SUPPORT VECTOR MACHINE TEXT CLASSIFIER FOR ARABIC ARTICLES: ANT COLONY OPTIMIZATION-BASED FEATURE SUBSET SELECTION

BY

ABDELWADOOD MOHAMMAD ABDELWADOOD MESLEH

THESIS ADVISOR:

DR. GHASSAN KANAAN

A THESIS SUBMITTED TO

THE FACULTY OF INFORMATION SYSTEMS & TECHNOLOGY

OF THE ARAB ACADEMY FOR BANKING AND FINANCIAL SCIENCES

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COMPUTER INFORMATION SYSTEMS

2008
AUTHORIZATION

I hereby declare that I am the sole author of the thesis.

I authorize the Arab Academy for Banking and Financial Sciences to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize the Arab Academy for Banking and Financial Sciences to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

____________________________________

ABDELWADOOD MOHAMMAD ABDELWADOOD MESLEH

JUNE 2008
بسم الله الرحمن الرحيم
الأكاديمية العربية للعلوم المالية والمصرفية
كلية نظم وتقنية المعلومات
النظام: 3/6/2008

قرار لجنة مناقشة رسالة دكتوراه رقم (18) لسنة 2008

اجتمعت اللجنة المكلفة بمناقشة أطروحة الدكتوراه في نظم المعلومات الحاسوبية المقدمة من الطالب
عبدالودود مصلح بعنوان:

Support Vector Machine Text Classifier for Arabic Articles: Ant Colony Optimization-Based Feature Subset Selection

اليوم الثلاثاء الموافق 3/6/2008 الساعة العاشرة والنصف صباحاً بٍرئيسة الدكتور رياض الشلبي
وحضور أعضاء اللجنة السادة:

1. الاستاذ الدكتور طه العريف
2. الدكتوربة بيان أبو شاور

وبعد المناقشة توصي اللجنة بنتيجة:

ناجح

اعتماد اللجنة:

1. الاستاذ الدكتور طه العريف
2. الدكتور رياض الشلبي
3. الدكتوربة بيان أبو شاور
ACKNOWLEDGMENTS

The defense of Ph.D. thesis is one of the many significant transitions in my life. This book represents the outcome of past years research efforts and is the summary of my contributions in Arabic text classification tasks. This research becomes true because of many people who have accompanied me during the whole or part of this research. Therefore, is this also the occasion to express my warm thank to all of them and to acknowledge their help.

With all my heart, I pray to Allah to bless the soil of my beloved father, who will remain my inspiration forever. In the same breath, I deeply thank my mother, wife and kids.

I very warmly thank my thesis advisor Dr. Ghassan Kanaan for the support during this work, for providing the necessary scientific freedom to carry out this research, for his patience during this time, for making this research more enjoyable, for taking the time to read and to advise me on this work.

I am grateful to Professor Martha W. Evens, who has been of invaluable help in the later stages of writing this thesis. This thesis would not be the way it is without her valuable comments.

Thanks to Dr. James Tin-Yau Kwok for introducing machine learning and kernel methods. Many thanks to Professor Raed Abu Zitar for introducing Ant Colony Optimization.

I am also indebted to Dr. Mahmoud Al Omari, and Dr. Bassam Hammo for the comments and discussions on my first proposal; their discussions have sharpened my understanding
of different research issues and provided new points of view and directions for my research.

I am also indebted to Professor Taha Ibrahim Elarief and Dr. Bayan Abu Shawar to be my thesis examination committee members.

I am grateful to Dr. Riyad AL-Shalabi for the comments and discussions on my first proposal. Thanks to him to be my thesis examination committee chairman.

I acknowledge the efforts from all the faculty members who have taught me in the Faculty of Information Systems & Technology, especially, Professor Asim EL Sheikh, Dr. Khalid Khanfer and Dr. Khalil El-Hindi.

I would like to acknowledge the constant encouragement, support and help received from my very dear friend Dr. Bassam Mahasneh.

Finally, I am grateful to my colleagues in the Arab Academy for Banking and Financial Sciences. I am also grateful to my colleagues in the Faculty of Engineering Technology, especially, Dr. Tariq Al-Mugrabi, MEng. Belal Ayyoub, MEng. Bilal Zahran and Nawal Al_Zaban for the continuous support and help.
DEDICATION

I wish to dedicate this dissertation to my beloved and cherished father, Mohammad Abdelwadood Hassan Mesleh (July 7, 1937 – April 6, 2007), who lived a life of dignity, courage, wisdom and patience, and who will remain my personal hero and my inspiration forever.

May Allah bless my father’s soul, Amen.
TABLE OF CONTENTS

Title Page ........................................................................................................................................ i
Authorization .................................................................................................................................... ii
Signature Page ................................................................................................................................. iii
Acknowledgments .......................................................................................................................... iv
Dedication ........................................................................................................................................... vi
Table of Contents ........................................................................................................................... vii
List of Figures .................................................................................................................................... x
List of Tables ...................................................................................................................................... xi
Abstract ............................................................................................................................................ xii
Chapter 1: Introduction ...................................................................................................................... 1
  1.1 Research Contributions ............................................................................................................. 2
  1.2 Thesis Layout and Brief Overview of Chapters ........................................................................ 3
Chapter 2: Related Work .................................................................................................................... 5
  2.1 Arabic Text Classification Research ....................................................................................... 5
  2.2 Feature Subset Selection for Arabic Text Classification .......................................................... 9
Chapter 3: Text Classification .......................................................................................................... 10
  3.1 Document Pre-processing ......................................................................................................... 11
  3.2 Text Classifier Construction ..................................................................................................... 12
    3.2.1 Support Vector Machine Classifier ............................................................................... 13
    3.2.2 $K$-Nearest Neighbors Classifier ............................................................................... 21
    3.2.3 Naïve Bayes Classifier ................................................................................................. 22
    3.2.4 Other Text Classifiers .................................................................................................... 24
    3.2.5 Text Classifier Comparison and Discussion .................................................................... 24
  3.3 Document Classification ............................................................................................................ 25
Chapter 4: Feature Subset Selection .................................................................................................. 26
6.1 Ant Colony Optimization Algorithm .................................................................61
  6.1.1 Ant Colony Optimization Flavors ..................................................................63
  6.1.2 Applications of Ant Colony Optimization ...................................................64
6.2 Ant Colony Algorithms for Feature Subset Selection ........................................65
6.3 Steps to Solve a FSS Problem by Ant Colony Optimization Algorithm ............65
6.4 Ant Colony Optimization Based Feature Subset Selection Steps .....................68
6.5 Proposed Ant Colony Optimization-Based Feature Subset Selection Steps ......69

Chapter 7: Experimental Results ............................................................................75
  7.1 Arabic Data Collection ....................................................................................75
  7.2 Arabic Dataset Pre-processing .........................................................................76
  7.3 Text Classification Methods ...........................................................................77
  7.4 Feature Subset Selection ..................................................................................78
  7.5 Text Classification Evaluation .........................................................................78
  7.6 Text Classification Experimental Results .......................................................80
    7.6.1. SVM Classifier with Chi-square FSS Experiments .................................80
    7.6.2. Light Stemming Effect on the SVM Classifier with Chi-square FSS
          Experiments .................................................................................................82
    7.6.3. Comparing the SVM Classifier with kNN and Naïve Bayes classifiers ......82
    7.6.4. FSS Comparison Study with the SVM Classifier for Arabic Articles .......83
  7.7 Ant Colony Optimization Based FSS Results ................................................85

Chapter 8: Conclusions & Future Directions ...........................................................91

Bibliography ...........................................................................................................93
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Number</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.1: Text Classification Architecture</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3.2: Maximal Margin Hyperplane with Support Vectors highlighted</td>
<td>14</td>
</tr>
<tr>
<td>Figure 4.1: Feature Subset Selection Process</td>
<td>28</td>
</tr>
<tr>
<td>Figure 4.2: Search Strategies</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4.3: GA process</td>
<td>51</td>
</tr>
<tr>
<td>Figure 5.1: Arabic words Categorization</td>
<td>56</td>
</tr>
<tr>
<td>Figure 6.1: Ant Colony Optimization Representation for Feature Subset Selection</td>
<td>70</td>
</tr>
<tr>
<td>Figure 7.1: Macro-averaging precision values for SVM classifier with the six FSS methods at different sizes of features</td>
<td>84</td>
</tr>
<tr>
<td>Figure 7.2: Macro-averaging recall values for SVM classifier with the six FSS methods at different sizes of features</td>
<td>85</td>
</tr>
<tr>
<td>Figure 7.3: Macro-averaging $F_1$ values for SVM classifier with the six FSS methods at different sizes of features</td>
<td>86</td>
</tr>
<tr>
<td>Figure 7.4: Macro-averaging Precision values for SVM classifier with the seven FSS methods at different sizes of features</td>
<td>88</td>
</tr>
<tr>
<td>Figure 7.5: Macro-averaging Recall values for SVM classifier with the seven FSS methods at different sizes of features</td>
<td>89</td>
</tr>
<tr>
<td>Figure 7.6: Macro-averaging $F_1$ measure values for SVM classifier with the seven FSS methods at different sizes of features</td>
<td>90</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Number</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4.1:</td>
<td>Comparison of search strategies</td>
<td>33</td>
</tr>
<tr>
<td>Table 4.2:</td>
<td>Filter FSS and Wrapper FSS comparison</td>
<td>37</td>
</tr>
<tr>
<td>Table 4.3:</td>
<td>Two-way Contingency Table.</td>
<td>39</td>
</tr>
<tr>
<td>Table 5.1:</td>
<td>Identical Arabic Words with different forms</td>
<td>59</td>
</tr>
<tr>
<td>Table 5.2:</td>
<td>Some Arabic words derived from the same root “ktb”</td>
<td>59</td>
</tr>
<tr>
<td>Table 5.3:</td>
<td>Four possible roots for the word “AymAn”</td>
<td>60</td>
</tr>
<tr>
<td>Table 7.1:</td>
<td>TC Arabic Dataset.</td>
<td>75</td>
</tr>
<tr>
<td>Table 7.2:</td>
<td>SVM results with Chi-square feature subset selection.</td>
<td>81</td>
</tr>
<tr>
<td>Table 7.3:</td>
<td>Different Classifiers’ performance in Arabic TC tasks.</td>
<td>82</td>
</tr>
<tr>
<td>Table 7.4:</td>
<td>SVM Performance Comparison Using Best 160 Features: Selected by Different FSS methods.</td>
<td>90</td>
</tr>
</tbody>
</table>
SUPPORT VECTOR MACHINE TEXT CLASSIFIER FOR ARABIC ARTICLES: ANT COLONY OPTIMIZATION BASED-FEATURE SUBSET SELECTION

By

ABDELWADOOD MOHAMMAD ABDELWADOOD MESLEH,

Doctor of Philosophy, 2008

ABSTRACT

In this thesis, we have implemented a support vector machine (SVM) text classifier for Arabic articles. Experimental results show that the SVM classifier outperformed Naïve Bayesian (NB) and \( k \)-nearest neighbor (kNN) classifiers. We investigated the effectiveness of six state-of-the-art feature subset selection (FSS) methods, which are commonly used in text classification (TC) tasks, for our Arabic SVM text classification system. We implemented an Ant Colony Optimization Based-Feature Subset Selection (ACO Based-FSS) method for our Arabic SVM text classifier. The proposed FSS method adapted Chi-square statistic as heuristic information and the effectiveness of the SVM classifier as a guide to improving the selection of features for each category. Compared to the six state-of-the-art FSS methods, our ACO Based-FSS algorithm
achieved better TC effectiveness. Evaluation used an in-house Arabic TC corpus\(^1\) that consists of 1445 documents independently classified into nine categories. The experimental results were presented in terms of macro-averaging precision, macro-averaging recall and macro-averaging F\(_1\) measures.

\(^1\) This TC corpus was collected by my thesis advisor, Dr. Ghassan Kanaan.
CHAPTER 1: INTRODUCTION

It is known that the volume of information available in Arabic on the Internet is increasing. This growth motivates researchers to develop tools that may help people to better manage, filter and classify these huge Arabic information resources. Internet users need a tool to help them sort incoming email messages into folders and to delete junk email; these are examples of text classification systems.

Text Classification (TC) (Manning & Schütze, 1999; Sebastiani, 2002) is the task of classifying texts into one of a pre-specified set of categories or classes based on their contents. It is also referred as text categorization, document categorization, document classification, or topic spotting. It is wise, however, to distinguish between text classification and text categorization. Text classification is the term for a machine learning process that sorts documents by their contents, while text categorization is another broader term for a machine learning process that includes any type of assignment of documents to classes, not necessarily based on their contents (Jackson & Moulinier, 2002).

TC is among the many important research problems in information retrieval (IR), data mining, and natural language processing. It has many applications (Sebastiani, 2002) such as document indexing (Biebricher et al., 1988), text filtering (Schütze, Hull & Pedersen, 1995), and word sense disambiguation (Schütze, 1998).

Despite the growth of Arabic text, there is a lack of work in the Arabic TC field, and those who have investigated this field have paid most attention to inventing or to enhancing text classifiers.

In this thesis, however, we have restricted our study of TC to binary classification methods and, in particular, to Support Vector Machine (SVM) classifiers and only for

1
Arabic articles. We have focused on feature subset selection (FSS) methods for TC tasks, and, in particular, to the usage of an ant colony optimization algorithm to optimize the process of FSS. To prove the effectiveness of our proposed Ant Colony Optimization Based-Feature Subset Selection Method, we have compared it with six state-of-the-art feature subset selection methods: Chi-square, Information Gain, Mutual Information, the NGL\(^2\) coefficient, the GSS\(^3\) score and the Odds Ratio in the classification of Arabic articles. Our proposed ACO-Based FSS algorithm performed best.

1.1 Research Contributions

This thesis contributes the following to the field of Arabic TC tasks:

- This thesis presented a support vector machine text classifier for Arabic articles (Mesleh, 2007a; Mesleh, 2007b).
- This thesis (Mesleh, 2007c; Mesleh, 2007d) presented an extensive comparative study of feature subset selection methods for Arabic text classification tasks, focusing on support vector machine classifier.
- Despite the usage of Ant Colony Optimization (ACO) in Feature Subset Selection (FSS) processes, the proposed ACO Based-FSS method is among the few feature subset selection methods that handle high dimensional data sets. ACO algorithm has been used to optimize the FSS process in TC tasks in general and in Arabic TC tasks, in particular.
- The unique tailoring of ACO algorithm for FSS in a high dimensional TC feature space, i.e., the unique pheromone update method, is achieved by combining two ant colony algorithm flavors: Ant System (AS) (Dorigo, Maniezzo & Colorni, 1996) and Elitist Ant System (EAS) (Dorigo, Maniezzo & Colorni, 1996) to produce the

\(^2\) NGL is named after the authors of (Ng, Goh, & Low, 1997).
flavor of FSS for TC tasks. On the other hand, we have proposed an FSS optimization approach that is applicable to poor classification effectiveness categories. Our proposed ACO-Based FSS method, however, can be applied to all categories.

1.2 Thesis Layout and Brief Overview of Chapters

This thesis comprises eight chapters briefly described as follows:

- Chapter 2 briefly surveys earlier work in the field of Arabic Text Classification. We look at two different components: the history of classifiers that have been used in Arabic text classification tasks, and then we describe earlier work on feature subset selection methods that have been implemented to handle the Arabic text classification task.

- Chapter 3 describes the Text Classification process. This chapter describes the text preprocessing phase, the text classifier construction phase, and the text classification phase (the ability of the system to classify unseen documents into one of a predefined set of categories).

- Chapter 4 presents Feature Subset Selection concepts, steps, approaches, enhancements, and optimization methods.

- Chapter 5 presents an introduction to Arabic language.

- Chapter 6 describes our proposed Ant Colony Optimization Based-Feature Subset Selection Method for text classification tasks.

- Experimental results are discussed in Chapter 7.

- Finally Chapter 8 draws conclusions and presents ideas for future work. We render a close look at the limitations of the new ACO-Based FSS, which is a main

---

3 GSS is named after the authors of (Galavotti, Sebastiani, & Simi, 2000).
contribution of the thesis. Finally, we conclude with a peek into the future with some suggested directions.
CHAPTER 2: RELATED WORK

Text Classification research has received much attention (Sebastiani, 2002). There are two main approaches for TC: the rule-based approach and the machine learning approach; in the rule-based approach, classification rules are manually created by experts; this method is accurate but expensive. On the other hand, in the machine learning approach, classification rules are automatically created; the result is a cost saving and it is easy to construct a system for a new domain.

TC has been studied using a binary classification approach (a binary classifier is designed for each category of interest). A lot of TC training algorithms have been reported in binary classification, e.g., the Naïve Bayesian method (Lewis & Ringuette, 1994; McCallum & Nigam, 1998; Yang & Liu, 1999), k-Nearest Neighbors (Guo et al., 2004; Yang, 1994), support vector machine (Joachims, 1998), decision trees (Lewis & Ringuette, 1994), etc. On the other hand, TC has been studied as a multi-classification approach, e.g., boosting (Schapire & Singer, 2000), and multi-class support vector machine (Gao et al., 2006).

Most of the TC research has been implemented, designed, and tested with English language articles (Joachims, 1998; Schapire & Singer, 2000; Gao et al., 2006). However, a number of TC learning approaches were carried out for other European languages such as German, Italian and Spanish (Ciravegna et al., 2000), and some other approaches were carried out for the Chinese and Japanese languages (Peng et al., 2003).

2.1 Arabic Text Classification Research

There is only a little work (Samir, Ata & Darwish, 2005) carried out for Arabic articles classification and there is only one commercial automatic Arabic text categorizer,
referred as the *Sakhr Categorizer*\(^4\). Compared to Indo-European languages (like English), the Arabic language (see Chapter 4) has an extremely rich morphology and a complex orthography; this is one of the main reasons (Samir, Ata & Darwish, 2005; El-Halees, 2007) behind the lack of research in the field of Arabic text classification. However, many machine learning approaches have been proposed to classify Arabic documents: Support Vector Machine classifier with the Chi-square feature extraction method (Mesleh, 2007a; Mesleh, 2007b)\(^5\), the Naïve Bayesian method (Elkourdi, Bensaid & Rachidi, 2004), \(k\)-Nearest Neighbors (Kanaan, Al-Shalabi & Al-Akhras, 2006; Al-Shalabi, Kanaan & Gharaibeh, 2006; Syiam, Fayed & Habib, 2006), maximum entropy (El-Halees, 2007; Sawaf, Zaplo & Ney, 2001), distance based classifiers (Khreisat, 2006; Duwairi, 2005, 2006), the Rocchio Algorithm (Syiam, Fayed & Habib, 2006), etc.

Sawaf, Zaplo and Ney (Sawaf, Zaplo & Ney, 2001) have used the maximum entropy method for Arabic document clustering. Initially, documents were randomly assigned to clusters. In subsequent iterations, documents were shifted from one cluster to another if an improvement was gained. The algorithm terminated when no further improvement could be achieved. Their text classification method is based on unsupervised learning.

El-Kourdi, Bensaid, and Rachidi (El-Kourdi, Bensaid & Rachidi, 2004) have used a Naïve Bayesian classifier to classify an in-house collection of Arabic documents, and have concluded that there is some indication that the performance of Naïve Bayesian algorithm in classifying Arabic documents is not sensitive to the Arabic root extraction algorithm, in addition to their own root extraction algorithm, they used other root extraction algorithms such as (Baeza-Yates & Ribeiro-Neto, 1999; Al-Shalabi & Evens, 1998).

\(^5\) To our best knowledge, this research is the first published Arabic TC work that used the SVM classifier.
Samir, Ata and Darwish (Samir, Ata & Darwish, 2005) have used a text classification algorithm which is based on a light stemming algorithm which removes suffixes and prefixes from the Arabic words.

Al-Shalabi, Kanaan and Gharaibeh (Al-Shalabi, Kanaan & Gharaibeh, 2006) have used \(k\)NN with document frequency (DF) as a feature reduction method to classify Arabic documents. In another similar study, Kanaan, Al-Shalabi and Al-Akhras (Kanaan, Al-Shalabi & AL-Akhras, 2006) have used \(k\)NN with information gain (IG) to classify Arabic documents. In their studies, the authors of the two papers have not compared their results with other classifiers or with other feature subset selection methods.

Khreisat (Khreisat, 2006) has presented a classifying algorithm for Arabic text documents using the N-gram\(^6\) frequency statistics technique. She used Manhattan distance and Dice measure as similarity measures. She has shown that the N-gram text classification method using the Dice measure outperformed classification using the Manhattan measure.

Duwairi (Duwairi, 2006) has proposed a distance-based classifier for Arabic TC tasks, where the Dice measure was used as a similarity measure. In her work, each category was represented as a vector of words. In the training phase, her text classifier scanned training documents to extract features that best capture inherent category specific properties. Documents were classified on the basis of their closeness to the feature vectors of the text.

El-Halees (El-Halees, 2007) has implemented a maximum entropy based classifier to classify Arabic documents. Compared with other text classification systems (such as El-Kourdi et al. and Sawaf et al.), the overall performance of his system was good (in his comparisons, he used the results as recorded in their published papers).

\(^6\) In her N-gram study, N was set to 3.
In Arabic natural language processing, a few publications have used SVM classifiers. The author of this thesis has used SVM classifier with the Chi-square FSS method (Mesleh, 2007a; Mesleh 2007b) to classify Arabic documents. This SVM classifier has outperformed Naïve Bayesian and $k$NN classifiers. The author of this thesis has studied the effect of light stemming\(^7\) on Arabic TC tasks with SVM and concluded that light stemming is not beneficial for Arabic TC tasks.

Al-Harbi and his colleagues (Al-Harbi et al., 2008) have evaluated the performance of SVM and C5.0 decision tree algorithm classifiers for Arabic TC tasks. They have presented the classification results on seven different Arabic corpora, and concluded that the C5.0 decision tree algorithm outperformed SVM in term of accuracy. However, the authors have not considered other performance measures such as precision, recall, and the $F_1$ measure.

Hmeidi, Hawashin and El-Qawasmeh (Hmeidi, Hawashin & El-Qawasmeh, 2008) reported a comparative study of SVM and $k$NN classifiers on Arabic TC tasks, and concluded that SVM classifier shown a better micro-averaging $F_1$ measure.

It is quite hard to fairly compare the performance of these approaches for the following reasons:

- They have used different TC corpora because there is no publicly available Arabic TC corpus.
- Even when scholars have used the same corpus, it is not obvious whether they have used the same documents for training/testing their classifiers or not.
- They have used different evaluation measures: accuracy, recall, precision, the $F_1$ measure and other measures.

---

\(^7\) The algorithm of light stemming was adapted from (Larkey, Ballesteros & Connell, 2002), in which only prefixes and suffixes were removed from the Arabic terms.
2.2 Feature Subset Selection for Arabic Text Classification

Syiam, Fayed and Habib (Syiam, Fayed & Habib, 2006)\(^8\) have evaluated the effect of many FSS methods (Chi-square, DF, IG, OR, GSS score, and NGL coefficient) for Arabic TC tasks with Rocchio and \(k\)NN classifiers. For evaluation they used a hybrid approach of light and trigram stemming. It has been shown that the use of any of those FSS methods separately gave close results, however it has been shown that NGL performed better than DF, Chi-square and GSS in term of the \(F_1\) measure. They have concluded that a hybrid approach of DF and IG is a preferable FSS method for the Arabic TC task. But, they have not reported the comparison results of all the mentioned FSS methods in terms of recall, precision, and \(F_1\) measure, and they have not considered SVM which was already known to be superior to the classifiers they have studied.

The author of this thesis, Mesleh (Mesleh, 2007c; Mesleh, 2007d) has investigated the effect of (Chi-square, DF, IG, OR, the GSS score, and the NGL coefficient) with SVM classifier for the Arabic TC task and concluded that Chi-square works best in terms of macro-averaging recall and the macro-averaging \(F_1\) measure. However, Chi-square and NGL work best in term of macro-averaging precision.

Most of the other Arabic TC papers (Al-Harbi et al., 2008; Hmeidi, Hawashin & El-Qawasmeh, 2008; Al-Shalabi, Kanaan & Gharabeh, 2006; Kanaan, Al-Shalabi & Al-Akhras, 2006; Mesleh, 2007a; Mesleh, 2007b) have used some FSS method but have not included any FSS comparison.

\(^8\) As a matter of fact, their study has investigated the effect of stemming techniques on Arabic text classifiers, and has concluded that a hybrid approach of light and statistical stemmers is the most suitable for Arabic text categorization tasks.
CHAPTER 3: TEXT CLASSIFICATION

The main goal of text classification (TC) systems is the classification of documents into a fixed number of predefined classes (Joachims, 1998). Each document may belong to multiple classes, one class, or none. Supervised learning (Moens, 2006) is a very popular machine learning approach, in which, TC algorithms learn classification patterns from a set of labeled examples, given a large enough number of labeled examples (training set), and the task is to build a TC model. Then we can use the TC model to predict the category (class) of new unseen examples (test set). The architecture of a TC system (Figure 3.1) usually comprises the following three main phases (Guo et al., 2004):

- Document pre-processing phase.
- Text classifier construction phase.

Figure 3.1: Text Classification Architecture
- Document classification phase

The document pre-processing phase represents text documents in a compact and an applicable form to train the text classifier. The text classifier construction phase, the core TC learning algorithm, constructs the categories, learns them, and is tuned using the compact form of the text documents. The document classification phase implements the function of classifying new documents.

3.1 Document Pre-processing

Document pre-processing in TC tasks usually comprises the following steps (Benkhalifa, Mouradi & Bouyakhf, 2001; Guo et al., 2004; Elkourdi, Bensaid & Rachidi, 2004):

- Because TC documents are available in different file formats such as SGML, HTML, XML, etc., they are converted to plain text format.
- Each document is processed to remove digits and punctuation marks.
- Relevant text is extracted and segmented into tokens; each token can be a single term (word) or a multi-term phrase (bigram, trigram, ...).
- Stop words are removed. (Stop words are the words that are not useful in information retrieval systems such as pronouns and prepositions).
- Tokens are optionally normalized by stemming or lemmatization.

---

9 SGML: Standard Generalized Markup Language.
10 HTML: Hyper Text Markup language.
11 XML: Extensible Markup language.
12 Stemming is (Manning & Schütze, 1999) a process that strips off affixes and leaves a stem.
13 Lemmatization is (Manning & Schütze, 1999) a process that attempts to find the lemma or lexeme of which one is looking at an infected form, where lexeme is a word that is a meaningful unit in some language and coincides with an abstract unit underling a given set of infected forms.
Some additional filtering can be applied, such as removal of infrequent terms.

Vector space representation\(^{14}\) (Salton, Wong & Yang, 1975) is used to represent text articles. In the vector space model (VSM), Term frequency \((TF_{ij})\) measures the number of times a term \(i\) occurs in document \(j\) while inverse document frequency \((IDF)\) measures the term occurrence in a collection of texts and is calculated by \(IDF(i) = \log\left(\frac{N}{DF(i)}\right)\), where \(N\) is the total number of training documents and \(DF\) is the number of documents that a term \(i\) occurs in.

In information retrieval, it is known that \(TF\) makes the frequent terms more important. As a result, \(TF\) improves recall (see Chapter 7 for the definition of recall). On the other hand, the inverse document frequency \(IDF\) makes the terms that occur rarely in a collection of texts more important. As a result, \(IDF\) improves precision (see Chapter 7 for the definition of precision). Using VSM (Salton & Buckley, 1988) while combining \(TF\) and \(IDF\) to weight terms \((IDF.TF)\) has been shown to give a better performance. Then each document feature vector is normalized to unit length and the \(IDF.TF\) is calculated.

Because of the high dimensional feature space in TC, a feature subset selection algorithm (see Chapter 4) can be applied to make the TC learning algorithm cost effective & more accurate, and to avoid “overfitting” problem.

### 3.2 Text Classifier Construction

Like any classification algorithm, TC algorithms have to be robust and accurate. Many TC methods (Mitchell, 1996) have been investigated in the literature, such as k-Nearest Neighbors \((kNN)\), Naïve Bayes \((NB)\) and Support Vector Machine \((SVM)\).

\(^{14}\) Vector Space Model is commonly used to represent text articles; however, some other approaches have been used in literatures to represent text articles such as the string kernel method (Lodhi et al., 2002).
In the following sections, SVM, kNN and NB classifiers are presented.

3.2.1 Support Vector Machine Classifier

Support Vector Machine (SVM) classifiers are binary classifiers, which were originally proposed by (Vapnik, 1995; Vapnik, 1998; Burges, 1998; Scholkopf & Smola, 2002).

Support Vector Machine Classifier Formulation

Given a set of training examples \( \{(x_i, y_i)\}_{i=1}^{l} \) with input data \( x_i \in \mathbb{R}^d \) and the corresponding binary class label \( y_i \in \{-1, +1\} \) (in general there can be more than two classes).

Where \( \mathbb{R}^d \) is the \( d \)-dimensional Euclidian space, \( \mathbb{R} \) is the set of reals, and \( l \) is the number of training examples.

In SVM, there is a hyperplane that discriminates all the training examples correctly (this condition will be relaxed in the non-separable case).

In a 2-D space this hyperplane is a line\(^{15}\) (see Figure 3.2), we need to find \( wx + b = 0 \) that separate the two classes. We prefer to construct this hyperplane with the largest possible margin:

\[
f(x) = w.x + b
\]

Where \( w \) is the weight vector in feature space, and \( b \) is bias (constant offset or threshold).

Figure 3.2 shows a hyperplane that separates positive and negative examples and it shows the distance between the hyperplane and its nearest vectors.

---

\( ^{15} \) This hyperplane can be a plane in 3-D space.
With reference to Figure 3.2, it is clear that a two margin boundary hyperplanes are formed by the nearest positive examples and the nearest negative examples. Assume that $x_1$ is in X’s side and $x_2$ is in O’s side, then the following equations hold:

$$wx_1 + b = +1$$

and

$$wx_2 + b = -1$$

As a result, we conclude that:

$$w(x_1 - x_2) = 2$$
and

\[
\text{margin} = \frac{w}{|w|} (x_1 - x_2) = \frac{2}{|w|}.
\]

where \(|w|\) is the norm of \(w\).

Now, it is clear that maximizing the margin is equivalent to minimizing the norm of \(w\).

To insure that all the training examples are classified correctly, the following equation must hold for the nearest examples; as a result the hyperplane can be computed.

\[
w^x + b = \begin{cases} 
  \geq 1 & \text{if } y_i = 1 \\
  \leq -1 & \text{if } y_i = -1
\end{cases}
\]

Or, equivalently:

\[
y_i(x, w + b) \geq 1
\]

This is a general framework for the SVM mechanism. The SVM classifier is dealing with two different cases: the \textit{separable case} and the \textit{non-separable case}.

\textit{Support Vector Machine Separable Case}

In the separable case, the training data is linearly separable, and the norm \(|w|\) minimization is accomplished according to the following equation (separable case primal problem):

\[
\min \frac{1}{2} |w|^2 \\
s.t. \forall i, y_i(x, w + b) - 1 \geq 0
\]
where $|w| = \sqrt{\langle w, w \rangle}$ is the 2-norm of $w$, where $\langle w, w \rangle$ is the dot product between $w$ and $w$.

To solve the mentioned primal optimization problem, we derive the so-called dual problem, which can be shown to have the same solutions as the primal optimization problem.

In this case, it is more convenient to deal with the dual problem. To derive it we introduce the Lagrangian:

The Lagrangian $L$ of this primal problem is computed as follows:

$$L(w, b, \alpha) = \frac{1}{2} |w|^2 - \sum_i \alpha_i (y_i (x_i \cdot w + b) - 1)$$

where $\forall i, \alpha_i > 0$ are Lagrange multipliers.

Under the constraint $\alpha_i(\forall i)$, the Lagrange problem $L(w, b, w)$ needs to be minimized with respect to $w$ and $b$. Consequently, the derivatives of $L$ with respect to the primal variables must vanish:

$$\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0$$

Substituting the above results in the Lagrange form, we get the following:

$$L(w, b, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$$
According to the Lagrange theory, to obtain the optimum, it is enough to maximize the Lagrange with respect to $\alpha_i (\forall i)$.

We obtain a dual problem of the separable case primal problem:

$$\text{max. } \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$$

$$\text{s.t. } \sum_i \alpha_i y_i = 0,$$

$$\forall i, \alpha_i \geq 0$$

Because dual problems have quadratic forms, they can be solved more easily than primal optimization problems. A solution can be obtained by general-purpose optimization packages\(^{16}\) like LIBSVM\(^{17}\), SVMLight\(^{18}\), TinySVM\(^{19}\), etc.

To maximize this dual problem, we use $\alpha_i^* (\forall i)$, the optimal $w^*$ and $b^*$ are expressed as follows:

$$w^* = \sum_i \alpha_i y_i x_i$$

$$b^* = \frac{b^- + b^+}{2}$$

where

$$b^- = \max_{i : y_i = -1} (w^* . x_i)$$

\(^{16}\) See http://www.kernel-machines.org/software for more tools.

\(^{17}\) Available from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

\(^{18}\) Available from http://svmlight.joachims.org/

\(^{19}\) Available from http://chasen.org/~taku/software/TinySVM/
By substituting $w^*$ and $b^*$ in equation $(f(x) = w \cdot x + b)$, we obtain the following equation:

$$f(x) = \sum_i \alpha_i y_i x_i x + b^*$$

Noting that $i$ varies from 1 to $N_s$, where $N_s$ is the number of support vectors.

The sign of $f(x) = \sum_i \alpha_i y_i x_i x + b^*$ is used to classify new examples (unseen examples).

**Support Vector Machine Non-separable Case**

In the non-separable case, where data is not linearly separable, the norm $|w|$ is minimized by the following equation *(non-separable case primal problem)*:

$$\min \quad \frac{1}{2} |w|^2 + C \sum_i \xi_i,$$

s.t. $\forall i, y_i (x_i w + b) - 1 + \xi_i \geq 0,$

$\forall i, \xi_i \geq 0.$

where $\xi_i (\forall i)$ are positive slack variables, which are introduced to enable the non-separable problems to be solved (Cristianini & Shawe-Taylor, 2000), in this case we allow few examples to penetrate into the margin or even into the other side of the hyperplane (The hyperplane is separating the training data with a minimal number of errors). Moreover, the constant $C > 0$ determines the trade-off between margin maximization and training error minimization. It means that for an error to occur, the corresponding $\xi_i$ must exceed unity, and $\sum_i \xi_i$ is an upper bound on the number of

\[ b^+ = \min_{w, y_i \neq 1} (w^* \cdot x_i) \]
training errors, so we penalize $\sum_i \xi_i$ in the objective function as shown in the non-separable case primal problem.

Similarly, The Lagrangian $L$ of this non-separable case problem is:

$$L(w,b,\alpha,\xi,\mu) = \frac{1}{2} \| w \|^2 + C\sum_i \xi_i$$

$$-\sum_i \alpha_i(y_i(x_i w + b) - 1 + \xi_i) - \sum_i \mu_i \xi_i$$

where $\mu_i(\forall i)$ are Lagrange multipliers introduced to enforce positively of the $\xi_i$. The Lagrange problem $L(w,b,\alpha,\xi,\mu)$ needs to be minimized with respect to $w$, $b$ and $\xi$ under the constraints $\xi_i \geq 0$, $\alpha_i \geq 0$, $\mu_i \geq 0 (\forall i)$:

$$\frac{\partial L}{\partial w} = w - \sum_i \alpha_i y_i x_i = 0$$

$$\frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi} = C - \alpha_i - \mu_i = 0$$

Substituting the above results in the Lagrange form, we get the following:

$$L(w,b,\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$$

In Lagrange, to obtain an optimum, it is enough to maximize the Lagrange with respect to $\alpha_i(\forall i)$. We obtain a dual problem of the non-separable case primal problem:
\[
\begin{align*}
\text{max.} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\
\text{s.t.} \quad & \sum_i \alpha_i y_i = 0, \\
& \forall i, 0 \leq \alpha_i \leq C
\end{align*}
\]

where \( \alpha_i \leq C \) comes from \( \mu_i \geq 0 \) and \( C - \alpha_i - \mu_i = 0 \).

Using optimal \( \alpha^*_i (\forall i) \), the optimal \( w^* \), \( b^* \) are expressed as in the above mentioned separable case.

**Kernel Trick**

Kernel methods (Vapnik, 1995) are often combined with SVM to compensate for their limited separating ability. In kernel methods, the dot products in dual equations are replaced with a more general inner product \( K(x_i, x) \), called the kernel function. Commonly used kernels include polynomial, Radial Basis Function, Sigmoid, and Gaussian function. This means that the feature vectors are mapped into a higher dimensional space and linearly separated there. The significant advantage is that only the general inner products of two vectors are needed. This leads to a relatively small computational overhead. On the other hand, the crucial issues for SVM are choosing the right kernel function and the parameter tuning.

Why Support Vector Machine Classifiers Work Well for TC Tasks

Empirical results in Text Classification have shown that SVM classifiers perform well, simply because of the following text properties (Joachims, 1998):

- High dimensional input space: In text documents, we are dealing with a huge number of features. Since SVM classifiers use overfitting protection, which does not necessarily depend on the number of features, SVM classifiers have the potential to handle a large number of features.
- Few irrelevant features: we assume that most of the features are irrelevant to avoid these high dimensional input spaces. Feature subset selection methods try to determine these irrelevant features (in TC tasks, there are very few irrelevant features).

- Document vectors are sparse: For each document, the corresponding document vector contains only a few entries that are not zero.

- Most text classification problems are linearly separable: many of the TC categories are linearly separable, and the main idea of SVM classifiers is to find linear separators.

SVM classifiers have proved empirically to be well suited for TC. One of the main advantages of SVM classifiers over other conventional methods is their robustness.

In addition to TC tasks, SVM classifiers have achieved high accuracy in various tasks, such as object recognition (Pontil & Verri, 1998), fault detection and diagnosis (Chiang, Kotanchek & Kordon, 2004), face recognition (Dong, Li & Chan, 2001), etc.

In addition to SVM classifiers, other kernel based methods (Hofmann, 1999; Takamura, Matsumoto & Yamada, 2004) have shown empirical successes in the field of TC. Some of those methods have outperformed SVM classifiers such as the hyper plane-based TOP kernel (Takamura, Matsumoto & Yamada, 2004) and the multiclass maximal figure-of-merit learning approach (Gao et al., 2006).

3.2.2 K-Nearest Neighbors Classifier

A $k$-nearest neighbors ($k$NN) classifier (Mitchell, 1996) classifies a new document $d$ based on a similarity measure. $k$NN calculates the distance between the new document $d$ and all training examples in the training set, then selects $k$-nearest examples to $d$ in the
training set. Finally, $k$NN assigns $d$ to class $C_i$, which contains most of the neighbors. It is worth pointing out that $k$NN uses the cosine as a similarity metric.

A $k$NN is a simple TC approach, but it suffers from the computational complexity problem and the difficulty of determining the proper value of $k$.

3.2.3 Naïve Bayes Classifier

The Naïve Bayes (NB) classifier (Manning & Schütze, 1999) uses a probabilistic model of text. It (McCallum & Nigam, 1998) achieves good performance results on TC tasks. NB is called “naïve” because of the assumption of independence between features. Because of its simplicity, NB is often used in TC tasks.

NB is based on Bayes’ formula:

$$P(C_i \mid d) = \frac{P(C_i)P(d \mid C_i)}{P(d)}$$

where

- $P(C_i \mid d)$ is the posterior probability of class $C_i$ given a new document $d$.
- $P(C_i)$ is the probability of class $C_i$.
- $P(d \mid C_i)$ is the probability of a document $d$ given a class $C_i$.
- $P(d)$ is the probability of document $d$.

Notes:

- $P(C_i)$ can be calculated by $P(C_i) = \frac{N_i}{N}$, where $N_j$ is the number of articles assigned to class $C_i$, and $N$ is the number of classes.
In a TC corpus, every article has the same probability, so $P(d)$ can be eliminated from Bayes’ formula.

Because the independence assumption of NB, the $P(d \mid C_i)$ can be calculated by

$$P(C_i \mid d) = P(C_i) \prod_{k=1}^{l} P(t_k \mid C_i),$$

where $t_k$ is a feature that co-occurs with class $C_i$.

Similarly, $P(t_k \mid C_i)$ is calculated by

$$P(t_k \mid C_i) = \frac{1 + n_{ki}}{l + \sum_{h=1}^{l} n_{hi}},$$

where $n_{ki}$ is the total number of documents that contains feature $t_k$ and belongs to class $C_i$, $l$ is the total number of distinct features in all documents (training documents that belong to class $C_i$).

As a result, given a new document $d$, NB calculates the posterior probability $P(C_i \mid d)$ for each class, and then it assigns the document $d$ to the highest posterior probability’s class, i.e. $C(d) = \arg\max_{i=1}^{\mid C \mid} (P(C_i \mid d))$.

Let $d$ is a new document, $C$ is the set of all possible classes, i.e. $C = C_i$, $i = 1,...,\mid C \mid$, where $\mid C \mid$ is the total number of classes.

Begin:
For each class $C_i$

- NB calculates an estimate of $P(C_i \mid d)$;
- NB Classifies $d$ to $C_i$ with highest estimate;

End;

NB has been applied in TC tasks using two flavors:

- Multivariate Bernoulli Model.
- Multinomial Model.
The Multivariate Model considers the presence or absence of document words, on the other hand, the Multinomial Model considers the frequency of words.

Generally, the Multivariate Model performs well with small datasets. However, the Multinomial Model shows better performance than the Multivariate Model (McCallum & Nigam, 1998).

3.2.4 Other Text Classifiers

Many other text classifiers have been used for text document classification. In addition to SVM, Naïve Bayes and kNN classifiers, Sebastiani in his comprehensive survey (Sebastiani, 2002) summarized the following text classification methods:

- Regression methods such as the Linear Least Squares Fit (Yang & Chute, 1994).
- Online learning methods such as Perceptrons (Dagan, Karov & Roth, 1997).
- Rocchio method (Schapire, Singer & Singhal, 1998).
- Neural Networks (Ruiz & Srinivasan, 1999).
- Ensemble learning methods (Scott & Matwin, 1999).
- Decision Trees (Cohen & Singer, 1999).

3.2.5 Text Classifier Comparison and Discussion

Many empirical studies have compared text classification methods on English text (Joachims, 1998). Naïve Bayes and support vector machine are always in the comparison list. The Naïve Bayes classifier is considered as a simple and effective classifier (Mitchell, 1996) and it is often used as a baseline text classifier. On the other hand, the support vector machine classifier has been among the best-performing text classifiers (Joachims, 1998).
3.3 Document Classification

After the text classifier has been evaluated, it can be used to implement the function of document classification. The evaluation process uses some performance measures (see Chapter 7) such as precision, recall, and the $F_1$ measure. The documents to be classified have to be pre-processed as in text classifier construction phase (see Figure 3.1).
CHAPTER 4: FEATURE SUBSET SELECTION

Feature subset selection (FSS) is the process that selects the best subset from the original feature set according to some criteria, i.e., given a feature set that describes a target (text category) using \( n \) features, the goal of the FSS process is to find an optimal feature subset of size \( m \) features (where \( m \ll n \)) while retaining a suitably high accuracy in representing the original feature set.

FSS is one of the important research problems in data mining (Dash et al., 2002), pattern recognition (Mitra, Murthy & Pal, 2002), bioinformatics (Saeys, Inza & Larrañaga, 2007) and machine learning (Kohavi & John, 1997). It has been widely applied to text classification tasks (Sebastiani, 2002; Leopold & Kindermann, 2002; Nigam et al., 2000; Mladenic, 1998; Yang & Pedersen, 1997; Lewis, 1992a).

4.1 Why Feature Subset Selection

When dealing with domains that suffer from the problems created by high dimensions, learning algorithms have not been very effective. As a result, the machine learning community has developed Dimensionality Reduction (DR) techniques. These DR techniques can be divided into:

1. Transformation based techniques (Feature Extraction).

2. Selection based techniques (Feature Subset Selection).

Feature Extraction (FE) methods reduce the dimensionality of the data by some linear algebra transformations such as Principal Component Analysis (Bishop, 2006), Semantic Indexing (Wiener, Pedersen & Weigend, 1995), Independent Component Analysis (Kolenda, Hansen & Sigurdsson, 2000), and word clustering (Cohen & Singer, 1999).
On the other hand, Feature Subset Selection techniques aim to remove non-informative features according to some statistics such as filtering irrelevant features (Yang & Pedersen, 1997) or wrapping the features around the classifier used (Kohavi & John, 1997).

In TC tasks, because the number of features is huge, and because some of these features are redundant and do not reveal significant input-output (document-category) characteristics, it is important to consider how to select the best features. This feature subset selection process makes the learning task more efficient and more accurate (Dash & Liu, 1997).

The FSS process provides several advantages for a text classification system:

1. FSS improves the performance of TC tasks in terms of learning speed, effectiveness, and the comprehensibility of the final classification model. (Building the classifier is usually simpler, faster, and more cost-effective), i.e. it produces better prediction performance in the case of supervised learning (classification) and better cluster detection in the case of unsupervised learning (clustering).

2. FSS reduces the number of data dimensions and removes irrelevant, redundant, or noisy data.

3. FSS provides deeper insight into the underlying processes that generated the data.

However, the advantages of FSS techniques come at a certain price, as the search for a subset of optimal or more relevant features introduces an additional layer of complexity in the modeling task. In TC tasks, additional complexity is added in the preprocessing
phase. On the other hand, FSS may decrease the classifier’s accuracy and make it more susceptible to the overfitting\(^\text{20}\) problem (Liu & Yu, 2005).

### 4.2 Feature Subset Selection: Main Steps

The FSS process consists of the following basic steps (Kohavi & John, 1997; Dash & Liu, 1997; Liu & Yu, 2005; Liu, 2005) (see Figure 4.1):

1. **Feature Generation**: in this step, some search process generates candidate feature subsets.

2. **Feature Evaluation**: using a certain evaluation criterion, the candidate feature subsets are evaluated to measure the goodness of the produced features. When compared

---

\(^{20}\) Overfitting occurs when the training data is over trained; as a result, the model is not able to correctly classify unseen examples.
with previous subsets of features, if a new subset of features is better, it replaces the previous best subset.

3. **Stopping Criterion:** a specific stopping criterion determines when to stop searching the feature space.

4. **Feature Validation:** using a validation procedure, a decision is made whether a feature subset is valid or not. (In fact, this step is not a part of FSS process itself, but in practice, we need to verify the validity of the FSS outcome).

The following sections are devoted to feature generation, feature evaluation, stopping criterion, and feature validation.

### 4.2.1 Feature Generation

In TC tasks, feature generation is a heuristic search problem that searches for candidate feature subsets; the characteristics of this search are determined by the search starting point and the search strategy used.

The search starting point can be initialized by one of the following methods:

1. Search may start with an empty feature subset and successively add features (i.e., forward).
2. Search may start with a full feature subset (i.e., all the features) and successively remove features (i.e., backward).
3. Search may randomly select a candidate feature subset.

On the other hand, search strategies can be classified as complete, sequential and random (Liu & Yu 2005) (see Figure 4.2):
Figure 4.2: Search Strategies
1. Exhaustive strategy (complete): theoretically, a complete generator can produce all the subsets of an inputted feature set. As a result, a feature subset selection algorithm with a complete generator can find out the optimal feature subset but with the cost of high time complexity. For a feature set with $N$ features, there exist $2^N$ candidate feature subsets. The following are examples of complete search strategies:

   a) Branch and bound search strategy (Narendra & Fukunaga 1977): In fact, it does not always mean that exhaustive search is necessary in real complete search problems (Schlimmer, 1993), for example, the branch and bound search strategy may greatly prune the search space.

   b) Beam search (Doak, 1992), a variation of best first search (BFS\textsuperscript{21}), is another example of a complete search strategy that uses a bounded queue to limit the search space.

2. Sequential (heuristic) strategy: the following are three well known examples of sequential search strategies:

   a) Forward selection (Liu & Motoda, 1998): a candidate feature subset is initialized to an empty feature subset, and then a best feature is successively added into it, until a specific stopping criterion is satisfied.

   b) Backward elimination (Liu & Motoda, 1998): a candidate feature subset is initialized full, and then the process successively eliminates the currently worst feature from it, until a specific stopping criterion is satisfied (e.g., there are no features left or a predefined performance effectiveness measure is degraded).

   c) Bi-directional selection (Liu & Motoda, 1998; Doak, 1992): a feature subset is initialized empty, full or randomly, and then a best current feature is added

\textsuperscript{21} Best first search is a simple search strategy that always expands the most promising path.
into it or a worst current feature is eliminated from it, until a specific stopping criterion is met.

3. Random strategy: a sort of search algorithm based on random choices such as the following:

   a) Random-start hill-climbing (Doak, 1992): it conducts a series of hill climbing (HC) (see section 4.7.1) searches from random generated initial nodes. This algorithm cannot ensure an optimal solution. However, a reasonable solution can usually be achieved.

   b) Simulated annealing (SA) (Henderson, Jacobson & Johnson, 2003): SA is a stochastic optimization method that escapes local maxima solutions (the algorithm stops even though the solution is far from optimality) by allowing some downhill moves (see section 4.7.2).

   c) Genetic algorithm (GA) (Yang & Honavar, 1998; Zhang, Jack & Nandi, 2005): GA is an optimization algorithm, which is based on Darwinian principle of natural selection. GA is a robust method to find optimal or near optimal solutions (Goldberg, 1989) (see section 4.7.3).

   d) Ant colony optimization (ACO) (Sivagaminathan & Ramakrishnan, 2007; Subbotin & Oleynik, 2007; Al-Ani, 2005): ACO is an optimization algorithm, which is derived from the study of real ant colonies and it is one of the promising approaches to better feature selection (see Chapter 6).

   e) Other optimization algorithms such as Particle Swarm Optimization (PSO) (Wang et al., 2007), Memetic Algorithm (MA) (Zhu, Ong & Dash, 2007) and Shuffled Frog Leaping (SFL) (Eusuff & Lansey, 2003) are also applicable to optimize search processes.
Table 4.1 shows a comparison between the three mentioned search strategies. It is clear that complete search strategies are optimal but expensive; on the other hand, sequential search strategies are not optimal but simple and fast. Random search strategies are designed to escape local maxima (minima), but their parameters are hard to tune. Therefore, depending on the application, a tradeoff between optimality and complexity shall be considered to select the right search strategy.

The main disadvantage of Forward Selection and Backward Elimination methods is that the selected candidate features that were once selected (or removed) cannot be later removed (or re-selected). However, “Floating Search” algorithm has been proposed to overcome this problem (Pudil, Novovicova & Kittler, 1994).

Table 4.1: Comparison of search strategies

<table>
<thead>
<tr>
<th>Search Strategy</th>
<th>Degree of Optimality</th>
<th>Complexity</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>Optimal</td>
<td>Exponential</td>
<td>Accurate</td>
<td>Highly complex</td>
</tr>
<tr>
<td>Sequential</td>
<td>Good</td>
<td>Quadratic</td>
<td>Simple &amp; fast</td>
<td>No backtracking</td>
</tr>
<tr>
<td>Random</td>
<td>Good</td>
<td>Low</td>
<td>Designed for optimality</td>
<td>Hard to tune parameters</td>
</tr>
</tbody>
</table>

Recently, famous FSS solutions are based on optimization algorithms such as GA (Siedlecki & Sklansky, 1989), SA (Debuse & Smith, 1997), ACO (Al-Ani, 2005), etc.

4.2.2 Feature Evaluation

The main objective function of the “feature evaluation step” is to evaluate the selected candidate feature subsets and to return a measure of their “goodness”. This goodness (a feedback) is used by the search strategy to select new (hopefully better) feature candidate subsets.
Evaluation functions are classified in two main groups: Filters and Wrappers.

1. In filters, the evaluation function evaluates candidate subsets by their information contents. Various evaluation measures are commonly used such as: Chi-square (Caropreso, Matwin & Sebastiani, 2001), Information gain (Caropreso, Matwin & Sebastiani, 2001) and Mutual information (Ruiz & Srinivasan, 1999).

2. On the other hand, in wrappers, the evaluation function evaluates candidate subsets by their predictive accuracy for a predetermined classifier (in our TC task, the SVM text classifier evaluates the selected candidate subsets by their $F_1$ measure values). The classifier’s accuracy can be used to evaluate the goodness of the selected candidate features on a validation set. Generally, evaluation computes the classifier’s accuracy through an $n$-fold cross validation process\(^{22}\).

4.2.3 Stopping Criteria

A stopping criterion determines when to stop the FSS process. The following are some examples of stopping criteria:

1. A specified feature selection quality:
   a) If the classifier’s effectiveness achieved a predefined threshold value (for example some accuracy, precision, recall value), the FSS process stops.
   b) If the classifier’s “mean square error” goes below a predefined threshold, the FSS process stops.

2. FSS convergence: the new selected features do not produce a significant improvement in feature quality.

\(^{22}\) Cross-validation breaks a dataset into, say, 10 subsets, and on each subset, tests the performance of a classifier trained on the remaining 90% of the dataset. In this way, one can estimate how well each of several learning algorithms perform on the available dataset. The best is then chosen to be trained on all of the data.
3. After it iterates a predefined number of iterations, the FSS process stops.

4. Number of features: a FSS process stops after selecting a predefined number of features.

4.2.4 Feature Validation
The goal of the feature validation step is to judge the pertinence of the FSS process. The most common approach to validate the quality of a selected feature subset is to measure the classifier’s effectiveness; we may compare the effectiveness of the classifier trained on all features (without feature selection) and that trained on the subset of features that is selected by the FSS process. However, the computing cost of the FSS method has to be taken into account.

4.3 Filters Vs. Wrappers
FSS algorithms have received much attention in the machine learning literature (Liu & Yu, 2005; Liu, 2005; Zhao et al., 2002), where two main approaches have been studied – filter (Dash et al., 2002) and wrapper (Kohavi & John, 1997) approaches.

The major disadvantage of wrapper FSS methods (Dash et al., 2002; Blum & Langley, 1997) over filter FSS methods is the former's computational cost. However, wrapper FSS methods are more effective (most often, in term of classifier’s accuracy) than filter FSS methods (Doak, 1992).

Selecting the right FSS method is a key issue for better classification systems. Selecting the right FSS method is influenced by many factors such as the following:

1. Data size: With large amounts of data, we prefer filter FSS methods to wrapper FSS methods and heuristic search strategies over complete search strategies.
2. Data consistency: whether the data is noisy or not; the consistency or inconsistency of the data may determine our choice. Some FSS methods may work well for consistent data but not for noisy data.

Table 4.2 summarizes the advantages and the disadvantages of filter and wrapper FSS methods.

Hybrid FSS methods (Das, 2001; Xing, Jordan & Karp, 2001) are other approaches for FSS that take advantages of filter and wrapper FSS approaches. In addition to an independent measure (such as Chi-square), Hybrid FSS methods use the effectiveness of a classifier \( C \) to improve the selection of an optimal subset of features.

4.4 Feature Subset Selection in Text Classification Tasks

In TC tasks, all the features in the training dataset are independently evaluated by a specific FSS function, a score is assigned to each single feature, and then the features are sorted according to their scores. Lastly, a predefined number of best features is selected to form a feature subset. Scoring of individual features is performed by a machine learning measure (Sebastiani, 2002; Mladenic, 1998; Yang & Pedersen, 1997; Forman, 2003) such as Chi-square statistic, GSS score, NGL coefficient, Odds ratio, Mutual Information, and Information gain.

For TC tasks, it is clear that most of the published FSS algorithms are filter based; this can be justified as follows:

1. The wrapper FSS approaches are expensive.
2. Wrapper FSS methods are impractical for extremely large training applications.
3. The evaluation phase in wrapper FSS methods is dependent on the classifier in use.
4. In TC tasks, the number of features is huge.
Table 4.2: Filter FSS and Wrapper FSS comparison

<table>
<thead>
<tr>
<th>FSS approach</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Wrapper FSS methods</strong></td>
<td>Wrapper FSS methods achieve better classification effectiveness.</td>
<td>For each feature subset, Wrapper FSS methods train a predefined classifier; as a result, they are much slower than filter FSS methods. i.e., wrapper FSS methods are classifier dependent feature selection methods.</td>
</tr>
<tr>
<td></td>
<td>Wrapper methods interact with classifiers.</td>
<td>Wrapper methods are less prone to local optima.</td>
</tr>
<tr>
<td></td>
<td>Because they are using cross validation, Wrapper FSS methods are able to avoid overfitting.</td>
<td>Without cross fitting, wrapper methods are prone to overfitting.</td>
</tr>
<tr>
<td></td>
<td>Wrapper methods model feature dependencies.</td>
<td>Since wrapper FSS methods train a predefined classifier, the selected feature subset is only optimal for that specific classifier.</td>
</tr>
<tr>
<td><strong>Filter FSS methods</strong></td>
<td>Filter FSS methods are faster than wrapper FSS methods.</td>
<td>Filter FSS methods do not perform better than wrapper FSS methods. Filter FSS methods select all the features as an optimal set. This is why it is the user responsibility to decide the number of features.</td>
</tr>
<tr>
<td></td>
<td>While selecting features, they do not interact with a specific classifier, as a result, the features selected by filter FSS methods work well with many classifiers.</td>
<td>Filter methods ignore interaction with the classifier.</td>
</tr>
<tr>
<td></td>
<td>Filter methods have less computational complexity than wrapper methods.</td>
<td>Filter methods are scalable.</td>
</tr>
</tbody>
</table>
4.4.1 Chi-square Statistic

The Chi-square statistic (Schütze, Hull & Pedersen, 1995; Yang & Pedersen, 1997; Galavotti, Sebastiani & Simi, 2000; Caropreso, Matwin & Sebastiani, 2001; Forman, 2003) measures the lack of independence between the text term $t$ and the text category $c$ and can be compared to the $\chi^2$ distribution with one degree of freedom to judge the extremeness.

The mathematical definition of Chi-square is as follows:

$$\chi^2(t, c) = \frac{N[P(t, c)P(\overline{t}, \overline{c}) - P(t, \overline{c})P(\overline{t}, c)]^2}{P(t)P(\overline{t})P(c)P(\overline{c})}$$

where:

- $t$ denotes a term and $c$ denotes a category.

All probabilities are interpreted on events in the training document space. For example $P(t, \overline{c})$ denotes the probability that a term $t$ occurs in a document $x$ that does not belong to class $c$. $P(\overline{c})$ is estimated as the number of documents that do not belong to class $c$ divided by the total number of training documents.

Functions are specified “locally” to a specific category $c$. To globally assess $t$ values, either the sum $f_{\text{sum}}(t) = \sum_{i=1}^{n} f(t, c)$, the weighted sum $f_{\text{wsum}}(t) = \sum_{i=1}^{n} P(c)f(t, c)$, or the maximum of their category-specific values $f(t, c)$ are computed ($f_{\text{wmax}}(t) = \max_{i=1}^{n} f(t, c)$).

Using the two-way contingency table (Table 4.3) of a term $t$ and a category $c$, the term-goodness measure (Chi-square score) is defined as follows:
\[
\chi^2(t, c) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)}
\]

Table 4.3: Two-way Contingency Table.

<table>
<thead>
<tr>
<th></th>
<th>A = #(t,c)</th>
<th>C = #(¬t,c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B =</td>
<td>#(t,¬c)</td>
<td>D = #(¬t, ¬c)</td>
</tr>
<tr>
<td>N =</td>
<td>A + B + C + D</td>
<td></td>
</tr>
</tbody>
</table>

where:

- \(A\) is the number of times \(t\) and \(c\) co-occur.
- \(B\) is the number of times \(t\) occurs without \(c\).
- \(C\) is the number of times \(c\) occurs without \(t\).
- \(D\) is the number of times neither \(c\) nor \(t\) occurs.
- \(N\) is the total number of documents.

This \(\chi^2\) statistic has a natural value of zero if \(t\) and \(c\) are independent.

4.4.2 Information Gain

Information gain (IG\(^{23}\)) (Yang & Pedersen, 1997; Caropreso, Matwin & Sebastiani, 2001; Forman, 2003) measures the number of bits of information obtained for category prediction by knowing the presence or absence of a term \(t\) in a document.

\(^{23}\) IG is also known as Expected Mutual Information (Lewis, 1992b).
For each category $c_i$, where $i \in [1, M]$ and $M$ is the number of categories in a TC corpus.

The IG for a term $t$ is defined as follows:

$$IG(t) = -\sum_{i=1}^{M} P(c_i) \cdot \log P(c_i)$$
$$+ P(t) \sum_{i=1}^{M} P(c_i \mid t) \cdot \log P(c_i \mid t)$$
$$+ P(\bar{t}) \sum_{i=1}^{M} P(c_i \mid \bar{t}) \cdot \log P(c_i \mid \bar{t})$$

where:

$P(c_i)$ is the probability that a document $d$ belongs to class $c_i$.

$P(c_i \mid t)$ is the probability of a class $c_i$ given that a document $d$ contains a term $t$.

$P(c_i \mid \bar{t})$ is the probability of class $c_i$ given that a document $d$ does not contain a term $t$.

4.4.3 Mutual Information

The Mutual Information (MI) score (Yang & Pedersen, 1997; Dumais et al. 1998; Galavotti, Sebastiani & Simi, 2000) for a term $t$ and a category $c$ is defined as follows:

$$MI(t, c) = \log \frac{P(t)}{P(t) \times P(c)}$$

Using the Two-way contingency table (Table 4.3), MI is estimated as:

$$MI(t, c) \approx \log \frac{A \times N}{(A + C) \times (A + B)}$$

If the MI for a term $t$ is zero then a term $t$ and a category $c$ are independent.
4.4.4 GSS Score

The GSS (Galavotti, Sebastiani & Simi, 2000) score for a term $t$ and a category $c$ is defined as follows:

$$GSS(t, c) = P(t, c) \times P(\bar{t}, \bar{c}) - P(t, \bar{c}) \times P(\bar{t}, c)$$

Using the Two-way contingency table (Table 4.3), GSS can be estimated as:

$$GSS(t, c) = A \times D - C \times B$$

We may view the GSS as a simplified version of Chi-square, but it only considers positive features.

4.4.5 NGL Coefficient

The NGL (Ng, Goh & Low, 1997; Ruiz & Srinivasan, 1999) score for a term $t$ and a category $c$ is defined as follows:

$$NGL(t, c) = \sqrt{N}.[P(t, c)P(\bar{t}, \bar{c}) - P(t, \bar{c})P(\bar{t}, c)] \div \sqrt{P(t)P(\bar{t})P(c)P(\bar{c})}$$

Using the Two-way contingency table (Table 4.3), NGL can be estimated as:

$$NGL(t, c) = \frac{\sqrt{N} \times (AD - CB)}{\sqrt{(A + C) \times (B + D) \times (A + B) \times (C + D)}}$$

NGL can be viewed as a variant of Chi-square statistic, where:

$$[NGL(t, c)]^2 = \chi^2$$.
4.4.6 Odds Ratio

The Odds Ratio (OR) (Mladenic, 1998; Mladenic & Grobelnik, 1999; Galavotti, Sebastiani & Simi, 2000; Caropreso, Matwin & Sebastiani, 2001; Forman, 2003) measures the odds of a term $t$ occurring in documents in a category $c$ divided by odds of a term $t$ not occurring in documents in category $c$.

OR score for a term $t$ and a category $c$ is defined as follows:

$$ OR(t,c) = \frac{P(t | c).(1 - P(t | \overline{c}))}{(1 - P(t | c)).P(t | \overline{c})} $$

4.4.7 Other Feature Subset Selection Methods

Many other FSS methods have been studied for TC tasks, such as:

1. OR-square (ORS) and GSS-square (GSSS): ORS and GSSS (Zheng, Srihari & Srihari, 2003) are defined respectively as follows:

   $$ ORS(t,c) = OR^2(t,c) $$

   $$ GSSS(t,c) = GSS^2(t,c) $$

2. The Document Frequency (DF) (Yang & Pedersen, 1997) for a term $t$ is the number of documents in which term $t$ occurs. First, the DF for each term $t$ in the training set is calculated, second, the terms whose DF is less than some threshold are removed (e.g.: all terms with DF less than 3 are removed).

   The mathematical function for DF is:

   $$ \#(t,c) = P(t | c) $$
3. Relevancy Score (RS) (Sebastiani, 2002) is another FSS method that is defined as:

\[ RS(t, c) = \log \frac{P(t \mid c) + \kappa}{P(t \mid \bar{c}) + \kappa} \]

where \( \kappa \) is a constant.

4. Log probability ratio (Mladenic & Grobelnik, 1999) is defined as:

\[ PR(t) = \log \frac{P(t \mid pos)}{P(t \mid neg)} \]

where:

- \( P(t \mid pos) \) is the probability of term \( t \) occurring given class value “positive”.
- \( P(t \mid neg) \) is the probability of term \( t \) occurring given class value “negative”.

5. The Odds Numerator FSS method is a variant of the OR FSS method, it is the numerator of OR.

6. The Bi-Normal Separation (BNS) FSS score (Forman, 2003) for a term \( t \) is defined as:

\[ BNS(t) = F^{-1}(P(t \mid pos)) - F^{-1}(P(t \mid neg)) \]

where \( F \) is the standard normal distribution.

7. The Term Strength (TS) FSS method was originally proposed by Wilbur & Sirotkin (Wilbur & Sirotkin, 1992) and was then applied to text classification (Yang, 1995; Yang & Wilbur, 1996). TS measures term importance (goodness) based on how commonly a term is likely to appear in closely related articles. It
uses the cosine distance value to derive the similar pairs of documents from the training set. Let $d_1$ and $d_2$ be an arbitrary pair of distinct but related documents and $t$ be a term, the term strength of the term $t$ is defined as follows (Yang & Pedersen, 1997):

$$TS(t) = P(t \in d_2 \mid t \in d_1)$$

4.5 Feature Subset Selection Examples

Table 4.4 shows the best 35 Arabic text features for “Education” category, these features are selected by four different FSS methods. The listed features are sorted from best to worst according to their corresponding FSS score. It is clear that different FSS methods select different feature subsets (this may justify the usage of ACO algorithm to optimize the FSS process).

4.6 Feature Subset Selection Discussion

There are many valuable studies that investigate FSS methods for TC tasks; these studies have used different classifiers and many TC corpora:

1. Yang and Pedersen (Yang & Pedersen, 1997) have made a valuable study of FSS methods for TC tasks. They concluded that the IG and Chi-square FSS methods performed most effectively with the $k$NN classifier. On the other hand, they concluded that MI and TS performed terribly.

2. Forman (Forman, 2003) has presented another valuable study of twelve FSS metrics with SVM classifier for TC tasks. And he has shown that IG performed best in term of precision. However, he suggested the Bi-Normal Separation FSS method to improve $F_1$ measure.
Table 4.4: Best 35 Arabic text features for “Education” category selected by different FSS methods.

<table>
<thead>
<tr>
<th>Chi-square</th>
<th>NGL</th>
<th>GSS</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>التربوي</td>
<td>التعليم</td>
<td>تربية</td>
<td>المدرسي</td>
</tr>
<tr>
<td>التربوي</td>
<td>تربية</td>
<td>التعليم</td>
<td>الدراسي</td>
</tr>
<tr>
<td>الثاني</td>
<td>التربوي</td>
<td>الطفل</td>
<td>توجيه</td>
</tr>
<tr>
<td>جميع</td>
<td>المدارس</td>
<td>الابا</td>
<td>اتباعهم</td>
</tr>
<tr>
<td>المدرسيه</td>
<td>الطفل</td>
<td>المدرسة</td>
<td>التوضيح</td>
</tr>
<tr>
<td>الدراسي</td>
<td>الابا</td>
<td>المنزل</td>
<td>الجهاد</td>
</tr>
<tr>
<td>الابا</td>
<td>المدرسة</td>
<td>الطلاب</td>
<td>الشهادة</td>
</tr>
<tr>
<td>التعليم</td>
<td>المدرسيه</td>
<td>الدراسي</td>
<td>التحصيل</td>
</tr>
<tr>
<td>التعليم</td>
<td>الطلاب</td>
<td>المدرسي</td>
<td>تشتيت</td>
</tr>
<tr>
<td>المدرسي</td>
<td>الابا</td>
<td>الابا</td>
<td>يقظي</td>
</tr>
<tr>
<td>الابا</td>
<td>المدرسة</td>
<td>الابا</td>
<td>يهدة</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>الرادي</td>
<td>يستمع</td>
</tr>
<tr>
<td>الابا</td>
<td>الطلاب</td>
<td>الدراسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الابا</td>
<td>الطلاب</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الابا</td>
<td>الطلاب</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
<tr>
<td>الطلاب</td>
<td>الابا</td>
<td>المدرسي</td>
<td>ينفعل</td>
</tr>
</tbody>
</table>
3. Rogati and Yang (Rogati & Yang, 2002) have studied four FSS methods (DF, IG, Chi-square, MI and IG224) with NB, Rocchio, kNN and SVM classifiers. Their results suggested filter based FSS methods, which include the Chi-square statistic, combined with DF or IG.

4. In their study, Eyheramendy and Madigam (Eyheramendy & Madigam, 2005) concluded that FSS methods act differently on different datasets and classification algorithms. In addition to their proposed RIP25 FSS method, they studied IG, BNS, Chi-square, OR and TS FSS methods. Their results shown that IG, Chi-square, BNS and RIP performed best.

From the above studies, we conclude the following:

1. There is no indication that there is a superior FSS for all datasets and for all classifiers.

2. It is hard to suggest a FSS method that is always most effective.

3. However, Chi-square is always among the best FSS methods, regardless of the used classifier.

4. Different FSS methods select different feature subsets from the same original feature set.

5. This work is based on our belief that the Chi-square FSS method can be optimized to better select features (see feature subset selection optimization algorithms section 4.7).

---

24 IG2: is a variation of IG, i.e. a binary version of IG FSS method.

25 RIP is a FSS method that evaluates the Posterior Inclusion Probability of a given feature over all possible models.
4.7 Feature Subset Selection Optimization Algorithms

To improve FSS methods, many machine learning methodologies such as neural networks (Castellano & Fanelli, 2000; Ye & Liu, 2002), the neuro-fuzzy scheme (Chakraborty & Pal, 2004; Li, Mukaidono & Turksen, 2002), fuzzy logic (Francesco, 2003; Zhang, Liu & Fan, 2006), string kernels (Lodhi et al., 2002), etc., have been utilized for FSS processes.

On the other hand, some improvements of the classical FSS methods have been reported (Kwak & Choi, 2002; Fleuret, 2004; Bakus & Kamel, 2006).

Theoretically, FSS has been shown to be an NP-hard problem (Blum & Rivest, 1992), as a result, automatic feature space construction and feature subset selection from a large dataset have become active research areas. On the other hand, it is obvious that SA, GA, ACO, PSO, etc., are well-known optimization algorithms that have proven their robustness in optimization problems. Moreover, it is known that these algorithms offer powerful and domain independent search capabilities that can be used in many learning tasks (such as the FSS process, information retrieval, optimization problems, etc.). For these reasons, the machine learning community has been motivated to study the FSS process as a search problem (optimization problem).

As a result, some researchers (Pudil & Somol, 2005) have further been motivated to categorize FSS methods into:

1. Optimal FSS methods: the FSS methods that are based on complete search strategies such as the branch and bound search algorithm. These methods search the whole feature space to find an optimal subset of features.

2. Suboptimal FSS methods: the FSS processes that are based on sequential and random search strategies such as sequential forward selection, sequential backward selection, genetic algorithms, etc.
In fact, we have to consider a tradeoff between optimality of the selected subset of features and the computational efficiency. This tradeoff between an accurate estimate and an extensive exploration of the feature search space is referred to as the *exploration* versus *exploitation* problem (Dorigo & Stützle, 2004). For TC tasks, because the number of features is huge, evaluating all features (in complete search strategies) is computationally non-feasible necessitating the need for other random based search methods (HC, GA, ACO, etc.).

The following sections present the use of Hill Climbing (HC), SA, and GA techniques as feature subset selection algorithms.

4.7.1 Hill Climbing

The main idea of the Hill Climbing (HC) search algorithm (Russell & Norvig, 2003; Wang, Youssef & Elhakeem, 2006) is to select the best solution from the neighborhood of a given (current) solution. The new selected solution must be better than current solution. HC terminates when the neighborhood does not contain any better solution.

The HC based FSS process can be summarized in the following steps:

1. Start from a randomly selected solution (subset of features), this solution can be represented by a vector that contains the initial candidate features, and the length of this vector indicates the number of selected features for a given solution.

2. Evaluate the goodness of the subset of features that has been selected in the initial step. Evaluation can be conducted by some evaluation function, such as the effectiveness of a specific classifier (most often, accuracy).

3. Generate a random neighboring subset of features, and then evaluate the new selected solution’s accuracy.
4. If the accuracy of the new selected solution is better than the accuracy of the current solution, then we replace the current solution (best solution) with the new selected solution.

5. Iterate until a predefined stopping criterion is met.

HC has one main drawback; it gets stuck in a local maximum trap, since the HC search stops when reaching the first local optimum solution. However, a modified version of HC involves the usage of multiple initial solutions, hoping that the next local maximum found is better than the previous one.

4.7.2 Simulated Annealing

Simulated Annealing (SA) algorithm (Debuse & Smith, 1997; Henderson, Jacobson & Johnson, 2003; Wang, Youssef & Elhakeem, 2006) is a local search algorithm inspired by the process of physical annealing with solids. SA selects a sequence of solutions from the neighborhood of a given solution.

The new selected solution does not necessarily have to be better than the current solution. If the selected solution is better than the current solution, SA acts similar to HC search, but if the next solution is not better than the current solution, the new solution replaces the current solution with a probability that depends on the difference in goodness between the new and the current solutions. This enables the SA algorithm to escape local maxima. The SA based FSS can be summarized as follows (Wang, Youssef & Elhakeem, 2006):

1. Step 1: Initialize SA parameters such as temperature ($T$) and Boltzmann constant ($K$), these parameters are used to control the probability of accepting bad moves (accept a newly selected bad solution as a current solution hoping that some new better solution will be reached later).
2. Step 2: Randomly select an initial solution. This solution is represented as a vector. The vector length represents the number of features for each solution. This initial solution is set as the best solution so far and as the current solution.

3. Step 3: Evaluate the goodness of the selected features.

4. Step 4: Randomly select a new solution from neighborhood of the current solution and evaluate its goodness.

5. Step 5: Compare the goodness of the two solutions (current and new selected solutions). If the goodness of the new solution is better than the goodness of the current solution, the new solution is set as the best solution so far. Otherwise, a random number \( R \) is generated and compared with the following selection probability \( P \):

\[
P = e^{\frac{\Delta E_i}{T_i \times K}}
\]

where \( \Delta E_i \) is the goodness difference between the current and the new selected solutions.

6. Step 6: If \( R < P \) then the new solution is accepted as current solution (but not as best so far).

7. Step 7: If a predefined stopping criterion is not met, then randomly select a new temperature parameter and go to Step 4.

4.7.3 Genetic Algorithms

Holland (Holland, 1975) introduced Genetic Algorithms in the 1970’s as an optimization algorithm based on both stochastic and probabilistic measures; GA (Goldberg, 1989) inspects the solution space (feature space) for an optimal solution (optimal subset of features) (Eiben & Schoenauer, 2002). The basic steps for genetic algorithms are
illustrated in Figure 4.3. GA starts with a randomly selected “population” of possible solutions for the target problem and lets them “evolve” over multiple generations to find better solutions (Thede, 2004). The GA algorithm is mainly based on the “survival of the fittest” principle (organisms that best “fit” their environment have the best chance of survival (Thomas, 2004)).

While executing GA, new population individuals are “born” while others “die”. In GA, a stochastic process, Crossover, takes two (or more) parent nodes that may generate offspring nodes (possible better solutions) by exchanging “chromosomes” between parent nodes. As a result, new individuals (better solutions) may be created in the next population. Mutation is another stochastic transformation of individuals that may modify their genotypes. Individuals are selected for “crossover” based on their fitness values (goodness of selected subset of features). Better goodness features are more likely to reproduce (survive).

Figure 4.3: GA process

GA as a FSS method can be summarized as follows (Oh, Lee & Moon, 2004; Raymer et al., 2000; Morariu, Vintan & Tresp, 2006):
1. Step 1: A random population is generated, this population consists of a predefined number of random feature solutions (subsets) (chromosomes), each individual solution is represented as a vector, and the vector length represents the number of selected features for each solution.

2. Step 2: The fitness of each solution is computed, fitness uses a specific evaluation function to measure the goodness of each solution (feature subset).

3. Step 3: Repeat until a curtain stopping criterion is met: a stopping criterion terminates the GA search algorithm when some fitness value is achieved, after iterating a predefined number of iterations, etc.

   The solution with best fitness is stored.

   Then a new population is generated:

   a) Select two parents from population
   b) Crossover the parents.
   c) Mutate new children (“offsprings”).
   d) Place the new generated “offsprings” to the population and randomly select a new population for a further run of the GA algorithm.

4. Step 4: Return the best solution.

4.7.4 Optimization Algorithms Discussion

A previous study by Wang and his colleagues (Wang, Youssef & Elhakeem, 2006) has shown that SA performs better than HC. However, a random-start hill-climbing algorithm performs better than HC, but as mentioned before it cannot insure an optimal solution (however, a reasonable solution usually can be achieved).
It has been found (Angeline, 1998) that PSO discovers a reasonable quality solution much faster than other evolutionary algorithms. On the other hand, when the number of generations is increased, it does not improve the solution quality.

For discrete optimization problems, a comparison study (Elbeltagi, Hegazy & Grierson, 2005) (between PSO, GA, MA, SFL, and ACO) found that PSO is generally performs best in terms of success rate and solution quality. However, for the same discrete optimization problems, ACO performs better than PSO in term of processing time.

Because of the huge number of features in TC tasks, the author of this thesis preferred to trade some solution quality for computational complexity and decided to use Ant Colony Optimization Algorithm to implement an Ant Colony Optimization based Feature Subset Selection Algorithm for Text Classification Tasks.
CHAPTER 5: INTRODUCTION TO ARABIC LANGUAGE

Arabic is the official language of over twenty Arab countries which stretch from Morocco to Iraq and it is the religious language of all Muslims of more than one billion Muslims spread all over the world. Arabic is the language of the Quran (the sacred book of Islam) as stated in its Surat (verse) 42:7 “... thus we have inspired unto you an Arabic Quran”.

5.1 Description of Arabic Language Words

Arabic language is a Semitic language and most of its words are built up from roots by following certain fixed patterns and adding infixes, prefixes and suffixes. For example, ‘مدرس’ ‘‘teacher’’ is formed from the root ‘درس’ ‘‘study’’. Prefixes and suffixes are added to Arabic words to form new meanings, for example adding ‘ون’ to ‘مدرس’ ‘ون’ forms a new word ‘مدرسون’ ‘‘teachers’’.

Arabic is an old language, and what is now known as Classical Arabic was standardized around fourteen centuries ago. The modern form of Arabic (Khoja, 2001) is called Modern Standard Arabic (MSA) and it is the form used in all Arabic-speaking countries in publications and media.

Genders in Arabic are masculine, feminine, and neuter. Arabic also contains three persons, one describes the speaker, one describes the person being addressed and one describes the person that is not present. Arabic deals with singular, dual and plural.

Arabic grammarians describe Arabic as being derived from noun, verb, and particle (Khoja, 2001; Khoja, Garside & Knowles, 2001):

1. A noun is a name or a word that describes a person, a thing, or an idea.
2. Similar to English verbs, verbs in Arabic are classified into Perfect, Imperfect, and Imperative.

3. Arabic Particles include Prepositions, Adverbs, Conjunctions, Interrogative Particles, Exceptions, and Interjections.

Arabic words are classified into the following five categories (Khoja, Garside & Knowles, 2001) (see Figure 5.1):

1. Noun: nouns are sub-categorized into:
   a) Common nouns such as ‘الحَلَف’ ‘كتاب’ “book” in ‘الحَلَف’ ‘كتاب’ “the boy took a book”.
   b) Proper nouns such as ‘ئيذ’ ‘شيرين’ “Shereen”.
   c) Pronouns: pronouns are further sub-categorized into:
      i. Personal such as ‘هو’ ‘ه’ “he or him”.
      ii. Relative: relative pronouns are further sub-categorized into:
          1. Specific such as ‘الذين’ ‘who’.
          2. Common such as ‘من’ ‘who’.
      iii. Demonstrative such as ‘هذا’ ‘this’.
   d) Numeral nouns: numeral nouns are further sub-categorized into:
      i. Cardinal such as ‘اربعة’ ‘four’.
      ii. Ordinal such as ‘رابع’ ‘fourth’.
      iii. Numerical adjective such as ‘رباعي’ ‘of four’.
   e) Adjectives are nouns that describe the object’s aspects such as ‘الرَبَّانِي’ ‘الولد الصغير’ “The small boy”.

55
Figure 5.1: Arabic words Categorization
2. Verb: verbs are categorized into:
   a) Perfect such as ‘كسرت’, “I broke”.
   b) Imperfect such as ‘كسر’, “I break”.
   c) Imperative such as ‘كسر’, “Break”.

3. Particle: particles are categorized into:
   a) Prepositions such as ‘في’ “in”.
   b) Adverbial such as ‘شو’ “shall”.
   c) Conjunctions such as ‘و’ “and”.
   d) Interjections such as ‘يا’ “you”.
   e) Exceptions such as ‘سوى’ “Except”.
   f) Negatives such as ‘لم’ “Not”.
   g) Answers such as ‘اُهْل’ “yes”.
   h) Explanations such as ‘إي’ “that is”.
   i) Subordinates such as ‘لو’ “if”.

4. Residual: the residual category is further sub-categorized into:
   a) Foreign words such as ‘روجر’, “Roger”.
   b) Mathematical formulae such as ‘+’ “+”.
   c) Numbers such as ‘3’ “3”.
5. Punctuation: the punctuation category contains punctuation symbols such as (?,?,!,

5.2 Challenges of Arabic Language in TC tasks

The difficulty of Arabic TC comes from several sources, the following are some of them (Khoja, 2001; Hmeidi, Hawashin & El-Qawasmeh, 2008):

1. Arabic language (Khoja, 2001) differs syntactically, morphologically, and semantically from other Indo-European languages.

2. Compared to English, Arabic language (Yahya, 1989; Goweder & De Roeck, 2001) is more sparsely, which means that English words are repeated more often than Arabic words for the same text length.

   Sparseness yields less weight for Arabic terms (features) than English features. The difference of weight among Arabic word features is less and this makes it more difficult to differentiate between different Arabic words (this may negatively affect Arabic text classifier’s effectiveness).

3. In written Arabic, most letters take many forms of writing. Moreover, there is a punctuation associated with some letters that may change the meaning of two identical words.

   In written Arabic, we may have two identical words with different punctuation; meanwhile, the meaning of the two words is different.

   Each row in Table 5.1 contains the same Arabic word, but the Arabic TC system may handle them differently unless a special care is taken in the preprocessing phase in the TC system (see TC architecture in Chapter 3).

4. The omission of diacritics (vowels) in written Arabic "altashkiil".
5. The punctuation associated with each letter may conflate the meaning of the Arabic word (see Table 5.1).

<table>
<thead>
<tr>
<th>Form 1</th>
<th>Form 2</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>葳نتهاء</td>
<td>葳تنا</td>
<td>Different forms for the first letter</td>
</tr>
<tr>
<td>葳تنا</td>
<td>葳تنا</td>
<td>Different forms for the last letter</td>
</tr>
<tr>
<td>葳تنا</td>
<td>葳تنا</td>
<td>Difference is the conjunction letter</td>
</tr>
<tr>
<td>葳تنا</td>
<td>葳تنا</td>
<td>Different punctuation on the last letter</td>
</tr>
</tbody>
</table>

6. Comparing to English roots, Arabic roots are more complex\(^{26}\). In fact, the same Arabic root, depending on the context, may be derived from multiple Arabic words. On the other hand, the same Arabic word may be derived from several different roots.

Table 5.2 shows some of the Arabic words that maybe generated from the same root ‘كتب’‘كتب’ ‘كتب’‘كتب’‘كتب’. Meanwhile, Table 5.3 shows some possible roots from which the same word ‘إيمان’‘إيمان’‘إيمان’‘إيمان’‘إيمان’ ‘AymAn’ can be derived from.

<table>
<thead>
<tr>
<th>Arabic word</th>
<th>English Meaning</th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>كتب</td>
<td>He wrote</td>
<td>ktb</td>
</tr>
<tr>
<td>كتاب</td>
<td>Book</td>
<td>ktAb</td>
</tr>
<tr>
<td>كاتب</td>
<td>Thier book</td>
<td>ktAbhm</td>
</tr>
<tr>
<td>كتب</td>
<td>He is writting</td>
<td>yktb</td>
</tr>
</tbody>
</table>

\(^{26}\) Table 5.2 and Table 5.3 are selected from (Darwish, 2002).
Table 5.3: Four possible roots for the word “AymAn”.

<table>
<thead>
<tr>
<th>Root</th>
<th>English Meaning</th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Amn”</td>
<td>peace</td>
<td>“Eman”</td>
</tr>
<tr>
<td>“Aym”</td>
<td>two poor people</td>
<td>“Ayyiman”</td>
</tr>
<tr>
<td>“ymn”</td>
<td>convenant</td>
<td>“Ayman”</td>
</tr>
</tbody>
</table>

7. There is no publicly available Arabic TC corpus that can be used to train and test Arabic text classifiers. In current situation, it is unfair to compare different Arabic TC approaches.

5.3 Discussion

Despite of the mentioned challenges (and others), there are some efforts in Arabic natural language processing community that have analyzed Arabic texts to better classifying Arabic documents. However, when compared to other languages, the analyses that have been made in recent years still insignificant. Most Arabic TC literature has investigated a “text classifier construction phase” and little work has investigated the “document pre-processing phase”.

To overcome the mentioned challenges (or at least some of them), a special (a careful) pre-processing of the Arabic TC corpus is required before constructing the text classifier (see TC architecture in Chapter 3). This may justify the special Arabic dataset pre-processing steps that we have applied to transform the Arabic documents into a compact and an applicable form to train the text classifier (see Section 7.2 in Chapter 7). On the other hand, this may justify the usage of ACO algorithm to optimize the FSS process for Arabic TC tasks.
CHAPTER 6: ANT COLONY OPTIMIZATION BASED-FEATURE SUBSET SELECTION ALGORITHM FOR TEXT CLASSIFICATION TASKS

6.1 Ant Colony Optimization Algorithm

In artificial intelligence, there is a plethora of algorithms that are inspired by the collective behavior of social insects. Many of these algorithms are inspired by the behavior of ants while foraging for food. This category of algorithms is known as Ant Colony Optimization, whose first member, called the Ant System, was initially proposed by Marco Dorigo (Dorigo, Maniezzo & Colorni, 1991) (see also (Dorigo, Maniezzo & Colorni, 1996; Dorigo, & Stützle, 2003; Dorigo, & Stützle, 2004)).

The ACO algorithm, which imitates the foraging behavior of real life ants, is a cooperative population-based search algorithm. While traveling, Ants deposit an amount of pheromone (a chemical substance). When other ants find pheromone trails, they decide to follow the trail with more pheromone, and while following a specific trail, their own pheromone reinforces the trail followed. Therefore, the continuous deposit of pheromone on a trail maximizes the probability of selecting that trail by next ants. Moreover, ants use short paths to the food source, return to the nest sooner and therefore, quickly mark their paths twice, before other ants return. As more ants complete shorter paths, pheromone accumulates faster on shorter paths; on the other hand, longer paths are less reinforced. Pheromone evaporation is a process of decreasing the intensities of pheromone trails over time. This process is used to avoid local convergence (old pheromone strong influence is avoided to prevent premature solution stagnation), to explore more search space and to decrease the probability of using longer paths.
Pseudocode for the ACO algorithm is shown below:

Initialize the pheromone trails and other parameters, and calculate heuristic information;

While termination condition not met do
    ConstructAntSolutions
    ApplyLocalSearch (optional)
    UpdatePheromones
endwhile.

After initialization, the ACO algorithm iterates over three main phases: at each iteration, a number of solutions are constructed by the ants; these solutions are then improved through a local search (this step is optional), and finally the pheromone is updated.

ConstructAntSolutions: A set of ants constructs solutions from elements of a finite set of available solution components. A solution construction phase starts from an empty partial solution. At each construction step, each ant extends its partial solution by adding a feasible solution component from a set of neighbor components that can be added to its current partial solution. The choice of a solution component is guided by a stochastic mechanism, which is biased by the pheromone associated with each of the elements of the set of components that can be added to the current partial solution.

ApplyLocalSearch: After constructing solutions, and before updating the pheromone trails, it is common to improve the solutions obtained by the ants through a local search. The ApplyLocalSearch phase, which is highly problem-specific, is optional.

UpdatePheromones: The UpdatePheromones phase aims (i) to increase the pheromone values associated with promising solutions, and (ii) to decrease those that are associated with bad solutions. Usually, UpdatePheromones phase is achieved (i) by decreasing all the pheromone values associated with all solutions through pheromone evaporation, and
(ii) by increasing the pheromone values associated with a chosen set of promising solutions.

6.1.1 Ant Colony Optimization Flavors

The ACO algorithm has many flavors as listed below (Dorigo, Maniezzo & Colorni, 1996; Dorigo, & Stützle, 2004):

1. The Ant System (AS) is the first proposed ACO algorithm, it has been applied to solve travelling salesman problem (TSP).

Three different flavors of AS have been reported:

- Ant-density (Dorigo, Maniezzo & Colorni, 1996).
- Ant-quantity (Dorigo, Maniezzo & Colorni, 1996).
- Ant-cycle (Dorigo, Maniezzo & Colorni, 1996).

In ant-density and ant-quantity, ants only deposit pheromone directly after crossing an arc. In ant-quantity, the amount of pheromone is inversely proportional to the length of the arc crossed, whereas in ant-density a constant amount of pheromone per unit distance is deposited.

In ant-cycle the ant are only allowed to deposit pheromone when they completed a tour.

2. Elitist strategy for ant system (ASe) (Dorigo, Maniezzo & Colorni, 1996), the main idea of this algorithm is to increase the importance of the ant that found the best solution.

---

27 More details are available in (Dorigo, Maniezzo & Colorni, 1996; Dorigo, & Stützle, 2004)
3. Ant Colony System (ACS) (Gambardella & Dorigo, 1996) and Ant-Q (Gambardella & Dorigo, 1995): the main difference between them is the definition of the pheromone trail formula.

4. The Rank-Based Version of the Ant System (Bullnheimer, Hartl & Strauss, 1999): in this algorithm, each ant deposits an amount of pheromone that decreases with its rank.

5. The Max-Min Ant System (Stützle & Hoos, 1996; Stützle & Hoos, 1998): this algorithm aims to exploit more strongly the best solutions found during the search process and to direct the ants’ search towards very high quality solutions. On the other hand, Max-Min Ant System aims to avoid premature stagnation of the ants’ search.

6.1.2 Applications of Ant Colony Optimization

The following are some selected applications of the ACO algorithm (see Chapter 5 in Dorigo, & Stützle, 2004):

1. To solve the Travelling Salesman Problem (TSP) (Dorigo, Maniezzo & Colorni, 1996; Cheng, & Mao, 2007).

2. To solve many optimization problems (Demirel, & Toksar, 2006; Feng, Yu & Zhang, 2007).

3. To optimize mobile networks and communication services (Fournier & Pierre, 2005; Shyu, Lin & Hsiao, 2006).

4. To optimize color reduction (Ghanbarian, Kabir & Charkari, 2007).

5. Data mining (Parpinelli, Lopes & Freitas, 2002).

6. Clustering (Machnik, 2006).
6.2 Ant Colony Algorithms for Feature Subset Selection

In (Al-Ani, 2005), an ACO process was proposed for a specific FSS method; the major applications of this FSS method were only speech segment and texture classification problems. The author used what was called the “Updated Selection Measure (USM)” to select features and “Local Importance” as a local heuristic function.

Similarly, (Jensen & Shen, 2003) proposed a feature selection mechanism that is based on ACO algorithm. Jensen and Shen have presented an entropy-based modification of the original rough set-based approach for FSS problems.

Another work (Schreyer, & Raidl, 2002) used ACO algorithm to label point features, a pre-processing step that reduces the search space.

A recent work (Sivagaminathan, & Ramakrishnan, 2007) has used a hybrid method of ACO algorithm and Neural Networks to select features; their feature selection algorithm was based on a medical diagnosis cost function.

After investigating these ant colony optimization algorithm-based FSS methods, we conclude that the main difference between them is the heuristic function used. On the other hand, one of the main differences between Ant colony optimization algorithms is the way pheromone trails are updated. This motivates the author of this thesis to propose a new pheromone update flavor to be used in a feature subset selection algorithm that is applicable to TC tasks.

6.3 Steps to Solve a FSS Problem by Ant Colony Optimization Algorithm

ACO algorithms have been applied to many different optimization and application problems, as shown by the list of ACO applications presented in section 6.1.2. Based on those applications, it is easy to identify the basic issues that play important roles in the
use of ACO in any combinational problem. These basic issues are the following (Dorigo & Stützle, 2004):

- **Construction Graph:**
  The application problem must be presented as a graph with a set of nodes and edges between nodes.

- **Pheromone Trails Definition:**
  A very important choice when applying ACO is the definition of the meaning of the pheromone trails update; this motivated Blum and Samples (Blum & Sampels, 2002) to study the influence of the way pheromones are defined and updated. As a matter of fact, the definition of pheromone trails is crucial and a poor choice will result in poor ACO performance. Typical methods involve selecting a number of best solutions (ants) and updating the edges they chose.

- **Balancing Exploration and Exploitation:**
  An effective metaheuristic algorithm must achieve an appropriate balance between exploitation of the search experience gathered so far and the exploration of the unvisited search space. This balance is typically achieved through the management of the pheromone trail, where updating the pheromone trails induces a probability distribution over the search space and determines in which part of the search space the constructed solutions with higher probabilities are located. One simple approach to exploiting the ant’s search experience is to define the pheromone trail updating as a function of the solution (ant) quality. However, it is wise to introduce an *elitist strategy* for higher quality solutions, whereby the best solutions (ants) contribute more strongly to the pheromone trail updating. Another simple approach in the balance of exploration and exploitation is tuning $\alpha$ and $\beta$, where $\alpha$ determines the influence of the pheromone trail and $\beta$ determines the effect of heuristic information.
- **Heuristic Information:**

  Using problem related knowledge as heuristic information to direct the ants’ probabilistic solution construction is an important factor to achieve better quality solutions. The main types of heuristic information are static and dynamic.

  Static heuristic information (problem specific knowledge) is computed once at the beginning of the ACO search and then remains unchanged throughout the whole run of the ACO algorithm.

  Static heuristic information has the advantage that it is easy to compute, and it has to be computed *only* once (at the ACO algorithm initialization step). As a result, static heuristic information leads to a significant savings of computation time.

  Dynamic heuristic information has to be computed at each step of the ant’s walk through the search space, i.e., dynamic heuristic information depends on the partial solution constructed so far. This leads to a higher computational cost that may be compensated by higher quality heuristic information. This approach may leads to higher quality solutions at the cost of higher computational complexity.

- **ACO and Local Search:**

  When coupled with local search algorithms, ACO algorithms perform best in many applications to NP-hard optimization problems. In fact, the ant produced solutions are locally optimized by an adequate local search process and these locally optimized solutions are used in the pheromone update. But generating initial solutions for local search algorithms is a hard task.

  Despite the fact that the use of local search with ACO algorithms is crucial to achieving best performance, it is important to note that ACO algorithms have achieved very good performance where local search algorithms cannot be applied easily (such as the application of ACO to the shortest common supersequence problem (Michel & Middendorf, 1999)).
- **Candidate Feature Lists:**

  If ants have a large number of possible moves from which to choose, then the computational complexity increases. In such situations, candidate feature lists constitute a smaller set of promising features of the current solution. Feature lists are created using a priori knowledge of the problem to be solved.

- **Number of Ants:**

  Generally speaking, although a single ant may generate a solution, the number of ants should be greater than one, and most of the time, the number of ants is set experimentally (Dorigo, Maniezzo & Colomi, 1996).

### 6.4 Ant Colony Optimization Based Feature Subset Selection Steps

To solve the FSS problem using the ACO algorithm, we have considered the following:

1. The FSS search space (features) is represented by a weighted graph (nodes with edges connecting them), where the nodes represent features and the edges denote the choice to select the next features. An optimal subset of features can be searched by an ant that traversed through the graph where a predefined number of nodes (features) are visited (selected) that satisfy a traverse stopping criterion.

2. The pheromone trail updating is defined to lay an amount of pheromone proportional to the quality of the best solutions achieved. On the other hand, pheromone trail updating is defined to lay more pheromone values for all the features in any high quality solution whose performance effectiveness is better than a predefined effectiveness value.

3. The heuristic information is defined by Chi-square statistic scores for the features in the search space. Choice of the Chi-square statistic is influenced by its performance effectiveness in the Arabic TC task (see Mesleh, 2007a).
4. The pheromone trail values and the Chi-square heuristic information scores are combined to form a probabilistic feature selection named “Chi-square based Feature Selection Probability” (CHIFSP).

5. $\alpha$ and $\beta$ are used to balance exploration and exploitation.

6. Because of the huge number of features in TC tasks, feature candidate lists are created using Chi-square statistic scores, we have selected a number of top features (the number of features in each candidate list is set to ten times the number of features in each feature subset).

6.5 Proposed Ant Colony Optimization-Based Feature Subset Selection

Steps

The FSS process may be reformatted into an ant colony optimization suitable problem. Figure 6.1 illustrates this scenario, the ant is currently at node “a”, and has a choice of which feature to select next to its path (paths are represented by dotted lines). The ant chooses feature “b”, this section is based on some “probabilistic feature selection” value, then the ant chooses feature “c” and then feature “d”. Upon arrival at feature “d”, the ant terminates its traversal (a stopping criterion is met – a suitably high classification accuracy has been achieved with the current feature subset) and outputs the current subset of features (a, b, c, and d).

A suitable heuristic desirability of travelling between features could be any feature subset evaluation function. In our proposed ACO-Based FSS method, we have selected Chi-square statistic as heuristic information.

A mount of pheromone is associated with each feature; this pheromone level can be updated according to some goodness measure. In our proposed ACO-Based FSS
method, this goodness measure is the effectiveness of the SVM classifier (macr-averaging $F_1$ measure).

Figure 6.1: Ant Colony Optimization Representation for Feature Subset Selection

The heuristic information (Chi-square statistic) and the pheromone levels associated with features are combined to form a probabilistic transition rule $P_{ij}^k(t)$, denoting the probability of an ant at feature $i$ choosing to select feature $j$ at time $t$:

$$P_{ij}^k(t) = \begin{cases} 
\frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{if } j \in J_i^k \\
0 & \text{otherwise}
\end{cases}$$

Where:

- $k$ is the number of ants.
When an ant at feature \( i \), \( \eta_{ij} \) is the heuristic desirability (Chi-square statistic score) to select feature \( j \).

\( \tau_{it}(t) \) is the amount of pheromone associated with the edge between feature \( i \) and feature \( j \).

\( J^k_i \) is the set of neighbor features of feature \( i \) which have not yet been visited by the \( k \)'s ant.

\( \alpha > 0 \) and \( \beta > 0 \) are two tuning parameters to determine the relative importance of the pheromone levels and the heuristic information (the choice of these tuning parameters is determined experimentally).

In our proposed ACO-Based FSS method, this probabilistic transition rule is the Chi-square based Feature Selection Probability (CHIFSP), the CHIFSP is defined as follows:

\[
CHIFSP_{i}^{S_j} = \begin{cases} 
\frac{[\tau_{ij}]^{\alpha}[CHI_{i}^{S_j}]^{\beta}}{\sum_{g \in \text{visited}} [\tau_{ig}]^{\alpha}[CHI_{i}^{S_j}]^{\beta}} & f_i \notin S_j \\
0 & \text{otherwise}
\end{cases}
\]

Where:

- \( CHI_{i}^{S_j} \) is the local importance of feature \( f_i \) given the feature subset \( S_j \).

\( CHI_{i}^{S_j} \) is measured using the Chi-square statistic formula (see section 4.4.1 Chi-square statistic for more details).

The ACO-Based FSS method starts by generating a number of ants, each ant selects one random feature. From these initial positions, each ant traverses edges probabilistically (Chi-square based Feature Selection Probability) until a specific stopping criterion is
satisfied. The resulting subsets of features are evaluated by SVM classifier (macro-averaging F₁ measure is used as a goodness measure). If an optimal subset is found or the algorithm has executed a certain number of times, then the feature subset selection process halts and outputs the best subset of features encountered. If neither condition holds, then the pheromone is updated, a new set of ants are created and the feature subset selection process iterates once more.

The pheromone trails are updated according to our proposed Text Classification Pheromone Update Formula (TCPUF), the TCPUF is defined as follows:

\[
\tau_i = \begin{cases} 
\rho \tau_i + \Delta \tau_i + w \Delta \tau_i & \text{if } f_i \in EBS_j \\
\rho \tau_i + \Delta \tau_i & \text{otherwise}
\end{cases}
\]

where:

- \( \tau_i \) is the pheromone level associated with each feature.
- \( \rho \) is a coefficient such that \((1 - \rho)\) represents the evaporation of the pheromone level. \(0 \leq \rho < 1\), \(\rho\) must be set to a value \(< 1\) to avoid unlimited accumulation of trails (\(\rho\) is determined experimentally).
- Elitist Best Solution (\(EBS_j\)) is any solution \(S_j\) (any subset of features) among the top best solutions (TBS) that outperformed BF₁ (where BF₁ is a predefined effectiveness value). In other words, any high quality solution (any subset of features) whose performance effectiveness is better than a predefined effectiveness value is considered as a member of \(EBS_j\). As a result, extra pheromone values shall be awarded to any solution that belongs to \(EBS_j\). This extra pheromone value is defined by \(w \Delta \tau_i\), where \(w\) is the performance effectiveness of the corresponding solution \(S_j\).
• \( f_i \) be a feature indexed by \( i \) (feature \( i \)).

• \( \Delta \tau_i \) (amount of pheromone change for each feature) is defined by:

\[
\Delta \tau_i = \begin{cases} 
\frac{\max_{g \in \text{TBS}} (F_{1g}^i) - F_{1,j}}{\max_{h \in \text{TBS}} \left( \max_{g \in \text{TBS}} (F_{1g}^i) - F_{1,h}^i \right)} & f_i \in \text{TBS}_j \\
0 & \text{otherwise}
\end{cases}
\]

• \( f_i \in \text{TBS}_j \) means that only the top best solutions (TBS) are used to calculate \( \Delta \tau_i \) values.

Below are the main steps of our proposed ACO-Based FSS algorithm:

**Step 1 - Initialization:**

Initially, the ant colony algorithm parameters are initialized:

• The amount of pheromone change for each feature \( \Delta \tau_i \) is set to zero (where \( i \) is a feature index, \( i \in [0, N] \), and \( N \) is the total number of features in the feature space).

• The pheromone level associated with each feature is initialized to some constant value \( \tau_i = 1 \).

• The desired macro-averaging \( F_1 \) measure is set to some threshold value (\( \text{BF}_1 = 88.11 \), where 88.11 was the best achieved SVM classifier effectiveness in term of macro-averaging \( F_1 \) measure).

• Define the number of solutions (number of ants - NAs).

• Define the number of Top Best Solutions (TBS).

• Define the maximum number of iterations (the maximum number of iteration is used as a stopping criterion).
- Chi-square statistic scores are pre-calculated for all the features of the original feature set (static heuristic information).

**Step 2 – generation ants for initial iteration:**

For each solution (ant) \((ant_i : i = 1 : NAs)\), randomly select a specific number of features (in our experiments, we have selected 140, 160 and 180 features for each ant).

**Step 3 – Evaluation solutions:**

For each solution \((ant_i : i = 1 : NAs)\), run the classifier (SVM text classifier) to evaluate the goodness of features in solution \(i\). Evaluation used the macro-averaging \(F_1\) measure.

**Step 4 – Stopping criterion:**

If a predefined stopping criterion is met (when the ACO Based FSS algorithm iterates more than the defined maximum number of iterations)

Then:

1. Stop the Ant Colony Optimization-Based Feature Subset Selection algorithm.
2. Return the best subset of features.

Else:

1. Update the pheromone levels associated with all features. Pheromone update is based on our proposed Text Classification Pheromone Update Formula (TCPUF).
2. Select new features for the \(NAs\) ants for the next iteration. Selection is defined by the Chi-square based Feature Selection Probability (CHIFSP).
3. Go to evaluation Step 3.
CHAPTER 7: EXPERIMENTAL RESULTS

7.1 Arabic Data Collection

Since there is no publicly available Arabic TC corpus to test the effectiveness of our SVM text classifier and to evaluate the effectiveness of the proposed Ant Colony Optimization Based FSS method, we have used an in-house corpus collected 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 corpus contains 1445 documents that vary in length. These documents fall into nine classification categories that vary in the number of documents. In this Arabic dataset, each document was saved in a separate file within the directory for the corresponding category, i.e., the documents in this dataset are single-labeled. Table 7.1 shows the number of documents in each category.

Table 7.1: TC Arabic Dataset.

<table>
<thead>
<tr>
<th>Category</th>
<th>Training Texts</th>
<th>Testing Texts</th>
<th>Total Number of Documents for Each Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>47</td>
<td>23</td>
<td>70</td>
</tr>
<tr>
<td>Economics</td>
<td>147</td>
<td>73</td>
<td>220</td>
</tr>
<tr>
<td>Education</td>
<td>45</td>
<td>23</td>
<td>68</td>
</tr>
<tr>
<td>Engineering</td>
<td>77</td>
<td>38</td>
<td>115</td>
</tr>
<tr>
<td>Law</td>
<td>65</td>
<td>32</td>
<td>97</td>
</tr>
<tr>
<td>Medicine</td>
<td>155</td>
<td>77</td>
<td>232</td>
</tr>
<tr>
<td>Politics</td>
<td>123</td>
<td>61</td>
<td>184</td>
</tr>
<tr>
<td>Religion</td>
<td>152</td>
<td>75</td>
<td>227</td>
</tr>
<tr>
<td>Sports</td>
<td>155</td>
<td>77</td>
<td>232</td>
</tr>
</tbody>
</table>

Dataset total number of articles 1445
7.2 Arabic Dataset Pre-processing

Pre-processing aims to transform the Arabic text documents into a form that is suitable for the classification algorithm. The Arabic documents are processed according to the following steps (Mesleh, 2007c; Guo et al., 2004; Benkhalifa, Mouradi & Bouyakhf, 2001):

- Each article in the Arabic dataset is processed to remove digits, numbers, hyphens, and punctuation marks.
- We have followed (Samir, Ata & Darwish, 2005) in the normalization of some Arabic letters: we have normalized the letters “ء” (hamza), “ي” (aleph mad), “ي” (aleph with hamza on top), “ـ” (hamza on w), “ي” (alef with hamza on the bottom), and “ئ” (hamza on ya) to “أ” (alef). The idea behind this normalization is that all forms of hamza are represented in dictionaries as one form and people often misspell different forms of aleph. We have normalized the letter “ق” to “ي” and the letter “ً” to “ٍ”. The reason behind this normalization is that there is not a single convention for spelling “ق” or “ي” and “ً” or “ٍ” when they appear at the end of an Arabic word.
- All the non Arabic texts were removed.
- Arabic function words (such as “อาท”, “อาท”, “อาท”, “อาท”, etc.) were removed. The Arabic function words (stop words) are the words that are not useful in IR systems, e.g. pronouns, articles and prepositions.
- Stemming: For Arabic documents, Aljlayl and his colleagues (Aljlayl et al., 2001) suggested that lightly stemmed words, where only common prefixes and suffixes are discarded, are perhaps better indexing terms for Arabic information retrieval tasks. In this work, we have not applied any stemming, because it is not always beneficial for text classification, since many terms may be conflated to the same root form (Hofmann, 2003).
However, we have tested the effect of light stemming on Arabic TC tasks with SVM classifier\textsuperscript{28}.

- The vector space representation (Salton, Wong & Yang, 1975) is used to represent the Arabic text articles, where term frequency and document frequency are obtained, then inverse document frequency is calculated. Then we followed (Salton & Buckley, 1988) in combining term frequency and inverse document frequency to weight text terms. Lastly, each document feature vector is normalized to unit length and the $IDF.TF$ is calculated.

### 7.3 Text Classification Methods

For the following reasons, we have selected support vector machine as a principal learning algorithm to build an Arabic TC system:

- Prior studies found SVM to have the best performance for TC (mainly for English article classification tasks) (Sebastiani, 2002; Yang & Liu, 1999; Dumais et al., 1998). In this work, we have implemented SVM classifier to classify Arabic articles.

- When evaluated the effect of many FSS methods for Arabic TC tasks, previous studies have not considered SVM which was already known to be superior to the classifiers they have studied (see Chapter 2). In this work, we have focused on support vector machine classifier to present a comparative study of feature subset selection methods for the high-dimensional domain of Arabic text classification.

However, for comparison purposes, we have implemented $k$NN and Naïve Bayes classifiers.

\textsuperscript{28} See (Mesleh, 2007a, Mesleh, 2007b)
7.4 Feature Subset Selection

We have implemented our proposed Ant Colony Optimization-Based Feature Subset Selection (ACO Based-FSS) algorithm. The proposed ACO-Based FSS algorithm adapted Chi-square statistic as heuristic information and the effectiveness of the SVM classifier as a guide to better selection of features for each category.

And to study the effect of FSS methods on SVM classifier (for Arabic TC tasks), we have implemented six classical FSS methods (Chi-square, IG, NGL, GSS, MI and OR).

7.5 Text Classification Evaluation

Text classification performance is always considered in terms of computational efficiency and classification effectiveness. When categorizing a large number of documents into many categories, the computational efficiency of the TC system must be considered. This includes the feature selection method and the classifier learning algorithm.

TC effectiveness (Baeza-Yates and Ribeiro-Neto, 1999) is measured in terms of Precision, Recall, and the $F_1$ measure. Denote the precision, recall and $F_1$ measures for a class $C_i$ by $P_i$, $R_i$ and $F_i$, respectively. We have:

$$P_i = \frac{TP_i}{TP_i + FP_i}$$

$$R_i = \frac{TP_i}{TP_i + FN_i}$$

$$F_i = \frac{2P_iR_i}{P_i + R_i} = \frac{2TP_i}{FP_i + FN_i + 2TP_i}$$

where:
- TP\(_i\) (true-positive): number of documents correctly assigned.
- FP\(_i\) (false positives): number of documents falsely accepted.
- FN\(_i\) (false-negative): number of documents falsely rejected.

To evaluate the classification performance for each category, precision, recall, and the F\(_1\) measure are used. To evaluate the average performance over many categories, the macro-averaging F\(_1\) (F\(_{1}^M\)), micro-averaging F\(_1\) (F\(_{1}^\mu\)), macro-averaging precision (macroP), micro-averaging precision (microP), macro-averaging recall (macroR) and micro-averaging recall (microR) are used and defined as follows:

\[
F_{1}^M = \frac{2\sum_{i=1}^{|C|} R_i \sum_{i=1}^{|C|} P_i}{N[\sum_{i=1}^{|C|} R_i + \sum_{i=1}^{|C|} P_i]}
\]

\[
F_{1}^\mu = \frac{2\sum_{i=1}^{|C|} TP_i / (\sum_{i=1}^{|C|} FP_i + \sum_{i=1}^{|C|} FN_i + 2\sum_{i=1}^{|C|} TP_i)}
\]

\[
\text{macroR} = \frac{\sum_{i=1}^{|C|} R_i}{|C|}
\]

\[
\text{microP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)}
\]

\[
\text{macroP} = \frac{\sum_{i=1}^{|C|} P_i}{|C|}
\]

\[
\text{microR} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}
\]
In this work, we focused on macro-averaging precision, macro-averaging recall and macro-averaging F$_1$ measure.

7.6 Text Classification Experimental Results

In all experiments, we have used the Arabic dataset described above for training and testing the Arabic text classifier. In addition to the mentioned pre-processing steps, we have filtered all terms with term frequency less than some threshold. (The threshold is set to three for positive features and set to six for negative features in training documents).

For each text category (see Table 7.1), one third of the articles were randomly specified and used for testing and the remaining articles were used for training the Arabic classifier (SVM classifier).

We have used an SVM package, TinySVM, with the soft-margin parameter $C$ set to 1.0 (other values of $C$ have shown no significant changes in results).

7.6.1. SVM Classifier with Chi-square FSS Experiments

In the first set of experiments, we have not applied any stemming processes. While conducting many experiments, we have tuned the Chi-square FSS method to achieve a better macro-averaging F$_1$ measure. After measuring the Chi-square scores for each feature, Chi-square tuning is conducted using two different approaches:

- Selecting the features with top-ranked Chi-square scores:

---

29 These results were published in (Mesleh, 2007a; Mesleh, 2007b).
The best SVM classification results were achieved when selecting the top 162 features (terms) for each classification category. We have noted that increasing the number of terms does not enhance the SVM classification effectiveness and makes the training process slower. On the other hand, the SVM classification effectiveness is negatively affected when the number of terms for each category is decreased.

- Selecting the features with a Chi-square score that is greater than a predefined Chi-square threshold value:

While conducting some other set of TC experiments, we have selected the features whose Chi-square scores are greater than the predefined Chi-square threshold. In this approach, an unequal number of terms is selected for each classification category. Using this approach, we could not achieve better results than those achieved using the previous approach (where the top Chi-square ranked terms were selected for each category).

Table 7.2 shows the results of the SVM classifier with the Chi-square FSS method. Results are shown in terms of Precision, Recall and the $F_1$ measure. The Macro-averaging $F_1$ score is 88.11 and the Micro-averaging $F_1$ score is 90.57.

Table 7.2: SVM results with Chi-square feature subset selection.

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>78.57143</td>
<td>68.75</td>
<td>73.33333</td>
</tr>
<tr>
<td>Economics</td>
<td>93.02326</td>
<td>71.42857</td>
<td>80.80808</td>
</tr>
<tr>
<td>Education</td>
<td>85.71429</td>
<td>85.71429</td>
<td>85.71429</td>
</tr>
<tr>
<td>Engineering</td>
<td>97.36842</td>
<td>97.36842</td>
<td>97.36842</td>
</tr>
<tr>
<td>Law</td>
<td>92.85714</td>
<td>81.25</td>
<td>86.66667</td>
</tr>
<tr>
<td>Medicine</td>
<td>95.06173</td>
<td>98.71795</td>
<td>96.85535</td>
</tr>
<tr>
<td>Politics</td>
<td>90</td>
<td>76.27119</td>
<td>82.56881</td>
</tr>
<tr>
<td>Religion</td>
<td>96.1039</td>
<td>98.66667</td>
<td>97.36842</td>
</tr>
<tr>
<td>Sports</td>
<td>100</td>
<td>85.71429</td>
<td>92.30769</td>
</tr>
</tbody>
</table>

Macro-averaging $F_1$ measure $= 88.11$
Micro-averaging $F_1$ measure $= 90.57$
7.6.2. Light Stemming Effect on the SVM Classifier with Chi-square FSS Experiments

In order to evaluate the effect of light stemming on Arabic TC tasks for Arabic articles. Following (Samir, Ata & Darwish, 2005) in the use of light stemming to improve the performance of Arabic TC tasks, we have adapted Larkey’s stemmer (Larkey, Ballesteros & Connell, 2002) to remove the suffixes and prefixes from the Arabic terms.

Unfortunately, we have concluded that light stemming does not improve the performance of our Arabic TC system, the Macro-averaging F\textsubscript{1} measure drops to 87.1.

As mentioned before, stemming is not always beneficial for text classification problems (Hofmann, 2003). This may justify the slight drop in the Macro-averaging F\textsubscript{1} measure.

7.6.3. Comparing the SVM Classifier with \(k\)NN and Naïve Bayes classifiers

For comparison purposes, we have used the same pre-processing steps to implement the Naïve Bayes and \(k\)NN classifiers. As Shown in Table 7.3, it is obvious that the SVM classifier outperforms the Naïve Bayes and \(k\)NN classifiers.

<table>
<thead>
<tr>
<th>Classifier Type</th>
<th>Macro-averaging F\textsubscript{1}-measure</th>
<th>Micro-averaging F\textsubscript{1}-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>88.11</td>
<td>90.57</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>82.09</td>
<td>84.54</td>
</tr>
<tr>
<td>(k)NN</td>
<td>75.62</td>
<td>72.72</td>
</tr>
</tbody>
</table>

30 These results were published in (Mesleh, 2007a; Mesleh, 2007b).

31 These results were published in (Mesleh, 2007a; Mesleh, 2007b).
7.6.4. FSS Comparison Study with the SVM Classifier for Arabic Articles\textsuperscript{32,33}

We have conducted four Arabic text classification experiments without any FSS method, i.e., all the features are used (the results of these experiments are referred as the original classifier). For each experiment and for each text category, we have randomly specified one third of the articles and used them for testing while the remaining articles are used for training the Arabic classifier. For each experiment, we have recorded the scores of macro-averaging precision, macro-averaging recall and macro-averaging $F_1$ measure. Finally, the results were averaged over the four runs.

Moreover, we have conducted four groups of Arabic text classification experiments to fairly compare the effect of six FSS methods on Arabic TC tasks; these FSS methods include Chi-square, NGL, GSS, OR, IG and MI. For each group and for each text category, one third of the articles were randomly specified and used for testing and the remaining articles were used for training the Arabic classifier. Each FSS method has conducted three feature subset selection runs: the first run selects the best 180 features, the second run selects the best 160 features, and finally the third run selects the best 140 features. As mentioned before, for each experiment we recorded the scores of macro-averaging precision, macro-averaging recall and macro-averaging $F_1$ measure. Finally, the results were averaged over the four experiment runs for each of the three different sizes of subset features.

Figure 7.1 shows the macroP values for the SVM classifier with the six different FSS methods at different sizes of subset features. Compared with the original classifier

\textsuperscript{32} FSS comparative study results were published in (Mesleh, 2007c), we have compared five FSS methods (Chi-square, NGL, GSS, OR and MI).

\textsuperscript{33} In addition to the FSS methods in (Mesleh, 2007c), we have added a new FSS method to the comparison (IG), results were published in (Mesleh, 2007d).
(without feature selection, i.e., all the 78,699 features are used for training the SVM classifier), only the Chi-square and NGL FSS methods perform better.

However, Chi-square is more stable than the NGL method (the Chi-square FSS method outperforms the original classifier at the three different feature subset sizes). However, the best SVM classification macroP result is obtained with the NGL FSS method (93.14 when selecting the best 180 features).

![Figure 7.1: Macro-averaging precision values for SVM classifier with the six FSS methods at different sizes of features.](image)

In Figure 7.2, we show the macroR results. It is observed that all the FSS methods outperform the original classifier. Chi-square, NGL and GSS performed much better than OR and MI. However, the best classification macroR result is obtained with Chi-square (84.00 when selecting the best 140 features).
Figure 7.2: Macro-averaging recall values for SVM classifier with the six FSS methods at different sizes of features.

Figure 7.3 shows the macro-averaging $F_1$ results. It is clear that all the FS methods outperformed the original classifier. Chi-square, NGL and GSS performed much better than OR and MI. However, Chi-square outperformed NGL and GSS, and achieved its best macro-averaging $F_1$ measure result when selecting the best 160 features.

7.7 ANT COLONY OPTIMIZATION BASED FSS RESULTS

To study the effect of the ACO-Based FSS algorithm, we have conducted four groups of Arabic TC experiments, for each group and for each text category, one third of the

---

articles were randomly specified and used for testing and the remaining articles were used for training the Arabic classifier.

![Macro-averaging F1 values for SVM classifier with the six FSS methods at different sizes of features](image)

Figure 7.3: Macro-averaging F₁ values for SVM classifier with the six FSS methods at different sizes of features

The proposed ACO-Based FSS algorithm has conducted three feature subset selection runs: the first run selects the best 180 features, the second run selects the best 160 features, and finally the third run selects the best 140 features. As mentioned before, for each experiment we recorded the scores of macro-averaging precision, macro-averaging recall and macro-averaging F₁ measure. Finally, the results were averaged over the four experiment runs for each of the three different sizes of subset features.

We have set the parameters of the proposed ACO-Based FSS method as follows:
- Number of ants (number of solutions) =30, however, increase the number of ants has shown no significant changes in SVM classification results.

- Maximum number of iterations is initialized to 30 (bigger values have made the selection process slower and have shown no significant improvement in SVM classifier results).

- Number of best solutions is set to 10 (other values have shown no significant changes in results).

- $\rho$ is set to 0.25 (other $\rho$ values have shown no significant changes in results).

- $\alpha$ and $\beta$ are set to 1 and 0.8 respectively. We have changed their values from 0.2 to 1.0 (0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0). After conducting many experiment, we found that best results were achieved with $\alpha = 1.0$ and $\beta = 0.8$.

Figures 7.4, 7.5 and 7.6 show the macroP, macroR and macro-averaging F1 measure results for the SVM classifier with the seven FSS methods at different feature subset sizes.

It is obvious that the proposed ACO-Based FSS algorithm has significantly outperformed the original classifier (where all the 78699 features are used for training the SVM classifier) in terms of macro-averaging precision, macro-averaging recall and macro-averaging F1 measure.

Comparing with the other six classical FSS methods, the proposed ACO-Based FSS algorithm has improved the performance of the Arabic TC system.

When running the experiments for Chi-square, IG, OR, MI, GSS and NGL FSS methods, the performance of SVM classifier was bad for small categories (such as Computers and Education categories). For this reason, our proposed ACO-Based FSS algorithm has
been *only* run to optimize the feature subset selection for the “Computers” and “Education” categories.

The macro-averaging precision, macro-averaging recall and macro-averaging $F_1$ scores (achieved by the ACO-Based FSS algorithm) indicate that the proposed ACO-Based FSS algorithm has positively affected the effectiveness of the Arabic TC system even if the optimization was selectively applied to particular categories.

![Macro-averaging Precision values for SVM classifier with the seven FSS methods at different sizes of features.](image)

**Figure 7.4:** Macro-averaging Precision values for SVM classifier with the seven FSS methods at different sizes of features.

Table 7.4 shows SVM performance with different FSS methods. It is clear that ACO-Based FSS algorithm has optimized the classifier’s effectiveness in terms of macro-averaging precision, macro-averaging recall and macro-averaging $F_1$ measure.
Comparing with the best scores which are achieved by the traditional feature subset selection methods, the improvement achieved by the ACO-Based FSS algorithm (see Table 7.4) in term of the macro-averaging precision was +2.47, the improvement in term of the macro-averaging recall was +1.51 and the improvement in term of macro-averaging F1 measure was +2.07.

Comparing with the original classifier, the proposed ACO-Based FSS algorithm (see Table 7.4) has improved the classifiers effectiveness in terms of macro-averaging precision (+2.80), macro-averaging recall (+20.28) and macro-averaging F1 measure (+15.57).

![Figure 7.5: Macro-averaging Recall values for SVM classifier with the seven FSS methods at different sizes of features.](image-url)

89
Figure 7.6: Macro-averaging $F_1$ measure values for SVM classifier with the seven FSS methods at different sizes of features.

Table 7.4: SVM Performance Comparison Using Best 160 Features: Selected by Different FSS methods.

<table>
<thead>
<tr>
<th>FSS Method</th>
<th>Macro-averaging Precision</th>
<th>Macro-averaging Recall</th>
<th>Macro-averaging $F_1$-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chi-square</td>
<td>91.64</td>
<td>83.98</td>
<td>87.54</td>
</tr>
<tr>
<td>NGL</td>
<td>91.67</td>
<td>82.49</td>
<td>86.50</td>
</tr>
<tr>
<td>GSS</td>
<td>90.68</td>
<td>83.39</td>
<td>86.50</td>
</tr>
<tr>
<td>OR</td>
<td>84.03</td>
<td>74.93</td>
<td>78.75</td>
</tr>
<tr>
<td>MI</td>
<td>87.74</td>
<td>72.51</td>
<td>78.53</td>
</tr>
<tr>
<td>IG</td>
<td>85.11</td>
<td>74.18</td>
<td>78.71</td>
</tr>
<tr>
<td>W/O FSS</td>
<td>91.34</td>
<td>65.21</td>
<td>74.04</td>
</tr>
<tr>
<td>ACO</td>
<td>94.145</td>
<td>85.49</td>
<td>89.61</td>
</tr>
</tbody>
</table>
CHAPTER 8: CONCLUSIONS & FUTURE DIRECTIONS

In this thesis, an SVM text classifier for Arabic articles was implemented. It was obvious that the SVM classifier outperformed the Naïve Bayes and $k$NN classifiers. Then, the effect of light stemming on Arabic TC tasks was evaluated with SVM classifier. We concluded that light stemming does not improve the performance of Arabic TC classifier.

In fact, it is sometimes claimed that feature subset selection is unnecessary for SVM classifiers. However, it was obvious that feature subset selection improved the performance of Arabic SVM text classifier.

- Better recall was achieved by all the feature subset selection methods (Chi-square, GSS, NGL, OR, IG and MI).
- Better $F_1$ measure was achieved by all the feature subset selection methods.
- Better precision was achieved by Chi-square and NGL feature subset selection methods.

A new Ant Colony Based-Feature Subset Selection algorithm was presented. Compared to the six classical FSS methods, our ACO Based-FSS algorithm achieved better TC effectiveness (the new proposed ACO-Based FSS algorithm has outperformed the original classifier too). In this new ACO-Based FSS algorithm implementation, the ACO algorithm was adapted to handle the large number of features in TC tasks.

In future, we look forward to:

- Investigating the performance of the ACO-Based FSS method with other larger TC datasets.
- Testing the proposed ACO Based-FSS method on other text classifiers such as C5.0 decision tree algorithm.
- Comparing our Ant colony flavor with other Ant Colony algorithms (such as Max-Min).
- Instead of using Chi-square statistic as heuristic information for ACO algorithm, we are looking forward to trying other FSS methods such as MI.
- Investigating the usage of other optimization algorithms to handle FSS problem for Arabic language texts.
- Deeply investigating the effect of stemming on Arabic TC tasks.
- Involving Arabic semantics in FSS processes for Arabic TC tasks.
- The main limitation of this Arabic TC work is the unavailability of an Arabic TC corpus. We look forward to cooperating with other parties to build a bigger Arabic TC corpus.
- Generalizing the idea of ACO-Based FSS method and modify it to be a generalized feature subset selection tool (freeware) shall be published at the authors website.


[Subbotin & Oleynik, 2007] Subbotin, S., Oleynik, A.: Modifications of Ant Colony Optimization Method for Feature Selection, the Experience of Designing and


