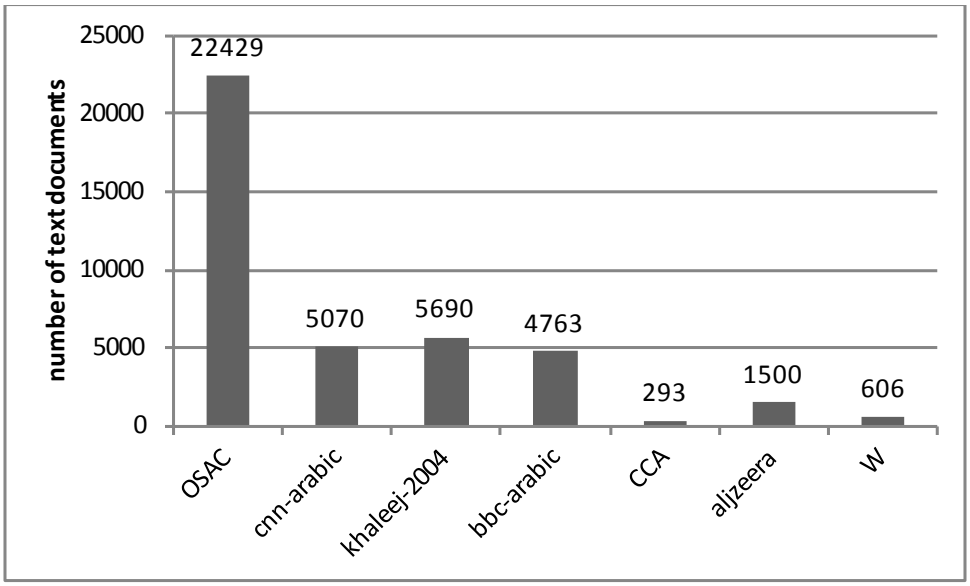

Figure 5.3: Number of text documents for each corpus

5.2.1 *CCA* corpus

The corpus of Contemporary Arabic (*CCA* Corpus) [15, 16] was released from the University of Leeds by Latifa Al-Sulaiti and Eric Atwell. Their survey confirms that the existing corpora are too narrowly limited in source-type and genre, and that there is a need for a freely-accessible corpus of contemporary Arabic covering a broad range of text-types. We merged some categories of *CCA* (like short stories with children stories, and Science A with Science B), and eliminated other categories because those categories have few text documents. The corpus contains 293 text documents belonging to 1 of 5 categories (Autobiography 73, Health and Medicine 32, Science 70, Stories 58, Tourist and travel 60). The corpus includes 95,530 district keywords after stopwords removal.

5.2.2 *W* corpus

W corpus was collected by [85]. The corpus includes 606 text documents. Each text document belongs 1 of 6 categories (Agriculture 100, Art 90, Economy 100, Health 116, Politics 100, Science 100). The corpus includes 40,437 district keywords after stopwords removal. The corpus was used by [85] to investigate different variations of vector space models (*VSMs*) and term weighting approaches using *KNN* algorithm. Dataset is collected from online Arabic newspapers (*Al-Jazeera, Al-Nahar, Al-hayat, Al-Ahram, and Al-Dostor*).

5.2.3 *Aljazeera* corpus

Aljazeera corpus was used by [73]. The corpus includes 1,500 text documents. Each text document belongs 1 of to 5 categories (Art, Economy, Politics, Science, Sport), each category includes 300 documents. The corpus includes 55,376 district keywords after stopwords removal.

5.4 *Khaleej-2004* corpus

Khaleej-2004 corpus was collected by [2, 3] from *Khaleej* newspaper of the year 2004. The corpus includes 5,690 text documents. Each text document belongs 1 of to 4 categories (Economy 909, Local News 2398, International News 953, Sport 1430). The corpus includes 122,062 district keywords after stopwords removal.

5.5 *BBC* Arabic corpus

We collected *BBC Arabic* corpus from *BBC* Arabic website bbcarabic.com, the corpus includes 4,763 text documents. Each text document belongs 1 of to 7 categories (Middle East News 2356, World News 1489, Business & Economy 296, Sports 219, International Press 49, Science & Technology 232, Art & Culture 122). The corpus contains 1,860,786 (1.8M) words and 106,733 district keywords after stopwords removal. We converted the corpus to *utf-8* encoding and stripped html tags. The corpus is available publically at [68].

5.6 *CNN* Arabic corpus

We collected *CNN Arabic* corpus from *CNN* Arabic website cnnarabic.com, the corpus includes 5,070 text documents. Each text document belongs 1 of to 6 categories (Business 836, Entertainments 474, Middle East News 1462, Science & Technology 526, Sports 762, World News 1010). The corpus contains 2,241,348 (2.2M) words and 144,460 district keywords after stopwords removal. We converted the corpus to utf-8 encoding and stripped html tags. The corpus is available publically at [68].

5.7 *OSAC* corpus

We collected *OSAC* Arabic corpus from multiple websites as presented in Table 5.4, the corpus includes 22,429 text documents. Each text document belongs 1 of to 10 categories (Economics, History, Entertainments, Education & Family, Religious and Fatwas, Sports, Heath, Astronomy, Low, Stories, Cooking Recipes). The corpus contains about 18,183,511 (18M) words and 449,600 district keywords after stopwords removal. We converted the corpus to utf-8 encoding and stripped html tags. The corpus is available publically at [68].

Table 5.4: *OSAC* corpus

Category	# of text documents	Sources
Economic	3102	bbcarabic.com cnnarabic.com aljazeera.net khaleej.com banquecentrale.gov.sy
History	3233	تاريخ الحكام www.hukam.net moqatel.com التاريخ altareekh.com تاريخ الاسلام islamichistory.net
Education and family	3608	صيد الفوائد saaid.net نصائح للسعادة الاسرية naseh.net المربي almurabbi.com
Religious and Fatwas	3171	CCA corpus EASC corpus moqatel.com شبكة الفتاوى الشرعية islamic-fatwa.com صيد الفوائد saaid.net
Sport	2419	bbcarabic.com

		cnnarabic.com khaleej.com
Health	2296	dr-ashraf.com العيادة الالكترونية CCA corpus EASC corpus W corpus kids.jo صحة الطفل arabaltmed.com العلاج البديل العربي
Astronomy	557	arabastronomy.com الفلك العربي alkawn.net الكون نت bawabatalfalak.com بوابة الفلك المغربية nabulsi.com الفلك – موسوعة النابلسي www.alkoon.alnomrosi.net
Low	944	lawoflibya.com القانون الليبي qnoun.com قانون كوم
Stories	726	CCA corpus kids.jo قصص الاطفال saaid.net صيد الفوائد
Cooking Recipes	2373	aklaat.com fatafeat.com
TOTAL	**22,429**	

In the next chapter, we shall present and discuss preprocessing techniques described in chapter 4 on the corpora that we described in this chapter. In addition, next chapter presents the result of applying different classifiers that we described in chapter 3 on the corpora we described in this chapter.

Chapter 6 : Experimental results and analysis

In this chapter, we present and analyze experimental results. Text Classification algorithms (*C4.5* Decision Tree (*TD*), *K* Nearest Neighbors (*KNN*), Support Vector Machines (*SVMs*), Naïve Bayes (*NB*), Naïve Bayes Variants (Naïve Bayes Multinomial (*NBM*), Complement Naïve Bayes (*CNB*), Discriminative Multinomial Naïve Bayes (*DMNB*))) are described in chapter 3, experimental setup is described in chapter 4, and corpora are described in chapter 5. We split each corpus to 2 parts (66% of the corpus for training and the remaining 34% for test). We could not run any classifier in batch mode because the corpora size is very large and did not fit to memory. All classifiers were run in incremental mode on 64-bit machine with 4GB RAM. We use cross-validation method provided by WEKA and RapidMiner to determine the optimal value of *K* for *KNN* experiments.

Experimental results investigate preprocessing time, classifiers accuracy and training time, the impact of morphological analysis, the impact of weighting schemes, and the impact of distance function on *KNN*.

We could not generate all text representation for *OSAC* corpus because it does not fit to memory. Also, we could not run all text classifiers on this corpus for the same reason. We generated two text representations for this corpus: (stemming + *wc-minFreq5*) and (light stemming + *wc-minFreq5*).

6.1 Dimensionality reduction

As mention previously, the very high dimensionality of text data is one of the problems of text mining. Popular dimensionality reduction techniques include term stemming and pruning. Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base

or root form. Term pruning is the process eliminating words that its count that less than a specific threshold. Figure 6.1 shows different dimensionally reduction techniques applied on different corpora compared to raw text. The Figure shows that stemming + term pruning with threshold of 5 words has the highest reduction. The order of reduction techniques from the highest to lowest reduction rate as shown in Figure 6.1 are: Stemming + *wc-minFreq5*, Stemming + *wc-minFreq3*, Light Stemming + *wc-minFreq5*, *wc-minFreq5*, Light Stemming + *wc-minFreq3*, *wc-minFreq3*, Stemming, and Light Stemming.

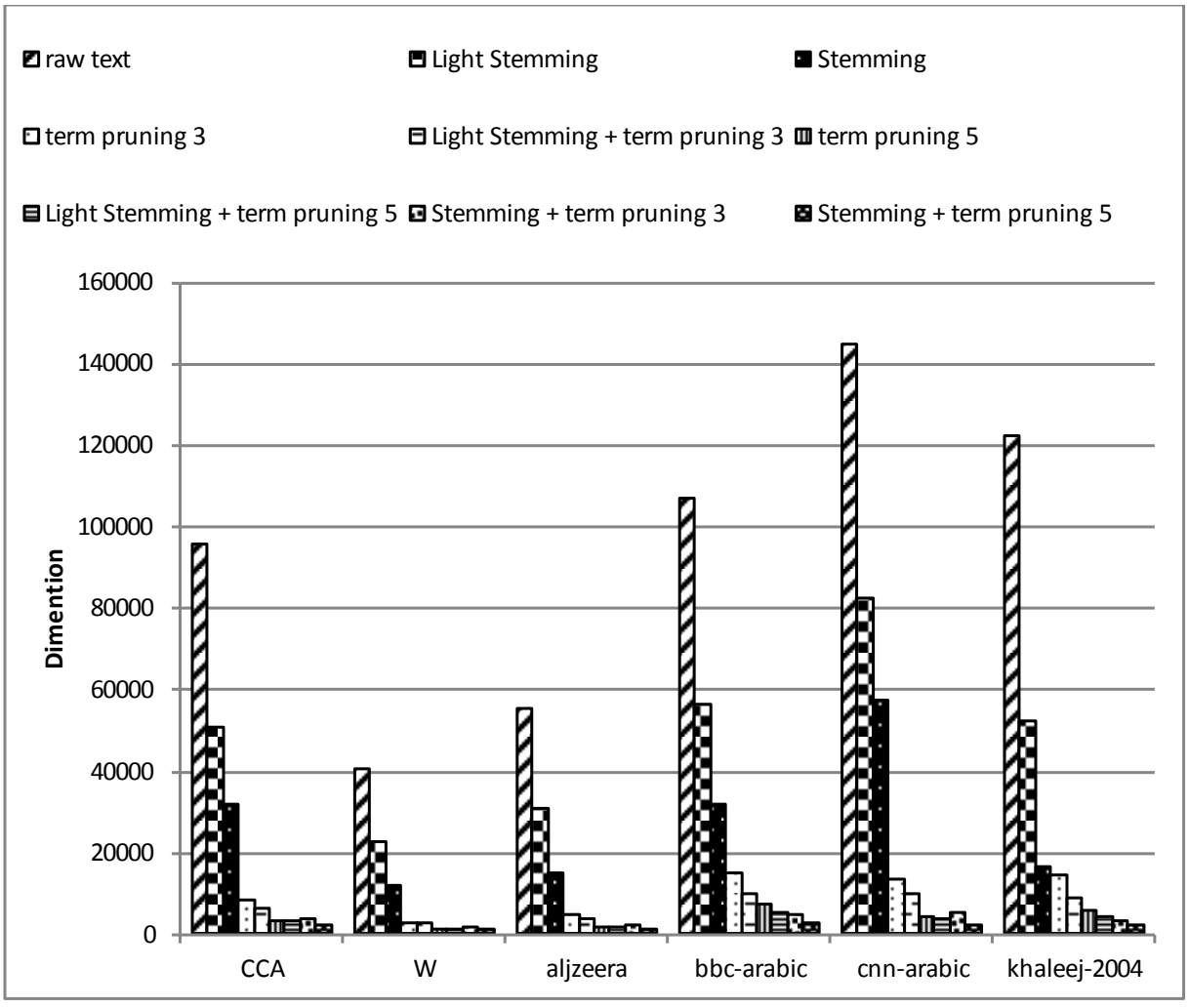

Figure 6.1: Dimensionality reduction using stemming and term Pruning

Figure 6.2 shows dimensionality reduction of stemming and light stemming for the corpora *BBC* Arabic, *CNN* Arabic, *Khaleej-2004*. Light stemming reduction ranges between 39-13% of the original text (raw text) with average of 27% while stemming reduction ranges between 56-42% of the original text (raw text) with average of 50%. Figure 6.3 shows the impact of applying term pruning with threshold 5 on the aforementioned corpora. (Light stemming + *wc-minFreq5*) reduction ranges between 2.6-5% of the original text (raw text) with average of 3.7% while (stemming + *wc-minFreq5*) reduction ranges between 1.6-2.6% of the original text (raw text) with average of 2%.

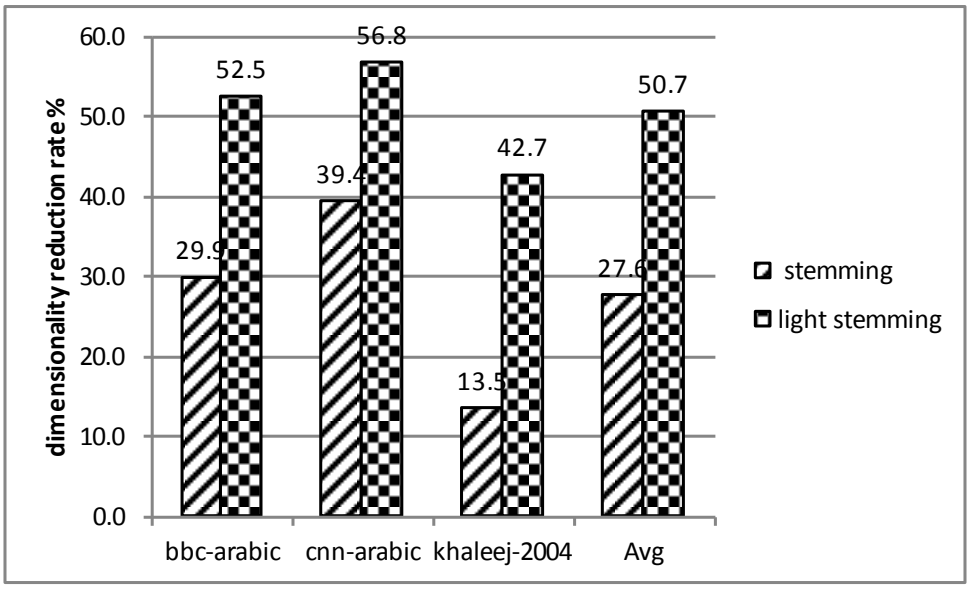

Figure 6.2: The percentage of dimensionality reduction of stemming and light stemming

Figure 6.4 shows the dimensionality reduction of (stemming + wc-minFreq5) vs. (light stemming + *wc-minFreq5*) for *OSAC* corpus. Light stemming reduced the features from 449,600 to 19,565 while stemming reduced the features to 10,899 (about the half of light stemming reduction). (Light stemming + *wc-minFreq5*) reduced the original feature (raw text) to 4.35% while (stemming + *wc-minFreq5*) reduced it to 2.42%. In other words the reduction rates for (light stemming + *wc-minFreq5*) and (stemming + *wc-minFreq5*) is 95.65% and 97.58%

respectively. These analyses about dimensionality reduction techniques have not been addressed in the literature as we posed in this research. Applying morphological analysis and term pruning greatly reduced the dimensionality of text data, this reduction is necessary to save storage and time when we classify a corpus. We shall discuss the impact of the dimensionality reduction on classifier accuracy in section 6.4.

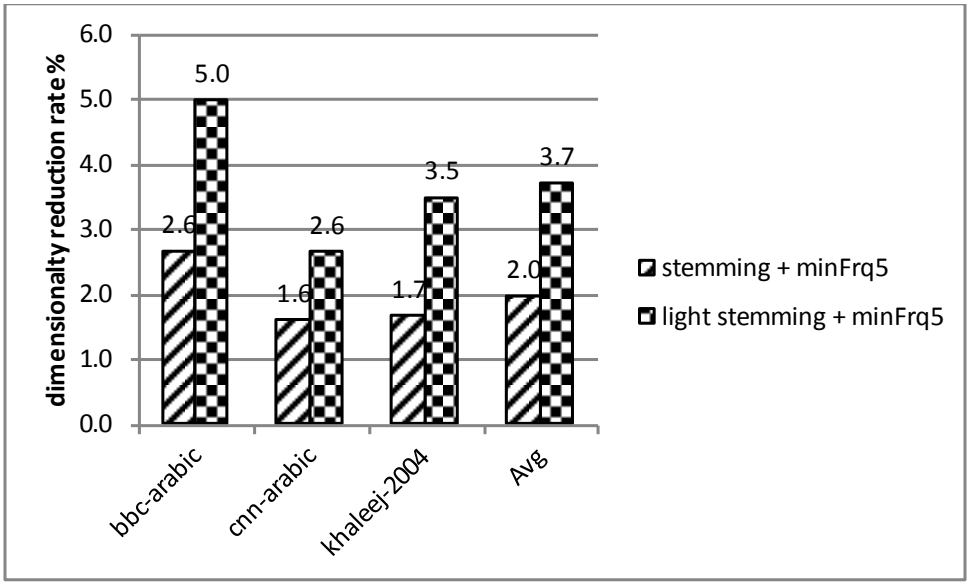

Figure 6.3: The percentage of dimensionality reduction of stemming and light stemming with term pruning

From section 1.2, we have seen that despite some words have the same root, it have different meaning. In other words, different morphologies of the same root have different meaning (see Table 1.9). Thus, we recommend light stemming + term pruning, as a feature reduction technique for Arabic Language even that stemming greatly reduces features because light stemming is more proper than stemming from linguistics and semantic view point. Another reason is that the reduction rates of stemming/light stemming + term pruning are convergent (2% and 3.7% of the original text (raw text)). Furthermore, stemming requires more preprocessing time because of root validation process which search for the valid root and patterns dictionaries.

The results by Duwairi [31, 32] show that the rate of dimensionality reduction is 55 and 77% from the original dataset for stemming and lighting stemming respectively. Duwairi did not applied term pruning and she also used tri-literal root extractor stemmer by Al-Shalabi et. al. [14].

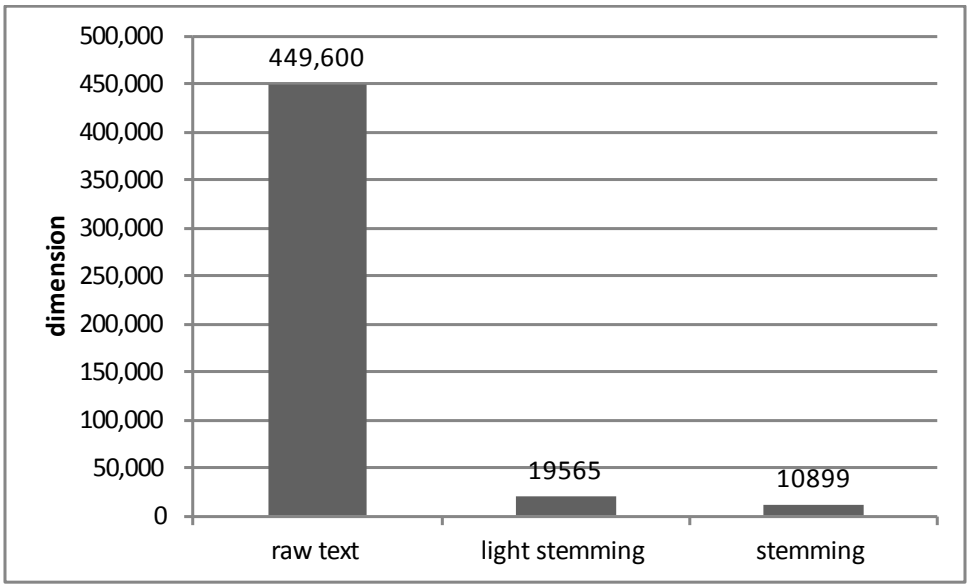

Figure 6.4: Dimensionality reduction of stemming vs. light stemming for *OSAC* corpus

6.2 Preprocessing time

As mentioned previously, text preprocessing includes morphological analysis and term weighting. Raw text requires string tokenization + stopwords removal + term matching (to add word as a count to existing feature or to add it as a new feature). Stemming/light stemming preprocessing requires the same steps in addition to one additional step after stop word removal which is stemming/light stemming. Figure 6.5 shows the average time required to analyze the corpora morphologically (stemming and light stemming), the Figure also shows the average time required to process raw text (without morphological analysis). Light stemming requires the least time to preprocess text data, even less than raw text preprocessing time, this is explained by two

reasons: (1) light stemming algorithm step is fast (just normalizes word and removes suffixes and prefixes), (2) light stemming reduces the original raw text to 50% and to 3.7% with term pruning, in other words, despite raw text does not preprocess text morphologically, it needs more time than light stemming preprocessing time because of high dimensionality of raw text (see Figures 6.1- 6.4). i.e., raw text preprocessing takes long time to search in large (high dimension) feature/dictionary for match terms.

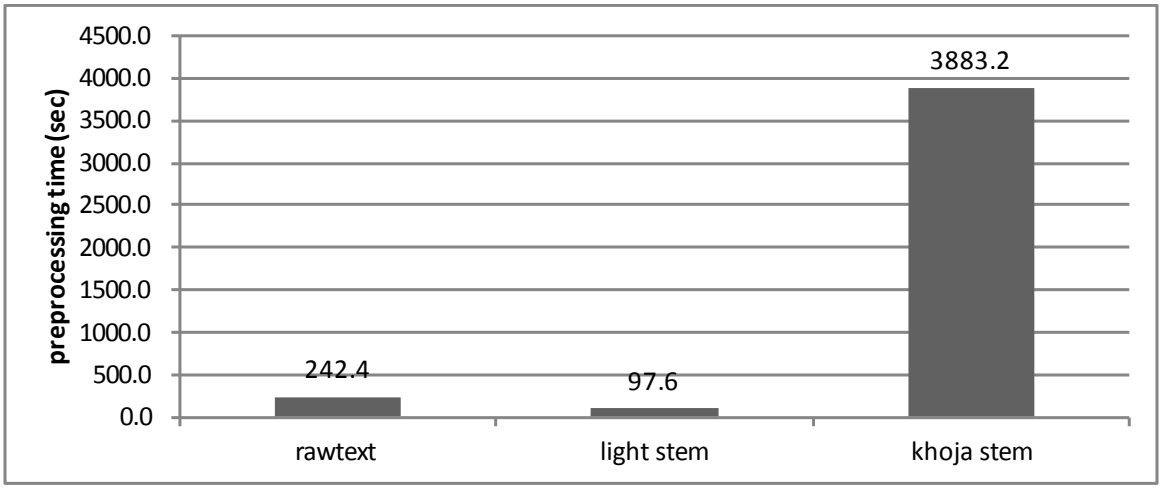

Figure 6.5: Average preprocessing time of raw text, light stemming, and stemming

The results by Duwairi [31, 32] show that preprocessing and classification time of stemming is the least. The result also states that the difference of stemming and light stemming preprocessing and classification time is slight. The reason is that Duwairi used stemmer by Al-Shalabi [14] which does not match pattern and validated extracted root is stemming process.

Preprocessing time of different term weighting schemes is shown in Figure 6.6. In average, all term weighting schemes have approximately similar preprocessing time despite each term weighting scheme has different counting formula. The least preprocessing time is achieved when applying term pruning because it greatly reduces dimensionality which leads to save the required time to look up into feature/dictionary to match terms.

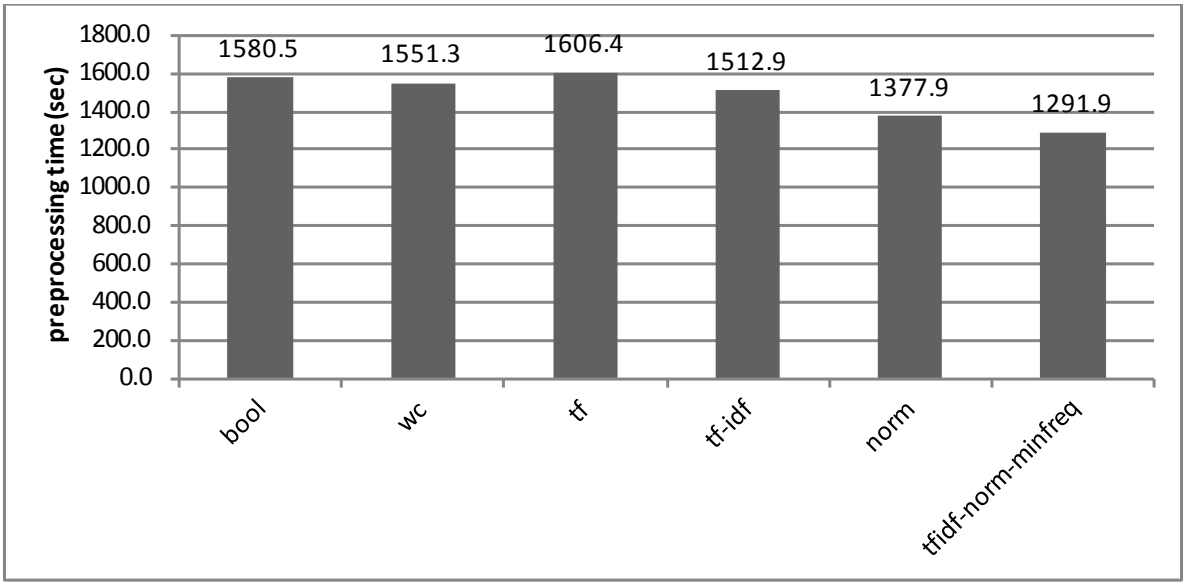

Figure 6.6: Average preprocessing time of different weighting schemes

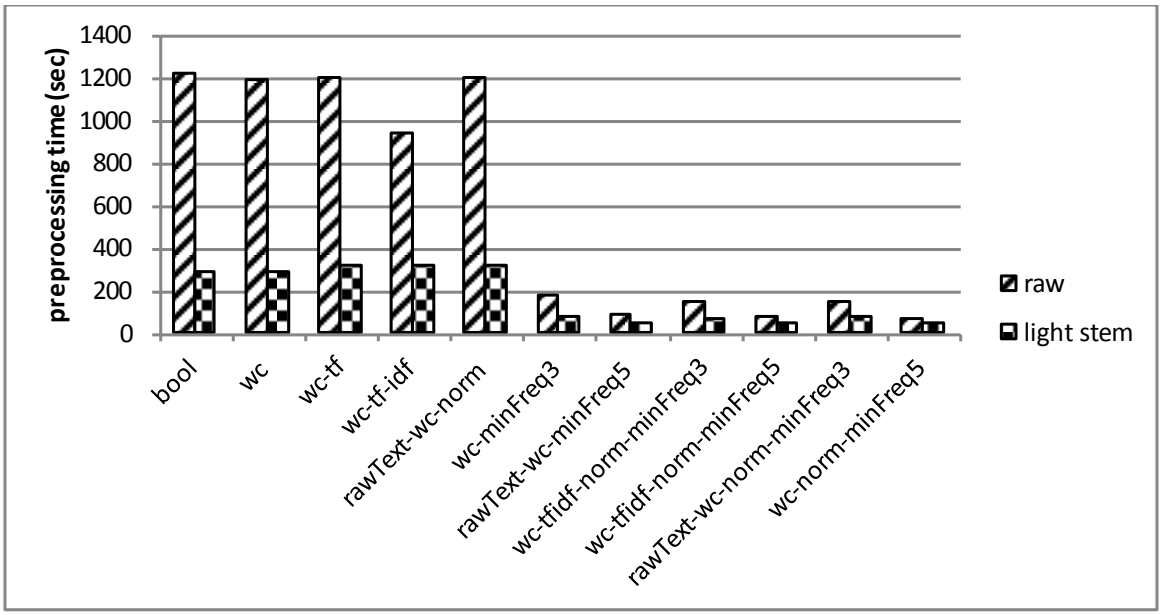

Figure 6.7: Preprocessing time of *khaleej-2004* corpus

Figure 6.7 shows the preprocessing time of *Khaleej-2004* corpus to generate 11 different term weighting schemes for both raw text and light stemming. We eliminate stemming because it requires long time for preprocessing. The Figure also emphasize that all term weighting schemes approximately have the same preprocessing time. Applying term pruning reduces preprocessing

time. Furthermore, Figure 6.7 emphasizes the comments on Figure 6.5 that the light stemming is faster than raw text.

6.3 Classifier Accuracy

Among seven classifiers applied on seven corpora, *SVMs* achieved the highest average accuracy (94.11%), then *DMNB* with average accuracy of 92.33%. *KNN* was the worst with average accuracy of 62.47%. Figure 6.8 shows the classifiers average performance.

Generally, *SVMs* and *NB* variants achieved the best average classification accuracy. *SVMs* achieved the best accuracy because it is a robust classifier, it maps data points into new dimension space, this makes different term weighting schemes have no impact on *SVMs* performance. In addition, *SVMs* is effective on high dimensional data because the complexity of trained classifier is characterized by the number of support vectors rather than the dimensionality of the data.

Text dataset requires considerations like language model, decision boundary for imbalanced text dataset, good parameters estimation, and word dependency. These considerations have been taken into account in *NB* variant classifiers, this makes them achieve the best average accuracy. Furthermore, *NB* variant classifiers inherit *NB* property of naïve assumption of independent features which make them simple and achieve respectable effective performance. We have described text classification consideration and the corrections to *NB* in details in chapter 3.

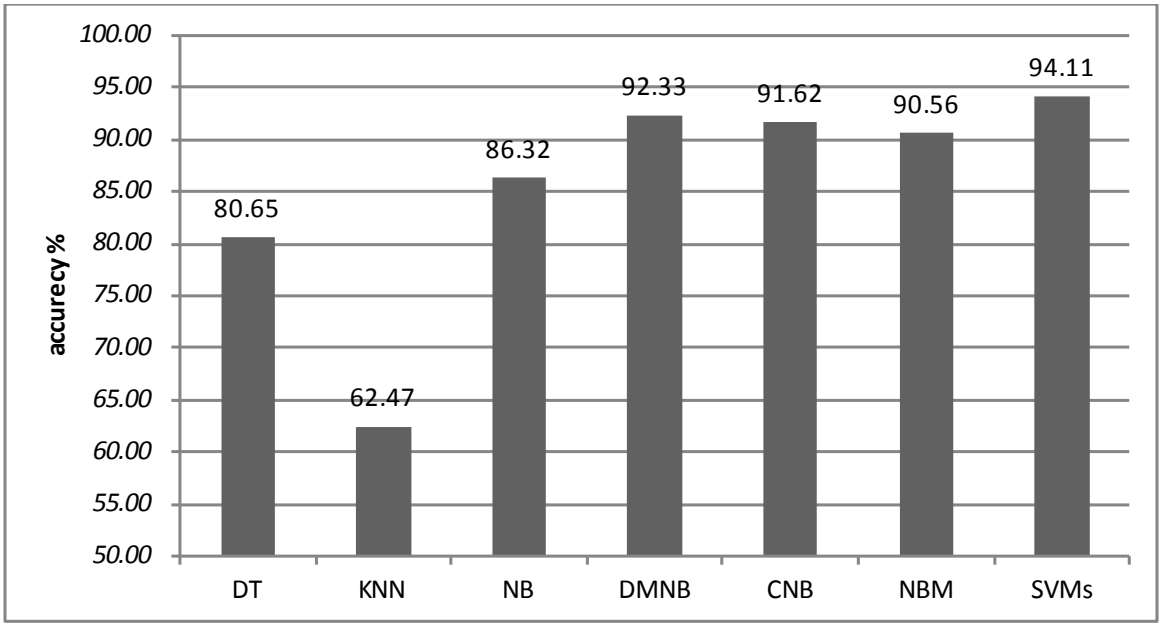

Figure 6.8: Average classification accuracy for seven classifiers

DT is not scalable in high dimensional dataset, and it requires very long training time [46, 86]. Additionally, term weighting schemes have a direct impact on *KNN* because it depends on distance function. Distance functions are not scalable in high dimensional space. *KNN* achieves high performance using (*tf-idf* + normalization + term pruning) term weighting schemes and light stemming feature reduction and term pruning as we will see later in Figure 6.17. Figure 6.9 shows the average accuracy of classifiers applied on *OSAC* corpus. The Figure also emphasizes the comment on Figure 6.8.

Training time is important factor for building any classification system. Due to nature of high dimensionality of text dataset, training takes time. Figure 6.10 shows training time in seconds for the seven text classifiers. *SVMs* and *NB* variants classifiers take shortest time for training, while *DT* required the longest training time. Figure 6.11 shows the training time in seconds for text classifiers that have been applied on *OSAC* corpus, the Figure shows that *SVMs* and *NB* variants outperforms *NB* in term of training time.

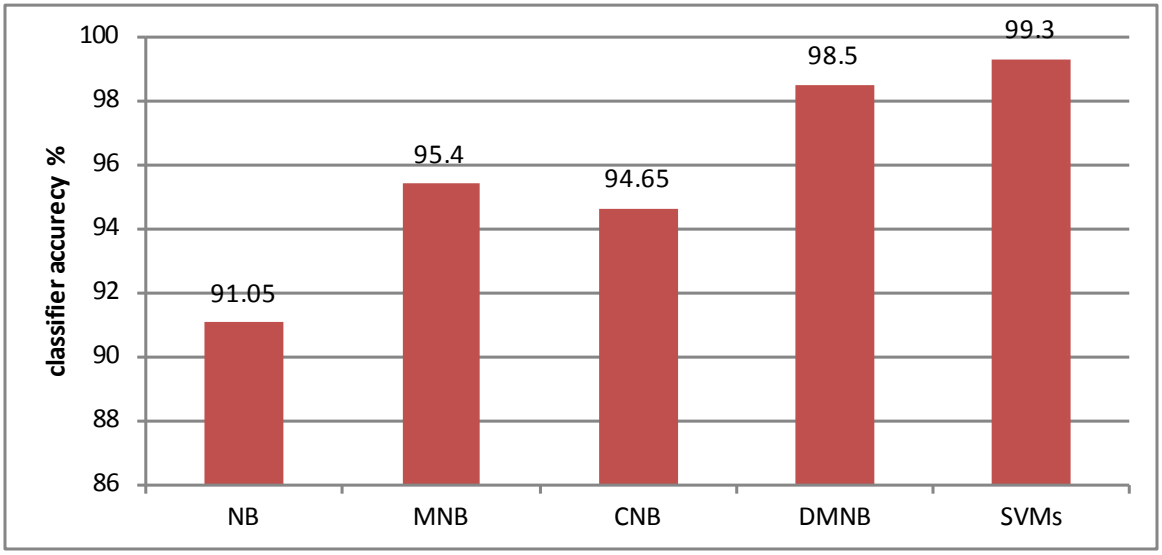

Figure 6.9: Average accuracy of classifiers that applied on *OSAC* corpus

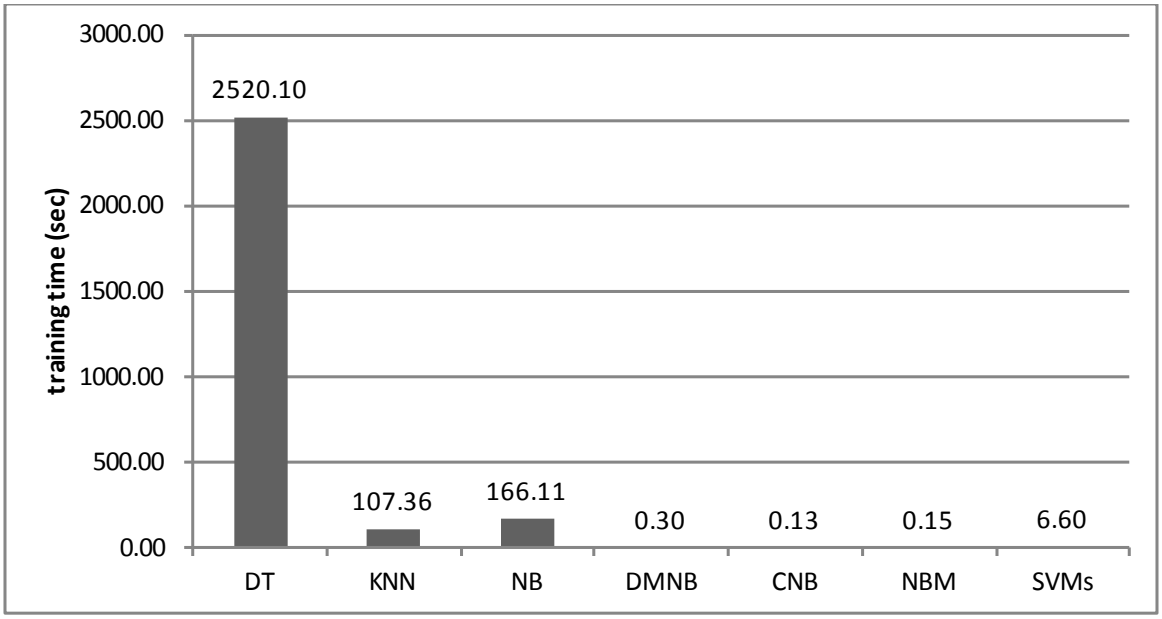

Figure 6.10: Average classifiers training time

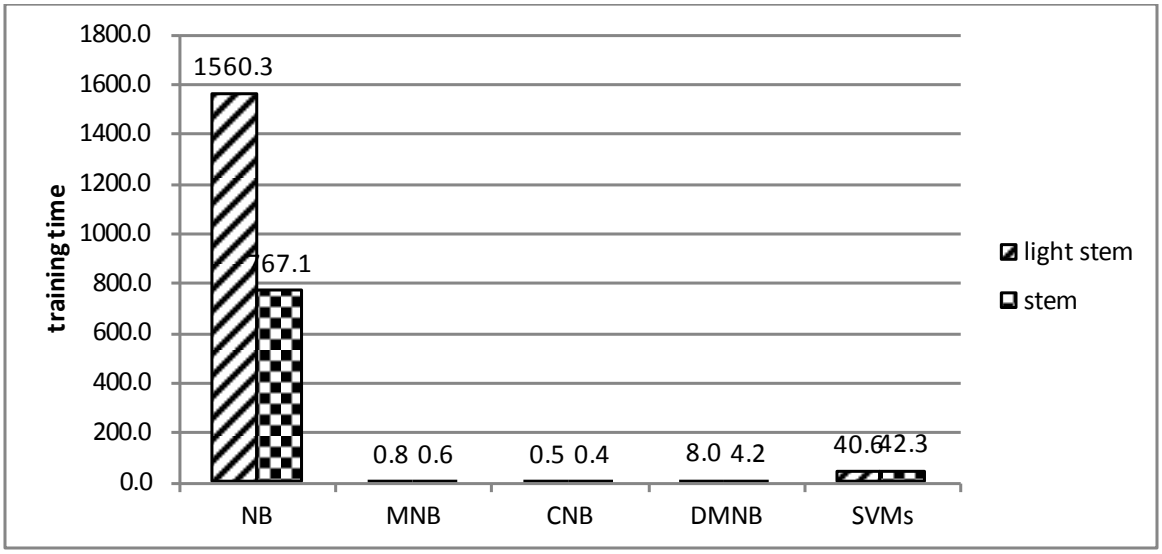

Figure 6.11: Training time for *OSAC* corpus

6.4 Morphological Analysis and term pruning

Morphological analysis tools (stemming / light stemming) can be used to reduce features as described in chapter 1 and 4. In addition, term pruning can be used for the same purpose. We discussed the impact of morphological analysis tools on feature reduction in section 6.1, in this section; we shall discuss the impact of morphological analysis tools on classification accuracy.

The impact of morphological analysis and term pruning on different corpora is depicted in Figure 6.12. The Figure shows that the average classification performance for raw text, stemming, and light stemming are convergent because the morphological analysis and term pruning have slight impact on most classifiers. In other words, (*SVMs* and *NB* variants) average classification performance is approximately the same as shown in Figures 6.12, 6.13, and 6.14.

Morphological analysis has obvious impact on *KNN* performance is shown as Figure 6.13. Figure 6.14 shows classification performance of stemming vs. light stemming vs. raw text for different corpus. Figure 6.15 shows stemming and light stemming average classification accuracy for *OSAC* corpus, Light stemming leads to superior performance for all classifiers.

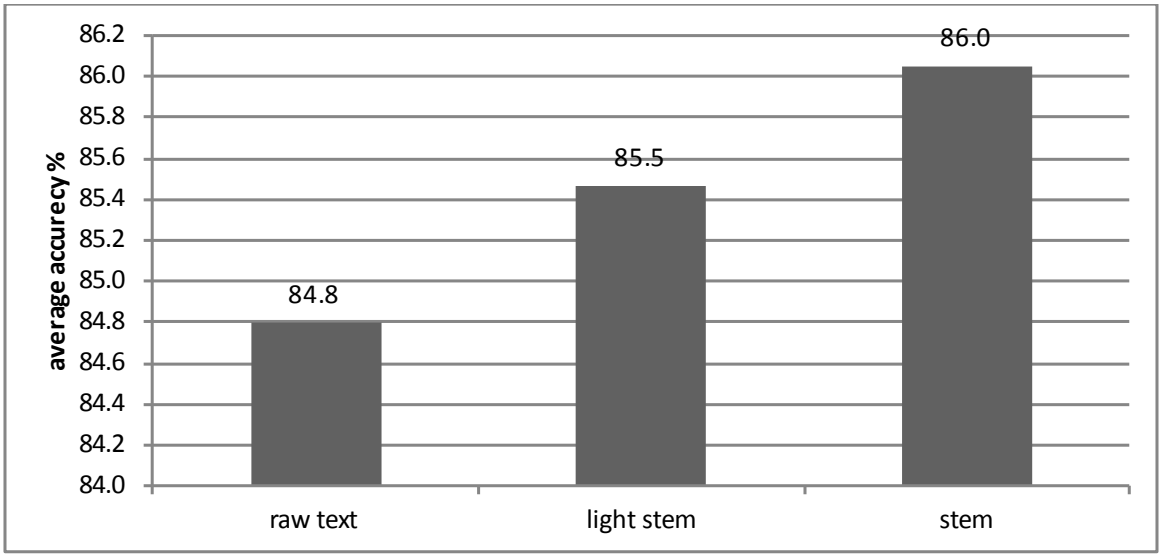

Figure 6.12: Stemming vs. light stemming vs. raw text (Average Accuracy)

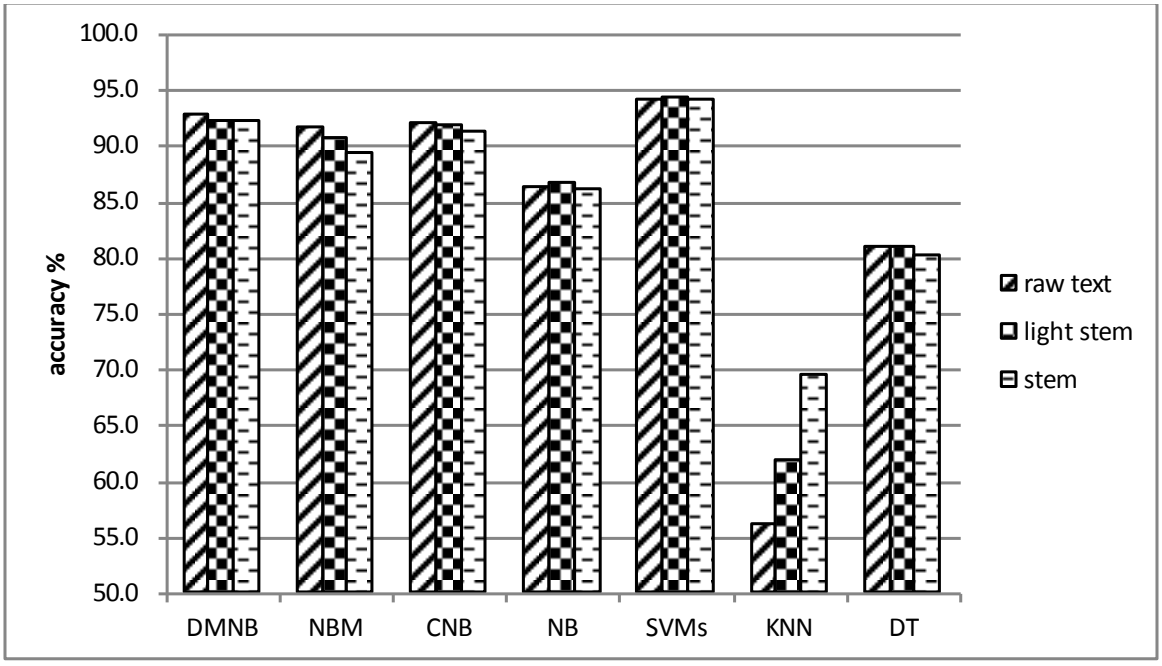

Figure 6.13: The accuracy of stemming vs. light stemming vs. raw text for different classifiers

Again, we recommend light stemming as Arabic morphological analysis tool to improve average classification performance; we recommend it despite stemming has slight better average accuracy because light stemming is more proper than stemming from linguistics and semantic view point and it has the least preprocessing time. The reason for stemming has the slight classification performance than light stemming is that the majorities of Arabic words have a tri-

lateral root, in fact between 80 and 85% of words in Arabic are derived from tri-lateral roots [8, 36]. The rest have a quad-letter root, penta-letter root or hexa-letter root. Khoja stemmer works accurately for tri-literal roots, this why it achieved the highest accuracy. Figure 6.15 shows the stemming / light stemming average classification accuracy for *OSAC* corpus. The Figure emphasizes that light stemming has better classification accuracy than stemming.

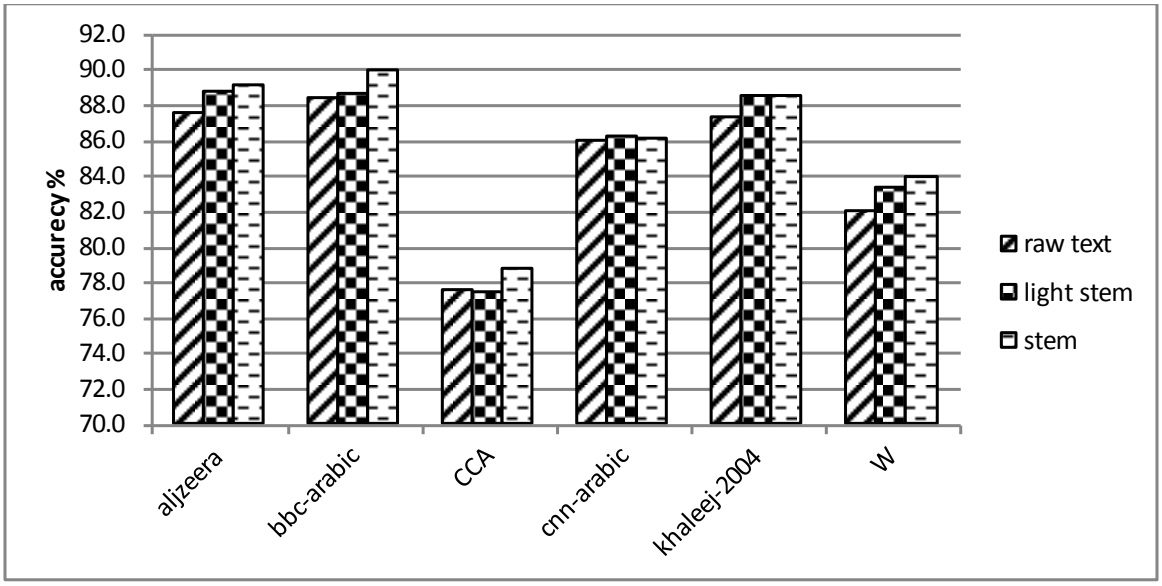

Figure 6.14: The accuracy of stemming vs. light stemming vs. raw text for different corpus

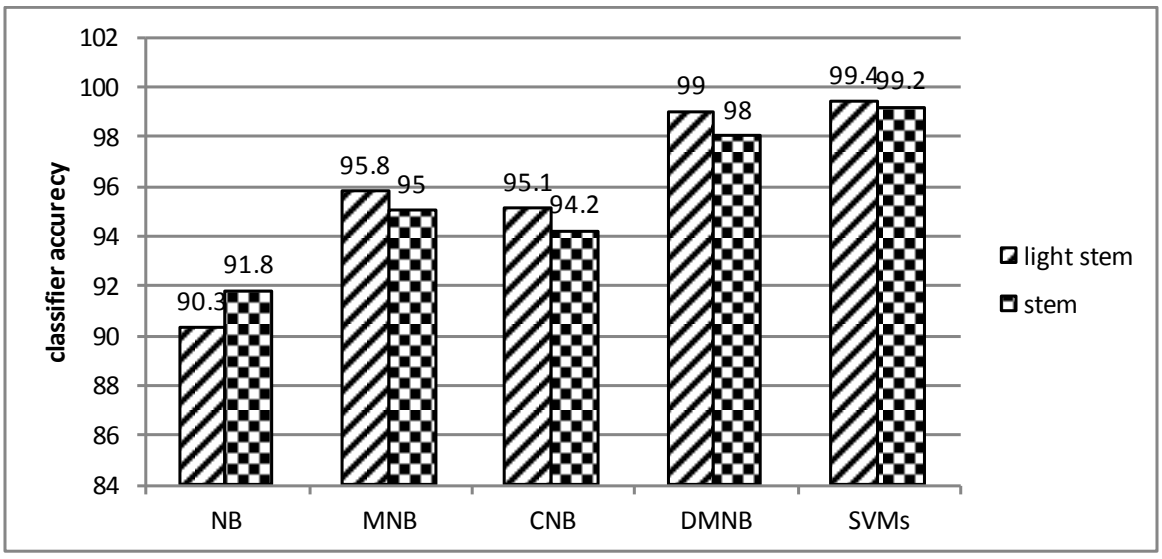

Figure 6.15: Stemming and light stemming classification accuracy for *OSAC* corpus

6.5 Weighting Schemes

The aim of term weighting is to give higher weight to most discriminative terms. Figure 6.16 show the average accuracy of different term weighting schemes. *tf-idf* + normalization + term pruning has the highest accuracy rate, Boolean model also achieved high accuracy rate. Generally, Figure 6.16 elucidates that all term weighting schemes approximately have convergent accuracy rate. This resulted from *SVMs* and *NB* variant classifiers which are insensitive to different term weighting schemes as shown in Figure 6.17. *KNN* classifier is very sensitive to term weighting schemes because it depends on distance function to determine the nearest neighbors. *KNN* achieved the highest accuracy using *tf-idf* + normalization + term punning as shown in Figure 6.17.

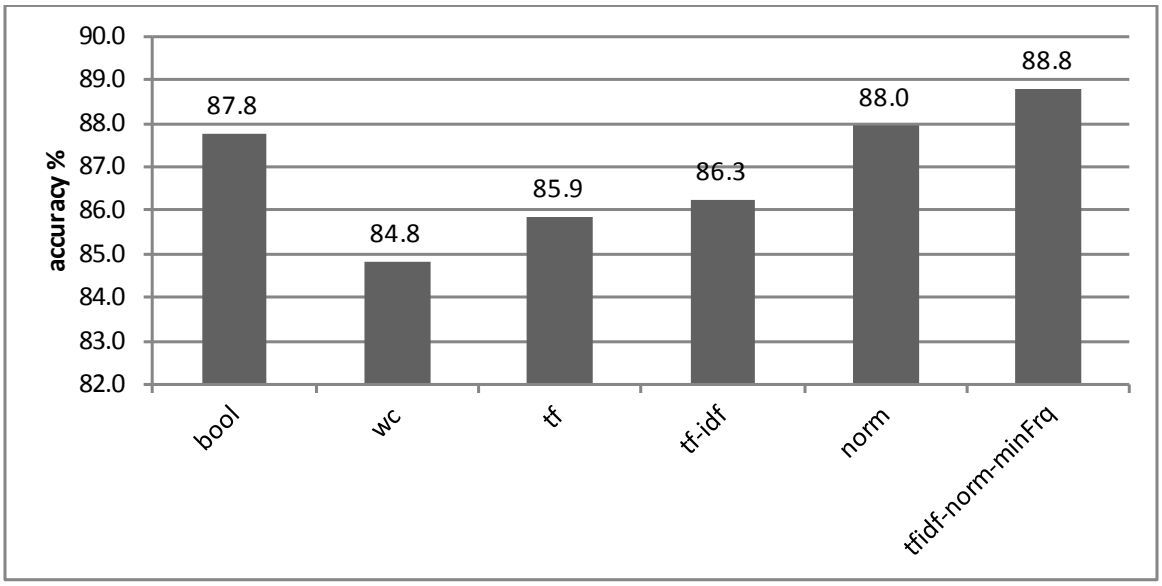

Figure 6.16: Average accuracy for term weighting schemes

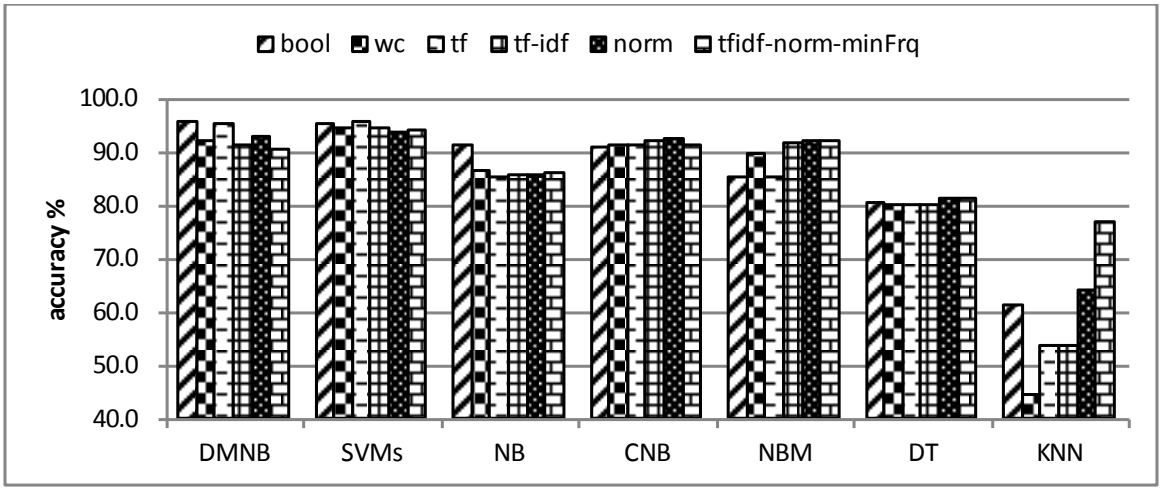

Figure 6.17: Term weighting schemes vs. classifiers

6.6 *Cosine* vs. *Euclidian* Distance Metric

In this experiment, we use *KNN* with *K*=11 as a text classifier, and applied *Cosine* and *Euclidian* distance function on *tf-idf* and binary text representations of *OSAC* corpus. We use light stemming with percentual term pruning (min threshold = 3%, max threshold = 30%) as a feature reduction techniques. Prune below/above percent ignores words that appear in less/more than this percentage of all documents. 3 and 30% are the default values of percentual pruning in RapidMiner operator. Figure 6.18 shows *Cosine* / *Euclidian* Distance vs. *tf-idf / bin* performance. Term weighting has direct impact on *Euclidian* distance function while it has no impact on *Cosine* distance function.

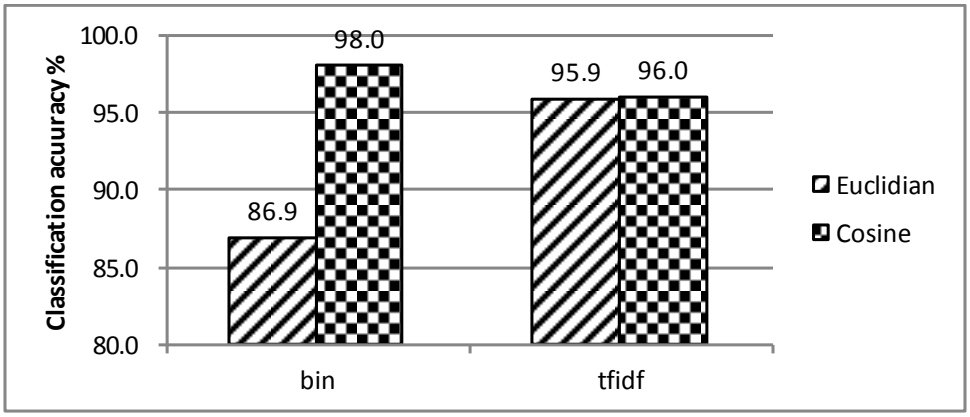

Figure 6.18: *Cosine* / *Euclidian* Distance vs. *tf-idf* / *bin*

Chapter 7 : Conclusion and Future work

7.1 Conclusion

Text mining is on the cross road of information retrieval and machine learning. Arabic text mining is promising research field due to the complexity and problems in different aspects: The lack Arabic corpus, lack of language tools, and lack of comprehensive study on Arabic text preprocessing.

In this research, we successfully compiled the largest freely accessible corpora with 18M words and about 0.5M district keyword. We also implement and integrate Arabic morphological analysis tools to leading open source machine learning and data mining tools. Using the collected corpora and implemented tools, we conduct a comprehensive study that investigates the impact of Arabic text preprocessing (morphological analysis and term weighting schemes) on Arabic text classification using seven text classifiers. We resolved debates and contradictions in the literature.

Experimental results showed that we cannot avoid feature reduction for Arabic language to reduce complexity for classifiers, reduce storage requirements and to save time. Stemming / light stemming greatly reduced features to average of 30% and 50% of the original feature space respectively and to 2% and 4% of the original feature space respectively when prune terms. We conclude that light stemming + term pruning is the best feature reduction technique because light stemming is more proper than stemming from linguistics and semantic view point, and it has the least preprocessing time, it also has superior average classification accuracy.

SVMs is a robust classifier even in high dimensions. Language consideration in *NB* variants improved performance. *SVMs* and *NB* variant have superior performance and achieved the best classification accuracy.

Term indexing and weighting aim to represent high quality text. The High quality in text mining usually refers to some combinations of relevance, novelty, and interestingness. Several approaches are used to index and weight terms but all of them share the following characteristics: The more the number of times a term occurs in documents that belong to some category, the more it is relative to that category. The more the term appears in different documents representing different categories, the less the term is useful for discriminating between documents as belonging to different categories. Term weighting schemes have direct impact on distance based classifiers. Distance based classifiers also affected by the used distance metric.

7.2 Future Works

In the future works, we shall work on extending and elaborating *BBC* Arabic corpus, *CNN* Arabic corpus, and *OSAC* corpus. Elaborations include performing extensive corpus analysis and tag them with Part of speech tags. We also open the door for other researchers and contributors to elaborate the open source corpora.

We shall develop a classifier that classifies any text document based on set of keywords, this will save the time to preprocess test text documents. Keywords are ranked based on different Language aspects based on semantic. Hierarchy classification is also will be supported by our future classifier.

References

[1]. Aas K., Eikvil L., "Text Categorization: A survey", *Technical report, Norwegian Computing Center.* 1999.

[2]. Abbas M., Smaili K., Berkani D., "A Trigger-based Classifier", *In The 2nd Int. Conf. on Arabic Language Resources and Tools (MEDAR 2009)*, 22-23 April 2009, Cairo, Egypt.

[3]. Abbas M., Smaili K., Berkani D., "Comparing TR-Classifier and KNN by using Reduced Sizes of Vocabularies", *In The 3rd Int. Conf. on Arabic Language Processing, CITALA2009, Mohammadia School of Engineers, Rabat, Morocco*. 2009.

[4]. Abdelali, A., Cowie, J., Soliman, H., "Building a modern standard corpus, Workshop on Computational Modeling of Lexical Acquisition", *In the Split Meeting, Split*, 2005.

[5]. Abdelali, A., Cowie, J., "Regional corpus of modern standard Arabic*", In 2nd Int. Conf. on Arabic Language Engineering*, 2005, Vol. 1, No. 1, 2005, pp. 1-12.

[6]. Al-Ansary, S. Nagi, M., Adly N., "Building an International Corpus of Arabic (ICA): Progress of Compilation Stage*", Bibliotheca Alexandrina*, 2008.

[7]. Al-Ansary, S., Nagi, M., Adly N*.,* "Towards analyzing the International Corpus of Arabic: Progress of Morphological Stage*", Bibliotheca Alexandrina*, 2008.

[8]. Al-Fedaghi, S., Al-Anzi, F*.,* "A new algorithm to generate Arabic root-pattern forms"*, In Proc. of the 11th National Computer Conf. King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia*, pp. 04–07, 1989.

[9]. Al-Harbi S., Almuhareb A., Al-Thubaity A., Khorsheed M., Al-Rajeh A., "Automatic Arabic Text Classification", *In JADT'08, France*, 2008, pp. 77-83.

[10]. Al-Marghilani A., Zedan H., Ayesh A., "A general framework for multilingual text mining using self-organizing maps"*, In the 25th IASTED Int. Multi-Conf: Artificial Intelligence and Applications (AIA'07), Innsbruck, Austria*, 2007, pp. 520-525.

[11]. Al-Marghilani A., Zedan H., Ayesh A.: *Text Mining Based on the Self-Organizing Map Method for Arabic-English Documents.* Proc. of the 19th Midwest Artificial Intelligence and Cognitive Science Conf. (MAICS 2008), Cincinnati, USA, pp. 174-181 , 2008.

[12]. Alruily M., Ayesh A, Zedan H., "Crime Type Document Classification from Arabic Corpus*"*, *In the 2nd Int. Conf. on Developments in eSystems Engineering*, pp.153-159, 2009.

[13]. Al-Shalabi R., Kannan G., Gharaibeh H., "Arabic text categorization using KNN algorithm"*, In the Proc. of Int. multi conf. on computer science and information technology CSIT06,* 2006.

[14]. Al-Shalabi, R., Kanaan, G., Al-Serhan, H., "New approach for extracting Arabic roots*", In the Int. Arab Conf. on Information Technology ACIT'2003, Egypt*, 2003.

[15]. Al-Sulaiti L, Atwell E., "Designing and developing a corpus of contemporary Arabic", *In the Int. Journal of Corpus Linguistics,* pp. 1–36, 2006.

[16]. Al-Sulaiti L, Atwell E., "Designing and developing a corpus of contemporary Arabic"*, In the Proc. of the 6th TALC conference*, 2004.

[17]. Al-Zoghby A., Eldin AS., Ismail NA., Hamza T., "Mining Arabic Text Using Soft Matching association rules", *In the Int. Conf. on Computer Engineering & Systems, ICCES'07*, 2007.

[18]. "Arabic diacritics" - Wikipedia, the free encyclopedia, (2010, August), [Online]. Available: http://en.wikipedia.org/wiki/Arabic_diacritics

[19]. "Arabic language" - Wikipedia, the free encyclopedia, (2010, August), [Online]. Available: http://ar.wikipedia.org/wiki/لغة_عربية
[20]. "Arabic Stop words", (2010, August), [Online]. Available: http://sourceforge.net/projects/arabicstopwords
[21]. Ayadi R., Maraoui M., Zrigui M., "Intertextual distance for Arabic texts classification", *In Int. Conf. for Internet Technology and technology*, 2009.
[22]. Bawaneh M., Alkoffash M., Al-Rabea A., "Arabic Text Classification using K-NN and Naïve Bayes", *In Journal of Computer Science,* 4 (7), pp. 600-605, 2008.
[23]. Brun A., "Topic Detection and Adaptation of language models for automatic speech recognition", PhD dissertation, Henri Poincaré University, Nancy1, 2003.
[24]. Callan J., "human language technologies, text categorization", (2010, August), [Online]. Available: http://www.cs.waikato.ac.nz. 2004.
[25]. Caruana, R., Niculescu-Mizil, A., "An empirical comparison of supervised learning algorithms", *In the Proc. of the 23rd int. conf. on Machine learning*, 2006.
[26]. Ciravegna F., Gilardoni L., Lavelli A., Ferraro M., Mana N., Mazza, S., Matiasek J., Black W., Rinaldi F., "Flexible Text Classification for Financial Applications: the FACILE System", *In Proc. of PAIS-2000, Prestigious Applications of Intelligent Systems sub-conf. of ECAI2000*, 2000.
[27]. Debole, F. & Sebastiani, F., "An analysis of the relative hardness of reuters-21578 subsets", *Journal of the American Society for Information Science and Technology (JASIST)*, 56(6), pp. 584–596, 2005.
[28]. Duda R., Hart P., Stork D., (2001), "Pattern Classification", (2nd Ed), *Wiley Interscience*.
[29]. Dumais S. T., Platt J., Heckerman D., Sahami M., "Inductive learning algorithms and representations for text categorization*", In the Proc. of ACM-CIKM98*, pp. 148-155, 1998.
[30]. Dunham M., (2003), "Data mining: Introductory and advanced topics", (1st Ed), *Pearson Education*.
[31]. Duwairi R., Al-Refai M., Khasawneh N., "Feature reduction techniques for Arabic text categorization", *Journal of the American Society for Information Science*, 60(11), pp. 2347-2352, 2009.
[32]. Duwairi R., Al-Refai M., Khasawneh N., "Stemming Versus Light Stemming as Feature Selection Techniques for Arabic Text Categorization", *In the 4th Int. Conf. of Innovations in Information Technology, IIT'07*, pp. 446 – 450, 2007.
[33]. Duwairi R., "A Distance-based Classifier for Arabic Text Categorization", *In the Proc. of the Int. Conf. on Data Mining, Las Vegas, USA*, 2005.
[34]. Duwairi R., "Arabic text Categorization", *In the Int. Arab journal of information technology*, 4(2), 2007.
[35]. Duwairi R., "Machine Learning for Arabic text Categorization", *Journal of the American Society for Information Science and Technology*, 57(8), pp. 1005-1010. 2006.
[36]. Eldin, S., "Development of a computer-based Arabic Lexicon", *In the Int. Symposium on Computers & Arabic Language, ISCAL, Riyadh, KSA,* 2007.
[37]. Eldos M., "Arabic Text Data Mining: A Root Extractor for Dimensionality Reduction", ACTA Press, *Ascientific and Technical Publishing Company,* 2002.
[38]. El-Halees A., "A Comparative Study on Arabic Text Classification", *Egyptian Computer Science Journal* 20(2), 2008.

[39]. El-Halees A., "Arabic Text Classification Using Maximum Entropy", *The Islamic University Journal (Series of Natural Studies and Engineering)*, 15(1), pp. 157-167, 2007.

[40]. El-Halees A., "Mining Arabic Association Rules for Text Classification", *In the 1st Int. Conf. on Mathematical Sciences, Al-Azhar University – Gaza, Palestine,* 2006.

[41]. El-Kourdi M., Bensaid A., Rachidi T., "Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm", *In the 20th Int. Conf. on Computational Linguistics, Geneva*, August 2004.

[42]. Fan R., Chang K., Hsieh C., Wang X., Lin C., "LIBLINEAR - A Library for Large Linear Classification", (2008), [Online]. Available: www.csie.ntu.edu.tw/~cjlin/liblinear

[43]. Feldman R., Sanger J., "The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data", *Cambridge University Press*, 2007.

[44]. Ghwanmeh S., "Applying Clustering of Hierarchical K-means-like Algorithm on Arabic Language", *In the Int. Journal of Information Technology,* 3(3), 2005.

[45]. Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I., "The WEKA Data Mining Software: An Update", *SIGKDD Explorations*, 11(1), 2009.

[46]. Han J., and Kamber M., (2006), "Data Mining: Concepts and Techniques", (2nd Ed), *the Morgan Kaufmann Series in Data Management Systems*.

[47]. Harrag F., El-Qawasmeh E., Pichappan P., "Improving Arabic text categorization using decision trees", *In the 1st Int. Conf. of NDT '09*, pp. 110 – 115, 2009.

[48]. Harrag F., El-Qawasmeh E., "Neural Network for Arabic text classification", *In the 2nd Int. Conf. of Applications of Digital Information and Web Technologies, ICADIWT '09*, pp. 778 – 783, 2009.

[49]. Hill T., Lewicki P., (2007), "STATISTICS Methods and Applications", (1st Ed), *StatSoft, Tulsa, OK*.

[50]. Hmeidi I., Hawashin B., El-Qawasmeh E., "Performance of KNN and SVM classifiers on full word Arabic articles", Journal of Advanced Engineering Informatics 22, pp. 106–111, 2008.

[51]. "The Open Directory Project", (2010, August), [Online]. Available: http://www.dmoz.org/about.html

[52]. Jing L., Huang H., Shi H., "Improved feature selection approach TFIDF in text mining*", In the Proc. of the 1st Int. Conf. of machine learning and cybernetics, Beijing*, 2002.

[53]. Joachims T., "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", *In the Proc. of ECML-98, 10th European Conf. on Machine Learning*, 1998.

[54]. Kanaan G., Al-Shalabi R., Ghwanmeh S., "A comparison of text-classification techniques applied to Arabic text", *Journal of the American Society for Information Science and Technology*, 60(9), pp. 1836 – 1844, 2009.

[55]. Kanaan Gh., Al-Shalabi R., Ghwanmeh S., Al-Ma'adeed H., "A comparison of text-classification techniques applied to Arabic text", *Journal of the American Society for Information Science and Technology*, 60(9), pp. 1836 – 1844, 2009.

[56]. Khoja S., Garside R., "Stemming Arabic text", *Computer Science Department, Lancaster University, Lancaster, UK*, 1999.

[57]. Khreisat L., "A machine learning approach for Arabic text classification using N-gram frequency statistics", *Journal of Informetrics, Elsevier*, 3(1), pp. 72-77, 2009.

[58]. Larkey L, Ballesteros L, Connell M., "Light Stemming for Arabic Information Retrieval", *Arabic Computational Morphology, book chapter, Springer,* 2007.

[59]. Lewis, D., Gale, W., "A sequential algorithm for training text classifiers", *In the Proc. of the 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, New York, Springer*, pp. 3–12, 1994.

[60]. Lewis, D., Ringuette M., "A comparison of two learning algorithms for text categorization", *In the 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV*, 1994.

[61]. Lewis, D., "Naïve Bayes at forty: The independence assumption in information retrieval", In J.G. Carbonell & J. Siekmann (Eds.), Lecture Notes in Computer Science, Vol. 1398: Machine Learning: Proc. of ECML-98 European Conf. on Machine Learning, pp. 4–15. Berlin: Springer. 1998.

[62]. Manning, D., Raghavan, P., & Schütze, H., "An introduction to information retrieval", *Cambridge, England: Cambridge University Press*, 2006.

[63]. Mccallum A., Nigam K., "A Comparison of Event Models for Naive Bayes Text Classification", *In AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[64]. McCallum, A., Nigam, K., "A comparison of event models for naïve Bayes text classification", *In AAAIWorkshop on Learning for Text Categorization*. 1998

[65]. Mesleh A., "Chi Square Feature Extraction Based SVMs Arabic Language Text Categorization System", *Journal of Computer Science*, 3(6), pp. 430-435, 2007.

[66]. Mesleh A., "Support Vector Machines based Arabic Language Text Classification System: Feature Selection Comparative Study", *In the 12th WSEAS Int. Conf. on APPLIED MATHEMATICS, Cairo, Egypt*, 2007.

[67]. Mierswa I., Wurst M., Klinkenberg R., Scholz M., Euler T., "YALE: Rapid Prototyping for Complex Data Mining Tasks", *in the Proc. of the 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining KDD-06*, 2006.

[68]. Motaz K. Saad, "Open Source Arabic Language and Text Mining Tools", (2010, August), [Online]. Available: http://sourceforge.net/projects/ar-text-mining

[69]. Mustafa S., Al-Radaideh Q., "Using N-grams for Arabic text searching", *Journal of the American Society for Information Science and Technology*, 55 (11), pp. 1002–1007, 2004.

[70]. Peng F., Huang X., Schuurmans D., Wang S., "Text Classification in Asian Languages without Word Segmentation", *In Proc. of the 6th Int. Workshop on Information Retrieval with Asian Languages (IRAL 2003), Association for Computational Linguistics, Sapporo, Japan,* 2003.

[71]. Quinlan R., "C4.5: Programs for Machine Learning", *Morgan Kaufmann Publishers, San Mateo, CA*. 1993.

[72]. Rennie J., Shih L., Teevan J., Karger D., "Tackling the Poor Assumptions of Naive Bayes Text Classifiers", *In the Proc. of ICML*, pp. 616-623, 2003.

[73]. Said D., Wanas N., Darwish N., Hegazy N., "A Study of Arabic Text preprocessing methods for Text Categorization", *In the 2nd Int. conf. on Arabic Language Resources and Tools, Cairo, Egypt*, 2009.

[74]. Salton G., Buckley C., "Term weighting approaches in automatic text retrieval", *In the Proc. of information processing & management,* 24(5), pp. 513-523, 1998.

[75]. Savoy, J., Rasolofo, Y., "Report on the TREC-11 experiment: Arabic, named page and topic distillation searches", *TREC-11,* 2002.

[76]. Sawaf H., Zaplo J., Ney H., "Statistical Classification Methods for Arabic News Articles", *In the Workshop on Arabic Natural Language Processing, ACL'01, Toulouse, France,* 2001.

[77]. Sawalha, M, Atwell, E., "Comparative evaluation of Arabic language morphological analyzers and stemmers", *In the Proc. of COLING'2008 22nd Int. Conf. on Computational Linguistics*, 2008.

[78]. Sebastiani, F., "Machine learning in automated text categorization", *ACM Computing Surveys*, 34(1), pp. 1–47, 2002.

[79]. Silvatt, C. & Ribeirot, B., "The importance of stop word removal on recall values in text categorization", *In the Proc. of the IEEE Int. Joint Conf. on Neural Networks IJCNN'2003, Portland, Oregon, USA*, Vol. 3, pp. 1661–1666, 2003.

[80]. Song, F., Liu, S., & Yang, J., "A comparative study on text representation schemes in text categorization", *Journal of Pattern Analysis & Applications*, 8(1-2), pp. 199–209, 2005.

[81]. Su J., Zhang H., Ling C., Matwin S., "Discriminative Parameter Learning for Bayesian Networks", *In the Proc. of ICML'2008*, 2008.

[82]. Syiam M., Fayed Z., Habib M., "An Intelligent System for Arabic Text Categorization", *In IJICIS*, 6(1), pp. 1-19, 2006.

[83]. Taghva, K., Elkhoury, R., Coombs, J., "Arabic stemming without a root dictionary", *Information Technology: Coding and Computing, ITCC*, Vol. 1, pp. 152 – 157, 2005.

[84]. Taira H., Haruno M., "Feature selection in SVM text categorization", *In the Proc. of the 16th national conf. on Artificial intelligence*, pp. 480 - 486, 1999.

[85]. Thabtah F., Hadi W. Musa, Al-shammare G., "VSMs with K-Nearest Neighbour to Categorize Arabic Text Data", *in the Proc. of the World Congress on Engineering and Computer Science, WCECS'2008, San Francisco, USA*, 2008.

[86]. Witten I., Frank E., (2005), "Data Mining: Practical machine learning tools and techniques", (2nd Ed), *Morgan Kaufmann, San Francisco*.

[87]. An NSF Workshop: Language Engineering for Students and Professionals Integrating Research and Education, (2010, August), [Online]. Available: www.clsp.jhu.edu/ws99/projects/mt

[88]. W0027: An-Nahar Newspaper Text Corpus, (2010, August), [Online]. Available: www.elda.fr/cata/text/W0027.html

[89]. W0030: Arabic Data Set, (2010, August), [Online]. Available: www.elda.fr/cata/text/W0030.html

[90]. Khoja S., "An RSS Feed Analysis Application and Corpus Builder", (2010, August), [Online]. Available: www.elda.org/medar-conference/pdf/73.pdf

[91]. K.U.Leuven Language Institute, (2010, August), [Online]. Available: http://ilt.kuleuven.be/arabic/ARAB/indexARAB.php

[92]. Arabic Newswire Part 1, (2010, August), [Online]. Available: www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2001T55

[93]. Arabic Gigaword, (2010, August), [Online]. Available: www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T12

[94]. CALLFRIEND Egyptian Arabic, (2010, August), [Online]. Available: www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC96S49
[95]. CALLHOME Egyptian Arabic Speech, (2010, August), [Online]. Available: www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC97S45
[96]. Nijmegen Corpus, (2010, August), [Online]. Available: www.let.kun.nl/wba/Content2/1.4.5_Nijmegen_Corpus.htm
[97]. Essex Arabic Summaries Corpus, (2010, August), [Online]. Available: privatewww.essex.ac.uk/~melhaj/easc.htm
[98]. ARABIC WORD FREQUENCY COUNTS, (2010, August), [Online]. Available: qamus.org/wordlist.htm
[99]. www.sites.univ-lyon2.fr/langues_promodiinar/Accueil.htm
[100]. Xu J., Fraser A., Weischedel R., "Empirical studies in strategies for Arabic retrieval", *In SIGIR'02, Tampere, Finland*, 2002.
[101]. Zhang H., "The Optimality of Naïve Bayes"*, In FLAIRS2004 conference,* 2004.
[102]. Zhu, M., Zhu, J., & Chen, W., "Effect analysis of dimension reduction on support vector machines*", In the Proc. of the Natural Language Processing and Knowledge Engineering IEEE NLP-KE,* Wuhan, China, pp. 592–596, 2005.