
University of
Strathclyde
Glasgow

Classification of Arabic Extremist Web Content through Arabic Textual Analysis

---

A Thesis

Presented to

Department of Computer and Information Sciences

University of Strathclyde

In Partial Fulfillment

of the Requirements for the Degree

OF DOCTOR OF PHILOSOPHY

of Ph.D. of Computer Science (Ph.D.)

---

by

Haya Mesfer Alshahrani

2020

PREFACE

The basis for this research originally stems from recognizing the need for developing better methods of data classification in Arabic. As the world moves further into the digital age, the internet may become a source of dangerous and destructive information. Violent content will surely have a damaging effect on young minds. This raises an important question: How can we sort and classify this content? Our intention is to find ways to organize such content in Arabic and break down barriers of classification for future research.

DEDICATION

I dedicate this thesis . . .

All this effort and hard work . . .

This project, every letter Included, all the data gathered, all information written, proofs, evidence, assembling and references . . .

To my little precious one . . . **Deema**

Who discovered she suffers from cancer during my PhD studies,

and suffered pain . . .

We both suffered every fatigue and every burning chemotherapy session . . .

With the hope in her eyes, making me forgets everything . . .

Most of this research was written in the hospital's lobby . . .

Between the corridors of surgery rooms & treatment rooms . . .

Where my sole and main motivation and concern in this world was that light of hope in her beautiful eyes urging me to continue my work and finish this research.

ACKNOWLEDGMENT

I would like to acknowledge my thanks with all the efforts spent to my supervisor in my PhD Dr. George Weir, who contributed and helped me all the way in each word and moment of research, and in every stages of my study, by his sincere instructions and opinion.

I would like to express my thanks to my internal supervisor Dr. Alaaldine Hafez who was my other eye by contacting, supervising & helping me with his personal consultation.

I would also like to express my thanks and gratitude to Professor Meg, for proofreading my study with all sincerity and providing all efforts to support me.

I send my great thanks to the Viva committee and examiners, who supervised my research, and helped me improving and adjusting any deficiencies in the research.

I would like to thank Dr. Murray for organizing and running the discussion in a smooth and nice manner despite the current Covid 19 conditions.

I would like to thank Dr. David the internal examiner who supported me with his wise guidance and observations, which helped me to overcome some mistakes I made and to fill some limitations in my research, as his notes were an expert reviews, and he had an insightful eye that helped me improve the quality of my work.

I want also to thank Dr. Amir for his valuable remarks and his decent manner during the time of the discussion and for his advices to cover all the gaps in my research

I deeply thank the University of Strathclyde that gave me the opportunity to study for doctorate and awarded me the degree. And I want to thank the university of princess Nora who granted me the scholarship, and I especially thank Dr. Fatima Al- Shehri the director of the mutual supervision program for her endless support during my study.

Also I would like to present my thanks, gratitude and my deep feelings to my dear father who was always beside me since my early age and was my first fan and my everlasting supporter of everything I do.

I want also to present this success to my first role model, my beloved mother, who I believe she is the owner of this presentation. And also to present my thanks and gratitude to my husband who stood by me and supported me since the beginning of this journey to the end, and always surrounded me with his love, care and help. And from the bottom of my heart, I want to thank my family, and my kids who beard my absence and busyness during my study period.

I also dedicate this great work to my brothers and sisters and thank them so much for being there for me.

Finally, I want to thank everyone who helped me and supported me during this difficult and beautiful period at the same time.

ABSTRACT

Many scholars have attempted to study the written and spoken word of terrorist groups and individuals to understand the underlying motivation for terrorist acts. However, until today, those scholars have not made use of automated linguistic analysis programs, especially those focusing on Arabic corpus, in an attempt to understand the mentality behind terrorist acts. A contribution in this regard will be made.

The division and classification of texts is an important science of linguistics, whether Arabic or otherwise, it summarizes the effort and time consuming of humans to classify these language texts. The importance of this research stems from not only the importance of the classification itself, but also a classification based on the extreme orientation of these linguistic texts.

In this research, the researcher has tried to prove his new methodology based on the division and classification of Arabic texts, a classification that distinguishes them from others according to the identity of the speakers, whether as extremists or against extremism or as neutral people who do not have any ideas belonging to any terrorist or counter category. This methodology is a numerical methodology that relies on dividing speech using two different tools and then analyzing the results using more than six algorithms in the Wiccan program. The researcher got very good results that make the judgment on this methodology a resounding success.

This thesis aims to put forward a comprehensive and detailed classification system to categorize different Arabic-speaking website pages with unscrupulous intentions and questionable language. It uses three specific Arabic corpora, (Pro-terrorism, Anti-terrorism, and neutral), from more than 7000 Arabic text to construct corpus (1,000,000 words approx.) from different sites and sources.

The division and classification of texts is an important science of linguistics, whether Arabic or otherwise, it summarizes the effort and time consuming of humans to classify these language texts. The importance of this research stems not only from the importance of the classification itself, but also a classification based on the extreme orientation of these linguistic texts.

In this research, the researcher has tried to prove his new methodology based on the division and classification of Arabic texts, a classification that distinguishes them from others according to the identity of the speakers, whether as extremists or against extremism or as neutral people who do not have any ideas belonging to any terrorist or counter category. This methodology is a numerical methodology that relies on dividing speech using two different tools and then analyzing the results using more than six algorithms in the WEKA program. The researcher got very good results that make the judgment on this methodology success.

This thesis employs a quantitative approach by using different algorithms (supervised) to build a model for data classification by using manually categorized information. The classification algorithm used to construct the model uses quantitative information extracted by Posit or SAFAR textual analysis framework. This model functions with (58) features combined from Posit – n-grams and morphological SAFAR V2 POS tools. This model achieved more than (94 %) success in the level of precision.

This model uses Posit method to make appropriate changes to the code so it can deal with Arabic content, secondly SAFAR V2, which is more suited to the domain of Arabic being based on analyzing the morphology of the word, and therefore, it can highlight all the essential features overlooked in Posit. This model has manual classification, pre-processing steps and can apply eight different experiments using WEKA APIs, a GUI (Graphical user interface) application.

The research concludes that the best results reaching 94% precision have been achieved by combining Posit + SAFAR + (18 attributes Posit+ SAFAR N-Gram). Moreover, the most reliable results have been achieved by applying a Random Forest classification algorithm using regression. The research recommends working more on this topic and using new algorithms and techniques.

TABLE OF CONTENTES

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER ONE: INTRODUCTION

1.1 Over view

In this brief chapter, the research will explain itself. As this chapter contains all the principles that may be useful to the reader in understanding this research. Beginning with the explained introduction, which is considered as a gateway to the search, as it contains many semantic words and matters related to the search. It charts the path taken by the researcher to walk through to the main objectives of the research and reaches what he is trying to establish from the beginning to reach the final results of the approach used. This chapter also contains the basic research questions that the researcher seeks to answer and contains the contributions made by the research in the world of information security and how this approach can make a significant contribution to revealing the extremist content of Arabic texts.

1.2 Research Introduction

Extremists and terrorists are similar in their forms of expression, political activities and in their use of mass media. Nevertheless, extremism expresses political thoughts and programs that are rejected by most ordinary people. However, terrorism chooses the use of force as a means to achieve a political agenda (El Zein, 2015).

Extremism can be literally defined as "the quality or condition of extremism" or "advocating extremist views." A term defined in the religious or political sense to refer to an ideology that is very far from the prevailing attitudes of society, where the views of extremists' contrast with the opinions of moderate people (Mogahed, 2006).

There is no universal agreement on a definition, but it can be said that terrorism is intentional, politically motivated violence committed by underground groups or agents. In general, terrorism includes the use of threats or violence, to achieve religious, political, social, or ideological goals (Schmid, 2011).

Media are essential for extremists and terrorists. Since the early sixties, both groups have recognized the importance of the media as a vehicle to disseminate their propaganda achieves their goals and spread their word. Extremists and terrorists have succeeded in using the media for this purpose, despite leaders in their countries vainly trying to stop this. Using the worldwide accessibility of the Internet, social media and other platforms, extremists and terrorists have made it possible to provide unrestricted access their websites. Some of these sites are designed to attract would-be militants and sympathizers in Western countries. To support their causes, terrorist organizations utilize the internet to raise funds for finance.

In 2005, the Secretary-General of the United Nations created the CTITF (The United Nations Counter-Terrorism Implementation Task Force) to secure all necessary cooperation and concerted efforts of the United Nations regime in the field of anti-terrorism.

The last few years have seen a significant increase in the activities of radicalized extremists, launching terrorist attacks around the whole world. They exploit modern technologies such as the Internet and social media, widely used by the general public, to plan and maintain contact with their group (Alvari, Sarkar, & Shakarian, 2019).

Social media like Facebook and Twitter are currently being utilized by extremist groups to create direct contact with their worldwide groups. By the very nature of these applications (i.e., free and unregulated), encouraged extremists to quickly form virtual societies and disseminate their thoughts and their coaching tools without paying attention to the usual means of censorship in the general media (Alvari et al., 2019).

During this last year or so, social networks have started to intervene by implementing countermeasures against these groups. Twitter was considered the main promotional vehicle for ISIS, so in August 2016, Twitter started taking more stringent measures by closing more than 36,000 feeds that were believed to belong to ISIS (Alvari et al., 2019).

This thesis employs a quantitative approach by using different algorithms (supervised) to build a model for data classification by using manually categorized information. The classification algorithm used to construct the model uses quantitative information extracted by Posit or SAFAR textual analysis framework. During each experiment, the corresponding model was used to classify unseen data.

1.3 Research Importance

Fundamentally, the benefits of using data collected from social media depend on the factual accuracy of the statements being collected from the users or their groups. However, it was established that additional effective procedures such as utilizing algorithms to uncover clues in the content that points to violence automatically supported this feedback and improved its performance. Notwithstanding, the feedback resulted in social websites closing a significant number of accounts, however, it was not guaranteed to be accurate, because owners of the pending accounts can create new accounts and resume their activities, or are able to relocate to different social websites. More research is required to be carried out by governments in order to counter the radicalized extremists and stop, or at least reduce their threat (Alvari et al., 2019).

Therefore, is thesis employs a quantitative approach by using different algorithms (supervised) to build a model for data classification by using manually categorized information. The classification algorithm used to construct the model uses quantitative information extracted by Posit or SAFAR textual analysis framework. During each experiment, the corresponding model was used to classify unseen data. in fact, this technique aids identification of Arabic extremist websites, enabling authorities to remove them, thus preventing their harmful ideology on impressionable minds and help combat terrorism in the world.

1.4 Research Scopes and Questions

The thesis seeks to establish a comprehensive system to reveal any publication or entity that has malicious intentions emanating from extremism or seeks terrorism, and that is in various Arabic web pages. This is crucial to a comprehensive understanding of the extreme world. So, the main question that this message hopes to

answer is, "How can we improve and automate the classification of Arabic words in scenarios where the intention is vague or misleading?"

This central question can be broken down into two sub-questions:

a. “Can we label different occurrences of the same item (for example, word or phrase) under various contexts (Pro extremist – Anti extremist –neutral)?”

b. “Can we train robust and discriminative features to uncover any misleading emotional Arabic text, and properly categorize said text according to its underlying message, within a quantitative approach?”

1.5 Research Aim & Objectives

The main goal of this thesis is to put forward a comprehensive and detailed classification system to categorize different Arabic-speaking website pages with unscrupulous intentions and questionable language. The objectives of this research can be listed as follows:

- Visiting and determining several websites related to terrorism, neutral and anti-terrorism websites.
- Extracting keywords and Build corpus of Arabic words, this corpus is divided three categories (terrorist & extremist, anti-terrorist, neutral).
- Develop a new system based on the Posit and SAFAR tools that can distinguish between (terrorist & extremist, anti-terrorist, neutral) sites in Arabic language.
- Train the System the collected corpus using the machine learning feature
- Test the new system on new sites to determine the effectiveness and make any required modifications if it needed

1.6 Research Contributions

The main contributions of the thesis can be summed up as follows:

1- Building an individually combined corpus of (5,100) text files, and more than (1,000,000) Arabic words; some of them from extremist websites, others from anti-

extremist and neutral ones. Such a collective corpus was not available for this type of research, which is why we had to compile a corpus for ourselves, and it is hoped that further research will be carried out that will increase the corpus size and accuracy.

2- Introducing new enhanced POS (part of speech) from the Posit tool developed by Weir (2007, 2009); however, it was necessary to modify the code to deal with Arabic content.

3- Developing a classification model that functions with (58) features combined from Posit – n-grams and morphological SAFAR V2 POS tools. This model achieved more than (99 %) success in the level of precision.

1.7 Research Structure

This thesis is divided into an introduction and seven chapters; Chapter two presents a general overview and literature review of previous work and Arabic morphological classification. This covers some of the contributions to the main aspects of extremism classification and examines supervised modeling and feature selection and review in the same or similar fields, which was used to great benefit.

Chapter three introduces the new methodology used in this thesis, beginning with Data Preparation, compiling, and building of the Arabic corpus, and covering the classification methodology. Chapter four explains the implementation of this methodology detailing the code used to split the massive corpus into individual text files. Information preparation began by extracting quantitative information from Arabic corpus using Posit and SAFAR frameworks. Chapter five discusses the experiments and the setup required - starting with the software and ending with WEKA and how it was edited to fit with our approach. This chapter also explains all the details of POS and SAVAR and the attribute of N-gram.

Chapter six tackles the classification results, analysis of the eight hypotheses that have been put forward and the discussions. The first part of this chapter will detail the hierarchical system evolved from the idea of forming a table of the results. Within this table, eight test datasets are experimented by applying six classifiers. After that,

the results of applying these classifiers to those variants will be explained in detail. After that, these variants, and the reasons for using them will be discussed. The third part of this chapter contains the main discussion of the approach and the main aspects of the results. The results to those of an application previously custom-built by researcher will be compared.

In chapter, seven presents the conclusions after applying these experiments. A discussion any future work intended to be carried out, and the individual and separated utility applications and classifiers into one web-based application will be integrated.

CHAPTER TWO: BACKGROUND AND LITERATURE REVIEW

2.1 Background overview

This chapter discuss the most important terms and methods used in the research, with the most important related concepts and aspects. Extremism terms will be defined and its relationship to social media. I will also shed light on Arabic language processing and how this language is characterized by strength, abundance of meanings and topics related to data processing systems and machine learning. I will also write briefly about data systems and mining including classification process. I will cover some of the famous and used models in texts classification. Once you have finished reading the first and second chapters, you will have a sufficient perception of the research. You will also have a slight idea of all basic terms in this research; beginning with extremism and means used by people in this area. Also, will talk about some of the approaches through which the Arabic texts were classified, which is the basic idea of this research. It will talk about the most relevant of them and its results.

2.2 Definitions

2.2.1 Neutral text

A neutral text is a general text that tackles extreme issues neither positively nor negatively, and thus conveys general meanings as it tackles a neutral life topic either religious, political, sports, artistic, scientific. However, the text conveys a completely neutral orientation. These neutral topics are often far from racism, hatred, or any other topic related to any affiliations or group that calls from extremism and one man's ideology (Ogun, M. N. (2012).

Spoken and written speech was one of the most important means used to express what is going on in man's mind. Therefore, the importance of messages addressed through texts is highlighted. Any text may preserve its content and orientation. However, a neutral content is always a content that does not belong to any political or religious extremism. (Hamzah,S, Mohamed (2012).

However, an extreme text is one that belongs to a particular religious or political policy, and implies an attempt to incite this trend. ‘Extremism’, as defined by Sheikh Mohamed Habib, is being far from moderation regarding everything. An extremist is a person who is far from the sound approach; moderating in everything (Hamzah,S, Mohamed (2012).

- In his book ‘Countering terrorism & extremism, De-radicalization’, Major Muhammad Hamza defined ‘extremism’ linguistically as being in the far extreme away from the middle, like sitting or standing or walking at the edge.. He then tackles extremism in religion, politics, behaviors or ideologies. It means also the end of anything and being extreme means excessiveness and going beyond moderation.

  Moreover, extremism is a normative term that depends on religious and legal norms, traditions and standards of a society, and that is why such a concept is debated among societies. For example, a flexible society may consider some certain behaviors legitimate, while another fundamental one does not.

  However, Social wise, extremism is to deviate from the ideals, traditions and customs of the society, to either the right, or left, positively or negatively.

  Legal wise, it means violation or deviation from the social or legal controls that govern the behavior of individuals within a society, and this deviation vary as an act that society denounces, or a crime that falls under the law.(Hamzah,S, Mohamed (2012).

2.2.2 Extremist

The term extremist and terrorism, which was widely used recently, indicates a certain type of crime (it is that occurs sometimes through violence or threatening, and perpetrators aim to compel the relevant authorities or bodies to perform an act or refrain from another that serves a political or national or personal interest, and thus jeopardize lives and wealth in case their demands are not met). Consequently, the mere fact that a crime is a crime of violence, even if this violence reaches its maximum, does not make it a terrorist crime as

per current definition, whatever the means used in such crime (Keene, S. D. (2011))

Thus far, the global community reaches no consensus upon a specific definition for ‘terrorism’ in spite of everyone agreeing to condemn and fight it. This is due to a clear reason that the terrorist, in some views, is fighting for freedom. Thus, a distinction must be made between national liberation struggle, and terrorism. However, the difficulty in such distinction is due to several reasons, including the following:

- The difficulty to reach an abstract definition of terrorism without introducing external elements; represented by divergent opinions about the legitimacy of the organizations and their activities.
- The difficulty in reaching international agreements or treaties in this regard due to the different interests of states, and that each is attempting to impose their point of view consistent with their principles, interests and historical backgrounds. All of this makes the goals and content of the agreements debatable.
- The conceptual confusion between political violence and terrorism, thus distinguishing can be hardly made between them, and it goes further to be mixed up with some forms of war or even ordinary crimes.
- Conceptual and terminological confusion between terrorism, as a phenomenon that has a specific role in political conflict, and as a form of political violence with many other forms of violence such as rebel movements, strikes and coups(Hamzah,S, Mohamed (2012).

As for anti-extremism texts, they are any texts that are against terrorism in all its forms, and defend peace and always condemn and abominate any violent, racism, or racial or human discrimination.

The neutral text is a general text that does not include any extreme subjects in its content positively or negatively. So, its general meaning will be related to a topic of life, whether religious, political, sports, artistic or scientific. However, the subject's orientation is completely neutral approach. Often neutral subjects are far removed from

racism, hatred, and any subjects in which they belong to any orientation or belong to any group that calls for extremism or a unifying ideology (Ogun, M. N. (2012).

Spoken and written speech was one of the most important means used to express what is going on in the minds of people. Hence, the importance of messages addressed through texts where any text that you may be able to preserve its content and orientation. Always neutral content is content that does not belong to any extreme political or religious orientation (Bin Hassan, M. H., & Pereire, K. G. (2006).

Extreme text is the text that belongs to a particular religious or political policy, and includes an attempt to incite this trend.

Extremism is as defined by Sheikh Mohamed Habib, the distance from mediating in everything. An extreme human is a person who is far from the correct and moderate approach in everything. Major Muhammad Hamzah also defined it in his book Combating Terrorism and Extremism and the Method of Intellectual Review. Extremism in the language means standing on the side, like extremism in standing, sitting, or walking(Hamzah,S, Mohamed (2012).

Then he turned to morale, such as extremism in religion, thought or behavior. Alternatively, it is the forelock and the end of everything, Extremism means reaching the side and exceeding the level of moderation and not mediating. - Extremism is a standard word that means violating the general or other line, which is defined by the traditions, customs, and legal and religious norms prevailing in society, which makes its concept subject of difference between societies, as the flexible environment may see legitimacy in certain actions while the tough see the opposite. It means the distance from the normal line of society, from a legal standpoint, it means departing or deviating from the social or legal controls that govern the behavior of individuals in society, this departing varies from an act that society denounces to an act, which constitutes a crime under penalty of law (Keene, S. D. (2011)..

2.2.3 Terrorism

These are the crimes that are sometimes committed through violence or the threat of it, its perpetrators aim to compel the authorities or bodies concerned to take action or refrain from an action that realizes political, national or private interests, and makes the lives of innocent people or their money at risk if their demands were not met.(Bin Hassan, M. H., & Pereire, K. G. (2006).

Consequently, the mere fact that crime is a violent crime 9, and even if this violence reaches, it does not involve it in the terrorist crime circle as it is meant by this word currently, and by whatever means used in it.

Until now, the global community has missed the consensus on a specific definition of terrorism, despite everyone agreeing to condemn and fight it, this is due to a clear reason that the terrorist in the view of some, is a warrior for the freedom from the point of view of others.

From here, we must distinguish between the fighters of the national liberation issues and the terrorists. The difficulty in this matter is due to several reasons, including the following:

It is difficult to reach an abstract definition of terrorism without introducing external elements of it, represented by divergent opinions about the legitimacy of the organizations and their activities. Difficulty in reaching international agreements or treaties in this regard due to the different interests of states and the attempt of each group of them to impose their point of view that is consistent with its principles, interests and historical backgrounds makes the goals and content of the agreements a disputed matter (Bin Hassan, M. H., & Pereire, K. G. (2006)

The confusion of political violence forms with terrorism, so that the separation between it and some forms of political crime and organized crime has become blurred. Rather, the matter goes beyond the confusion of the concept of terrorism with some forms of war or even ordinary crimes.

The confusion of the concept and meaning of terrorism as a phenomenon that has a specific role in political conflict, and a type of political violence with many other forms of violence such as rebel movements, insurrection and coups(Bin Hassan, M. H., & Pereire, K. G. (2006))

As against extremism, they are any texts that are against terrorism in all of its forms, and they defend peace and always abhor any act of violence, racism, or human or ethnic discrimination.(Bin Hassan, M. H., & Pereire, K. G. (2006)

2.3 Social Media and Extremism

In recent years, social media has afforded a dynamic form of communication and interaction between people around the world. Instant exchange of information and data among users has become easy. Making contacts and transferring ideas worldwide is now commonplace through services such as Facebook, Twitter, and Tumblr. Other varieties of media are also readily exchanged, e.g., via websites for transferring pictures, like Flicker and Imager, and websites for displaying videos such as Dailymotion, YouTube, and Vimeo. The popularity of social networking worldwide has a detrimental effect in that these platforms can be exploited by terrorist organizations as a vehicle to spread their doctrines and make available their training tools through direct communication with supporters around the world, with little fear of the censorship and supervision present in conventional regulated media (Agarwal & Sureka, 2015).

There are many paradigms of extremists and extremist organizations utilizing different social media and other Internet services. Jose Pimentel is an example of an extremist who has made extensive use of the Internet. Pimentel was captured preparing rudimentary bombs for attacks on police cars and postal offices in the United States. Tracking Pimentel's activities on the Internet showed that he was very active and was in regular contact with like-minded people, via activities on his Blogger website and YouTube channel. Anders Behring Breivik, from Norway, is another extremist who was utilizing the web and obtained information on how to create a destructive fertilizer bomb. He explicitly detailed his opposition to Islam and Socialism. Another example

of using the Internet to spread extremism ideology is Timur Abdul Wahab Al-Abdali, who, in 2010, killed himself with a suicide bomb in Stockholm. He also had engaged extensively on different social media websites like Facebook and YouTube (Johansson, Kaati, & Sahlgren, 2017).

Extremists have recognized the advantages offered by the Internet and have rapidly moved to social media chat to present their current thoughts, issues, and beliefs. They have adopted the simple usage, fast, inexpensive, secure, un-policed, and unmanaged platforms of the Internet. In consequence, they often succeeded in developing communities with similar beliefs, to induce the collecting and diffusion of information, inspiration to boost their income, and to defeat their opposition. Forums on the Internet have also allowed extremists to display their radical thoughts and extremist beliefs have been strengthened, instead of encouraging anti-extremist sentiments. These online areas provided stimulation and support to users, and extremist thoughts were reinforced, instead of meeting extreme opposition (Scrivens, Davies, & Frank, 2018).

With the dramatic growth in the online world and widespread use of information exchange, the manual procedures that had previously been used became almost impossible as measures to look for extremists, or potential terrorists, or indeed, users who publish radical views via the Internet and social media Web Pages. As a response, some researchers have turned to Machine Learning as a solution to help in quqiscrutinizing the vast quantities of data (Scrivens et al., 2017).

2.4 Arabic Language Processing and Machine Learning-Based Systems

Natural language processing (NLP) is an area of computer science that researches in developing theories, methods, and applications that can be used to understand, survey, and create the languages of the human beings to allow interaction among the people and the machine via writing or talking in a human language. In other words, NLP assists computers in accommodating the methods that people use in natural communication between humans (Marie-Sainte, Alalyani, Alotaibi, Ghouzali, & Abunadi, 2019).

After a sustained period of NLP on English and other natural languages, Arabic Natural Language Processing (ANLP) has recently grown to become an area of research interest, and numerous ANLP laboratories have been established. This growing interest has led to the development of many different ANLP applications, including word categorization, detection of spam and moral analysis (Marie-Sainte, Alalyani, Alotaibi, Ghouzali, & Abunadi, 2019).

Facing these advances in ANLP tools, are the fundamental issues with Arabic language, the set of letters of the Arabic language, and the role of diacritics that modify the vowels (Boukhatem, 2014).

Lately, ANLP implementations have advanced by utilizing Machine Learning (ML) algorithms. ML is a branch of artificial intelligence whose main target is to make computers capable of understanding without requiring explicit, comprehensive programming. ML has been successfully implemented in many application areas, including NLP generally and ANLP in particular (Boukhatem, 2014).

With the context outlined above, the importance of the subject addressed in the present research and its association with current events around the world, where many nations and individuals suffer from the dangers of extremism and terrorism. Those working on this issue, face significant challenges, especially about tracking the activities of extremism and terrorism organizations through the Internet and social websites. Also, how difficult it is when it publishes in Arabic in non-Arabic speaking countries and the challenges to detect it.

Arabic is a language with a rich cultural history. Computational Linguistic research applications of the Arabic language are more urgent today than they were a decade ago. In the next section, a look at Arabic morphology will be taken, with examples to illustrate how challenging and important this is when working with an ANLP application. Through this point, the issue of ambiguity will be appreciated with the massive quantity of terminology and terms of demystifying about Arabic morphology frequently used in social media websites. This is not intended to be a comprehensive account but may afford sufficient information that can be understand the scale of the challenge with Arabic tutoring.

Computational linguistics is an area that ranges from linguistics to computer science with the assistance of psychology and logic. It utilizes computers to help with language problems. It is considered a branch of computer science as well as in linguistics. However, cooperation must be between computer science and linguistics.

2.5 Arabic Morphology

Arabic is a morphologically powerful language; such morphological richness poses a challenge to any attempt to tag parts-of-speech (POS) in Arabic. Because of the morphological complexity, a single word in Arabic may be equivalent to a phrase or an entire sentence in English, which leads to great difficulty in data analysis. Since each word can be complicated, it will be divided into segments or morphemes, and then, each morpheme will then take its individual POS tag (Kübler & Mohamed, 2012).

For example:

وسى+كتب+ونها wasaya+ktub+uwnhA.

In Buckwalter transliteration, Eng (In addition, they will write it) is assigned the following POS tag:

[CONJ (و)+

FUTURE PARTICLE (س)+

VERB PREFIX (ي) +

VERB (كتب) +

VERB SUFFIX MASCULINE PLURAL

Third PERSON (ون) +

OBJECT PRONOUN FEMININE SINGULAR (ها)]

As an example, the Arabic word (و سيكتبونها) If it divided into clips, it would be like that (w + s + y - ktb - wn+ hA) which is meaning "and they will write it ". With two proclitics, the first one is ("و" spell as "W" and it means "and"). The second one is ("س" spell as "S" and it means "will"). A prefix which is ("ي"and spell as "Y" that

refers to the third person).A suffix which is ("ون" spell as "wn" and refers to the masculine plural), and a pronominal enclitic which is ("ها" spell as "ha" that refers to it or her)[10] (W. Salloum & Habash, 2014).

This word (و سيكتبونها) is equivalent to an entire sentence in English, consisting of conjunction, then a future particle, followed by an inflectional prefix, and the verb stem, then, an inflectional suffix, and finally, a pronominal object.

The separation- between segments are highlighted by + signs. As can be seen from this example, three of the parts, namely, the conjunction, the future particle, and the object pronoun, are independent as far as syntax goes, as the case with the stem ktb (to write). However,they cannot exist on their own and must be part of the written form of the word, So, orthographically speaking, i.e., they are clitics (Kübler & Mohamed, 2012).

When dealing with POS tagging in Arabic, if it is to be done automatically, one has two choices: either one can assign specific tags to complete words, or one may split the word into individual units, or segments, then assign POS tags to those individual segments. Previous work (AlGahtani, Black, & McNaught, 2009; Diab, Hacioglu, & Jurafsky, 2004; Kübler & Mohamed, 2012; Van den Bosch, Marsi, & Soudi, 2007) have opted for the segmentation method. Such work focused on POS segment tagging with the help of the Penn Arabic Treebank (ATB) (Maamouri & Bies, 2004)

Habash (2010) said, “Morphology is the study of internal word structure," a point to consider with NLP tasks. This project will focus on functional morphology in Arabic the function of a unit or segment within a word. Discussing functional morphology, two primary operations will be came across: cliticization and inflectional morphology. According to Anderson, the term, the clitic is the novel form of generalization of the more traditional categories of enclitic and proclitic. It comes at the start or end of a host or base word. Clitics include Determiner DET, and Pronominal PRO, conjunctions CNJ, and questions QST as well as particle proclitic PRT (see Table 2.1 examples in Arabic) (Anderson,2010). Attached to a base word, clitics take the following position:

[QST+ [CNJ+ [PRT+ [DET+ BASE +PRO]]]]

In Arabic, the pronominal enclitic could be joined to nouns as per possessives, or verbs and prepositions functioning as objects. However, in the case of the determiner, it can be attached to nouns only. Therefore, clitics cannot stand alone as words. The following type is a conjunction, which can occupy any position in a word. The first letter in the Arabic alphabet, and it comes at the beginning of the word, functioning as an interrogative sign. Lastly, the future particle is joined to verbs only. Question enclitics come at the beginning of the sentence and it, working as an interrogative sign. The Arabic equivalent of letter A is the first letter in the Arabic alphabet. Lastly, the future particle is joined to verbs only (Habash, 2010).Fig 2.1 show example of cliticization:

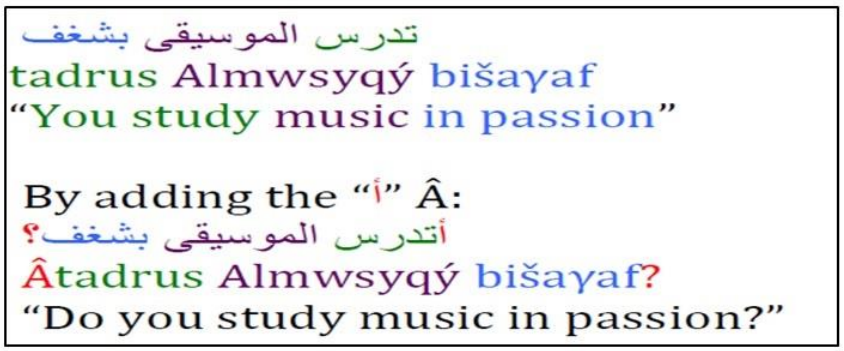


Figure 2. 1: Example of cliticization (Habash, 2010)

Clitics	Class	PATB POS Tag	Function	English
+أ Â+	QST	INTERROG_PART	interrogative همزة الإستفهام	*yes/no question*
+و *wa*+	CNJ	CONJ	coordination واو العطف	and
			connection واو الربط	and
		SUB_CONJ	circumstantial واو الحال	while
	PRT	PREP	oath واو القسم	by
			accompaniment واو المعية	with
+ف *fa*+	CNJ	CONJ	conjunction فاء العطف	and, so
		CONNEC_PART	connection فاء الربط	and, so
		RC_PART	response conditional فاء الجزاء	so, then
		SUB_CONJ	subordinating conjunction فاء السببية	so that
+ب *bi*+	PRT	PREP	preposition حرف جر	by, with, in
+ك *ka*+	PRT	PREP	preposition حرف جر	such as, like
+ل *li*+	PRT	PREP	preposition حرف جر	to, for
+ل *la*+	PRT	EMPHATIC_PART	emphasis لام التوكيد	will certainly
		RC_PART	response conditional اسمها	so, then
+س *sa*+	PRT	FUT_PART	future particle سين المستقبل	will
+ال *Al*+	DET	DET	definite article ال التعريف	the

Table 2. 1: Examples of proclitic and enclitic (Habash, 2010)

Moreover, there are negative particles in Arabic, amongst which are "ما" ma and "لا" "la". There is also the vocative particle "يا" "yA," used when calling out to someone, and finally, we have the interrogative particle "ما" "mA" which is the equivalent of "what". Furthermore, some Arabic dialects have additional clitics.

The second operation by which functional morphology takes place is inflection. The form of the word changes according to several aspects, among which are voice, subject (which differs with gender and number), and finally, mood. This part will focus on changes in the subject form concerning the person, gender and number (Habash, 2010).

The verbal subject in Arabic has the following features:

a. The person doing the action may be either: 1st"متكلم" "mutakalim" (speaker), 2nd" مخاطب" "mukhATab" (addressee) and 3rd"غائب" "γAŶb"(absent).
b. Gender differs: it is either masculine "مذكر" "muðkar" or feminine "مؤنث""mŴnath".

c. A number of the person may be singular " مفرد "mufrad", dual "مثنى" "muthnÝ" or plural "جمع""jama".

According to Habash (2010), “verbal subject is indicated through affixations, whose form is constrained at the verbal aspect". That is to say, the form of the verb changes by adding suffixes in the perfective aspect “ماضي” “mADy”, and circumfixes for imperfective ”مضارع “muDAriç” and imperative “ أمر Amr”.

Many language researchers have categorized English texts and much more such as French, Spanish, Japanese, and Chinese. Nevertheless, on the other side, it can be found that the researchers classifying texts for the Arabic language is somewhat limited.

The issue at the classification of a text that consists of many different sub problems that have been deliberated extensively through a lot of studies. Also, many different techniques were used for the classification of text like; Neural Networks (NN), Support Vector Machines (SVMs), Naïve Bayes (NB), K Nearest Neighbor (KNN), Decision Trees (DT), N-Grams, Association Rules and Maximum Entropy (ME) (Ayadi, Maraoui, & Zrigui, 2016).

After an extremist has been identified within social media and the concepts of Arabic language processing has been displayed. The tools will be displayed that the machine would use on these outputs using Data Mining to detect the extremists using Machine learning science and the Artificial Intelligence algorithms.

2.6 Data Mining

Data Mining is the operation of arranging vast amounts of data to determine patterns and build problem-solving relationships via analysis of the given data. Data Mining creates association rules by studying the data for repetitive patterns. After that, by utilizing support and confidence standards, almost all central relationships through the data are determined. Support indicates how often items be present in the database, and confidence indicates how often the data is precise (Rouse, 2019).

Data mining consists of different criteria like path analysis, classification, aggregation, and prediction. Path analysis or so-called sorting criteria find patterns in which behaviors are interdependent. A sequence is a list of items grouped into groups in a specific order. The classification is the operation of searching for new patterns and

can rearrange the data. The classification algorithm uses the factors in the database to extract variables figure 2.2 (Rouse, 2019).clustering parameter detects a set of previously unknown facts, they visually record them. Clustering gathers a collection of objects and groups them depending on their resemblance (Rouse, 2019).

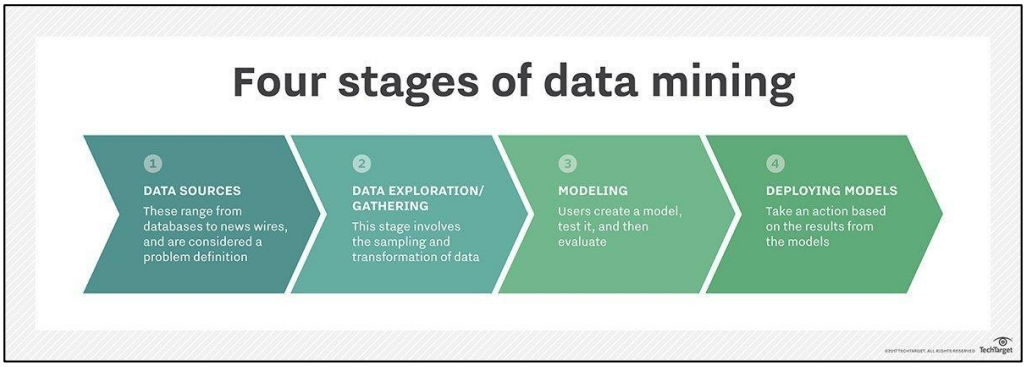


Figure 2. 2: Stages of data mining (Rouse, 2019)

2.7 Machine learning

Via machine learning, machines acquire knowledge of obtained data in a very professional way. When data is accessible and displayed, mainly with the availability of modern devices features that help to speed up the flow of information quickly, machine learning will be required. This flow illustrated in Fig 2.3.

Machine Learning (ML) is a class of algorithms that assists the programs in improving their performance to produce precise results without being programmed in advance. The prerequisite for ML is to create algorithms, which can utilize the received data by analyzing the data statistically to be able to produce renewable outcomes according to the novel inserted data (Burnes & Rouse, 2018).

ML algorithms are divided into two classes, which are supervised and unsupervised. The supervised algorithms perform by using input data analyzer through ML capabilities to achieve the desired results. Also, it gives feedback on the accuracy of the expected results at the stage of learning the algorithm. Data researchers identify variables or characteristics that need to be analyzed and used to improve forecasts. When the training phase ends, the algorithm implements the training on the new data (Burnes & Rouse, 2018).

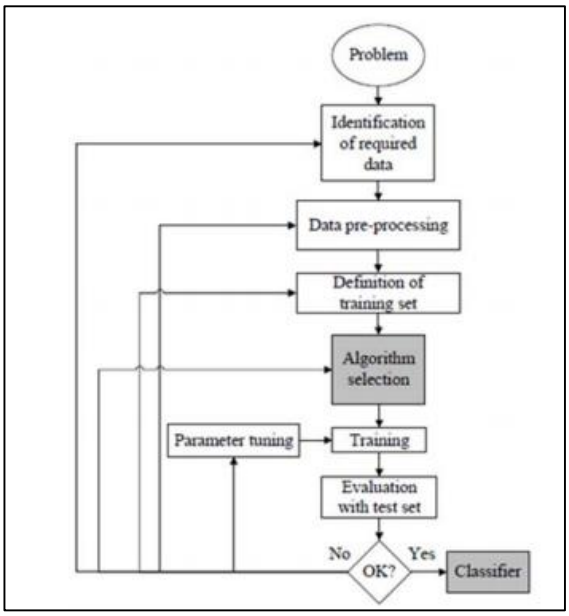


Figure 2. 3: Workflow of supervised machine learning algorithm (Dey, 2016)

The unsupervised algorithms Fig 2.4 should not be trained in the data of the desired results. Instead, they use a repetitive method called "deep learning" to analyze data and draw conclusions. Unsupervised learning algorithms are also known to neural networks, which are used to perform more complex processing than supervised learning systems like image identification, text-to-speech, and natural language creation.

Neural networks work by filtering enormous training data models, often finding small connections between multiple variables. After training, the algorithms can use their distribution bank to illustrate new data. These types of algorithms became available and possible in the era of big data because they needed a massive amount of training data(Burnes & Rouse, 2018).

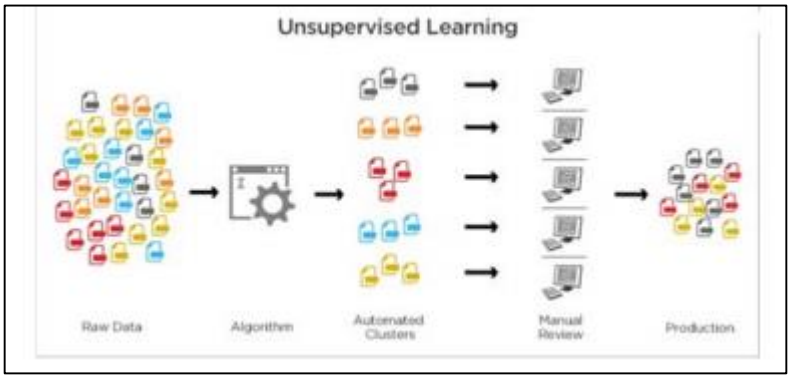


Figure 2. 4: Unsupervised Learning(Burnes & Rouse, 2018)

2.8 Classifier Modelling

In this thesis, a tackling process of the subjects of data mining and Natural Language Processing (NLP) will be performed. This will require the use of WEKA, a graphical user interface. WEKA is, in fact, a machine-learning environment, capable of applying classification algorithms for NLP and data mining tasks. The input data to WEKA should take the form of an ARFF file, the formats suitable for WEKA (Mavengere, 2013).

WEKA gives different types of classification algorithms, each with its own four kinds of experiments, namely, Percentage Split, Cross-Validation, as well as Use Training Set, in addition to Supplied Test Set. The first experiment performs the function of splitting input information into categories for training and testing data.

The algorithms employed here are J48 decision tree algorithm, together with Naïve Bayes Simple algorithm, KNN, SVM, Random Forest and classification by Regression. Text classification algorithms are carried out with a supervised learning approach. This is done to train the classifier, which means algorithms deduce and learn classification patterns from reading labeled data.

- ***Decision tree algorithm (J48)***

J48 is a decision tree for classification. Each node in the said tree depicts an attribute of the input data. A tree is built to act as a model for the classification process. Each tuple of the input data is included within the construction of the tree , and the result is the classification of that tuple (Dunham, 2006). Let us not forget that the input data is actually the outcome of our experiments.

The Univariate Decision Tree approach is implemented in this project, using the J48 algorithm. The Multivariate approach is presented as the Linear Machine method; both require using the Absolute Error Correction, as well as the Thermal Perceptron Rules. Decision Tree is a well-functioning technique for supervised classification, mainly when the results are examined manually. Multivariate Decision

Tree employs the idea of attributes correlation and yields the best performance for conditional tests, by comparison to Univariate approach (Kaur & Chhabra, 2014).

- ***Naïve Bayes classifier***

The naïve Bayes algorithm can be described as a probabilistic classifier; it is rather simple, hence the term naïve and it gives several of probabilities by keeping count of the frequency as well as the combinations of values in a given body of data. The algorithm employs Bayes theorem and works under the assumption that every attribute of a given object is considered independent of the value of any other characteristic of the said object. This independence assumption has its shortcomings, since, in the real world, attributes are not always independent. Nonetheless, the algorithm tends to function well and learn quickly in supervised classification problems (Patil & Sherekar, 2013).

In other words, it is a classification method based on Bayes' Theorem, operating on the principle of independence among features to be classified. A Naive Bayes classifier assumes that any feature of an object is unrelated in value to any other feature (Schutt & O'Neil, 2013).

Therefore, it can be said that although naïve Bayes is rather simple, it is suitable for sizable amounts of data. In comparison to other algorithms, it has a much higher success rate (Schutt & O'Neil, 2013).

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

Another classification algorithm is referred to as IBk or Instance-Bases learning with parameter (K). It can be also referred to this algorithm as Knn and; it works on the assumption that all features are of the same importance. In case the dataset is noisy, then by chance, it may be found a wrongly classified training instance to be the closest one to the test instance (Kirchner, 2010).

So, if one has two values of K where k=1 and k=3, then the value of k, would be considered the set of neighbors to query to come up with a prediction.

- ***Support Vector Machines (SVM)***

Support vector machines are built on the Structural Risk Minimization concept from computational learning theory. The idea behind the structural risk minimizations are to come up with a hypothesis h for which one can ensure the least confirmed error. The true or confirmed error of H is actually the probability that h may cause an error on a test example that must be unseen before and, at the same time, randomly selected.

A higher threshold can be used to link the true error of Hand the error of the same hypothesis h on the test set. And the complexity of H (calculated by VC-Dimension), as well as the hypothesis space, including H. Support vector machines found that the hypothesis H (approximately) reduces this threshold as far as the true error is concerned by efficiently taking control of the VC-Dimension of H.

SVMs are universal learners. SVMs can learn a linear threshold function. However, by a simple "plug-in" of a suitable kernel function, they can be employed to acquire knowledge of polynomial classier. They can also be used in the domain of basic radial function (REF) networks, as well as three-layer sigmoid neural nets.

One unique property of SVMS is that they can learn in ways that are being independent of the dimensions of the feature space. SVMs examine how complex hypotheses are based on the margin used to split the data, not the specific number of attributes or features. Thus, one can generalize, even in the presence of an abundance of features, as long as the data can be categorized with a wide margin, using functions from the hypothesis's environment.

The same margin concept brings to mind a heuristic approach for choosing reliable parameter settings for any new learner (like the kernel width in an RBF network). The most efficient parameter setting is the one that generates the hypothesis with the least VC—Dimension. This permits fully automated parameter tuning without the costly cross-validation (Joachims, 1998).

- ***Classification by regression***

Regression: it is to find out the value for any given input by using a minimal approximation method. The result is to come up with a mathematical rendering for a data sample.

One may also define it as the process of examining the relationship (Interpolation) and the impact of this relationship on future values outcomes (Multidimensional). Regression helps to identify how a variable behaves when other variables are altered in the process. That is why Regression analysis is of use in the domain of prediction and forecasting applications.

To be brief, when items areas signed to different categories, one may use classification algorithms; however, when it comes to predicting future values, then regression algorithms are the way to go (Ceci, Hollmén, Todorovski, Vens, & Džeroski, 2017).

- ***Multivariate Adaptive Regression Splines (MARS)***

For several classifications, one uses several or multi regression. Machine learning performs linear regression; therefore, the output was set to 1for the instances that are linked to the class and 0 for others that do not belong. Then, the class was chosen to have the highest output. This technique extends linear regression to a process of classification by regression.

Classification: is to identify the class or category in which a given output fits. What we have tried to do is to use a regression technique to classify a given set of data.

In cases where the class values are depicted as {0,1}, the process of classification depends on a selected threshold value, and consequently, all readings that come below are predicted as (0). Any text above this threshold value is predicted as (1).

Two class problem:

1. Training: the classes are called 0 and 1.
2. Prediction: a threshold was set for predicting class 0 or 1.

a. Multiclass problem: “multi-response linear regression”:

1. Training: perform regression for each class, where the output was set to 1 for training instances belonging to the class and set 0 for cases that do not belong.
2. Prediction: select the class with the highest output or use “pairwise linear regression," which carries out a regression for every pair of classes (Hall et al, 2009)
3. Try to add a new attribute (classification) wherein to store the regression result.

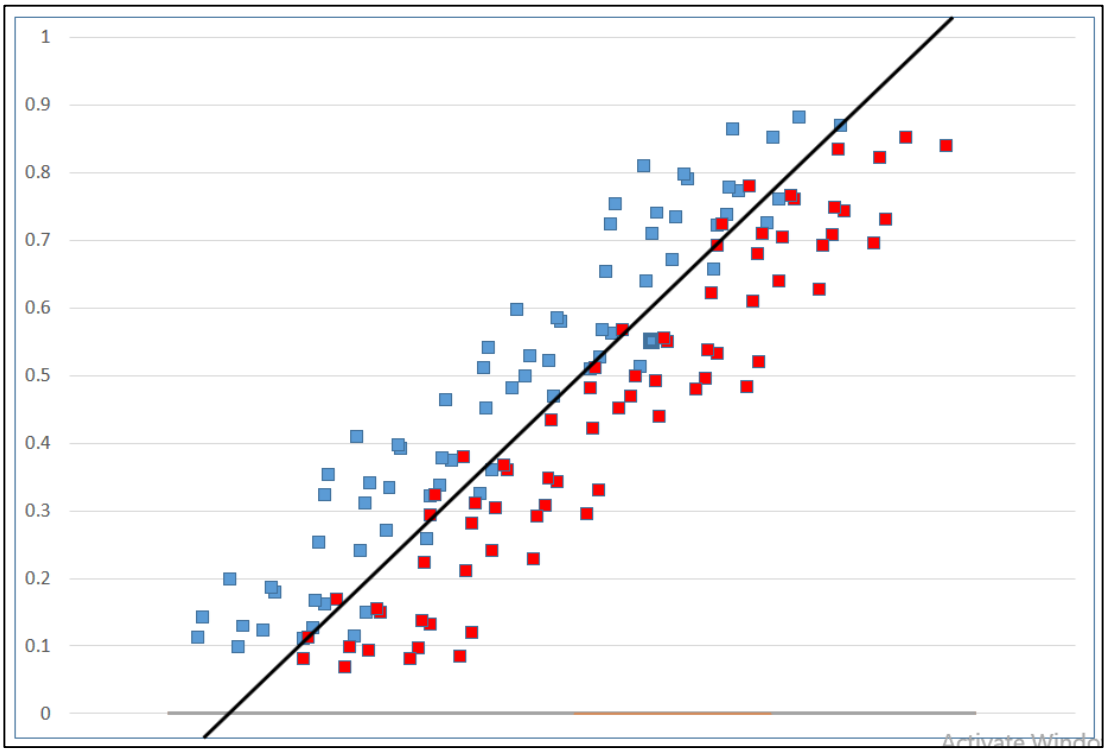


Figure 2. 5: Linear Regression process (Hall et al., 2009)

First, we must make sure that the class attribute is converted to binary 0,1. After that, we apply Linear Regression. Following the process of Linear Regression, the prediction values are now to be stored in the new attribute classification. Then, the type of class is turned back to nominal.

The algorithm is going to use the two values response from regression process classification , and the binary class attribute {0, 1} is used by One R to optimize the split point for the two classes (Hall et al., 2009).

- ***One R Algorithm***

Another classification algorithm is One R, which is short for "One Rule"; it is a simple but precise classification algorithm that produces one rule for each predictor, and then chooses the rule with the least total error as its "one rule". To generate a rule for a predictor, one builds a frequency table for each predictor against the target. it found that One R produces rules that are not much less accurate than top-notch classification algorithms, while still being easy enough for humans to interpret (Soman & Bobbie, 2005).

How does it work?

- For every value of the predictor, this research follow this following rule (Sayad, 2017):
    a. Firstly, examine how often each value of a target (class) comes up and make a count of it, then identify the most frequent class.
    b. Then, make the rule to assign that particular class to this value of the predictor.
    c. Next, calculate the total error of the rules of each predictor.
    d. Then, select the predictor with the least total error.

This is the method of applying a regression technique for classification. This algorithm helps to enhance the performance in the case of binary classification; however, in the case of multi-dimensional classification, one must opt for a multi-response linear regression.

Multiple linear regressions (MLR): is defined as a method used to model the linear relationship between a dependent variable (target) and one or more independent variables (predictors).

$$\text{observed data} \rightarrow \quad y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p + \varepsilon$$

$$\text{predicted data} \rightarrow \quad y' = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$$

$$\text{error} \rightarrow \quad \varepsilon = y - y'$$

Figure 2. 6: multiple linear regressions (Sayad, 2017)

MLR Figure 2.6 is built according to the principle of ordinary least squares (OLS); the model is made by minimizing the sum of squares of differences of two values, namely, the observed and predicted ones. Minimize $\sum (y - y`)^2$.

The MLR model rests on several assumptions, among which, for example, that errors are distributed with zero mean as well as constant variance. As long as the assumptions are fulfilled, the regression estimators are deemed optimal. That is to say, they are unbiased and efficient, as well as consistent. Unbiased implies that the expected value of the estimator is equivalent to the actual value of the parameter. Suitable means that the estimator has considerably less variance than other estimators do. As for consistency, it implies that the bias and variance about the estimator approaches zero, whereas the sample size approaches infinity (Sayad, 2017).

- ***Random Forest***

The combination of various learning models enhances the level of accuracy about classification, a process referred to as Bagging. Bagging is used as an averaging approach to noise, functioning with on-triggered models to make a model with low changeable values. This concept explains the idea of the Random Forest algorithm as a big group of de-correlated decision trees, utilized to make recognition patterns.

The term Forest was used here about many decision trees, which serve as tools to make the classification. Such is the utilization of the Bagging technique, in simple terms. Many different decision trees are made by randomly selecting various items.

The next point explains how Random Forest is built. Let us have a look at the data set to be classified below, as shown in Figure 2.8.

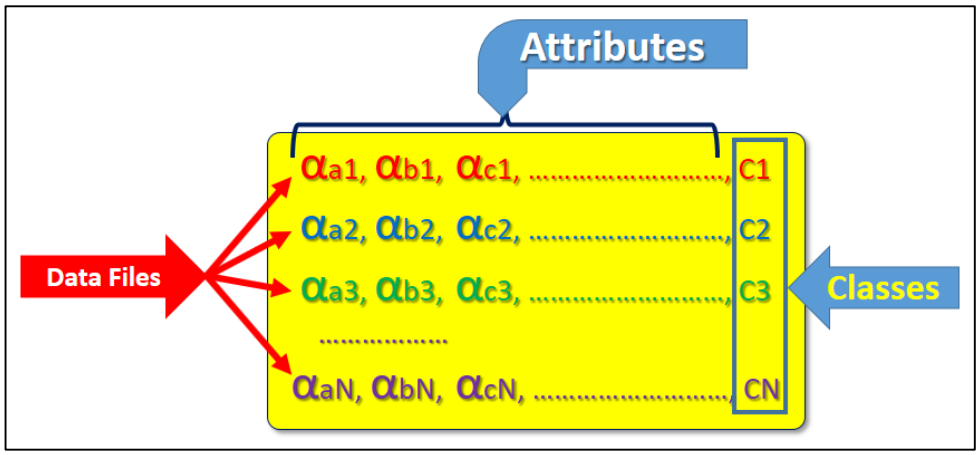


Figure 2. 7: Sample of data to be classified (Kaur & Chhabra, 2014)

Where:

αa1, αb1, αc1, ……, αcN are attribute values.

C1, C2, C3 …..... CN are class values.

1, 2, 3. ……...…….. N are the sample file number.

Random Forest algorithm randomly selects a number of data files to create random subsets, as shown in figure 2.8.

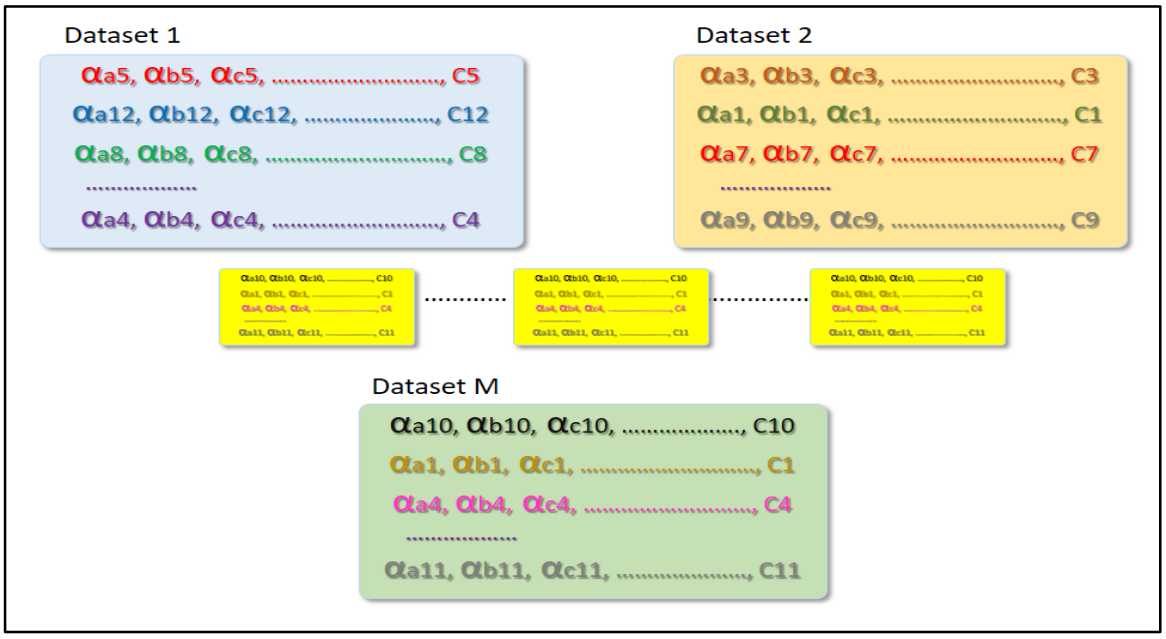


Figure 2. 8: random selection of data files (Kaur & Chhabra, 2014).

Every subset of datasets, created at random is in fact used to build a different decision tree. The group of decision trees makes up the Random Forest as shown in Figure 2.9.

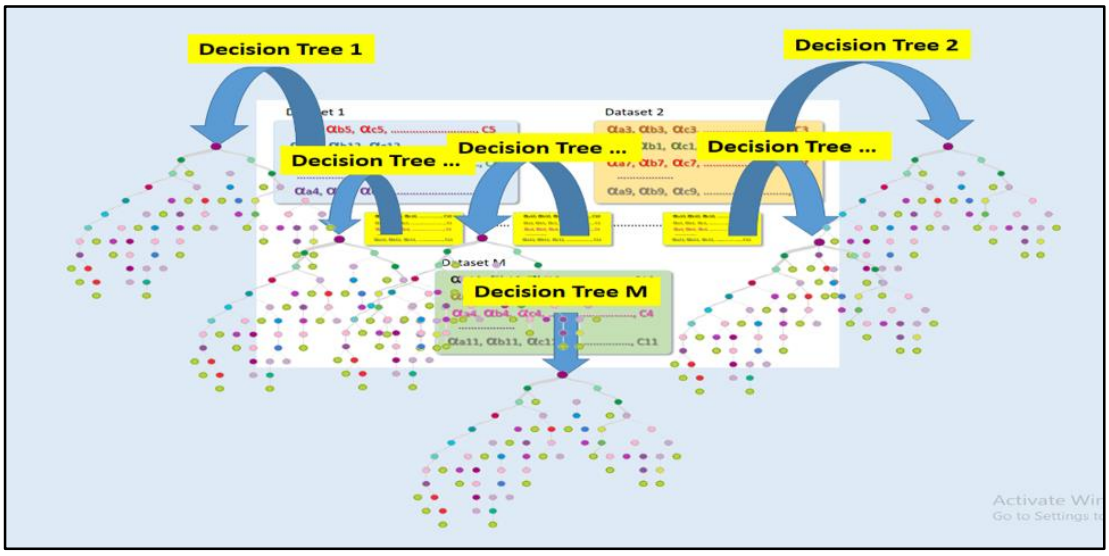


Figure 2. 9: decision tree created (Kaur & Chhabra, 2014)

Decision tree created for every randomly selected subset of data. The Random Forest concept relies on two main beliefs:

a. Most of the decision trees yield correct predictions.
b. The decision trees make errors at different points.

The algorithm makes use of all the decision trees to come up with a ranking approach for all classifiers. This ranking is carried out by applying every decision tree to predict the test file. Figure 2.10 demonstrates an example of ranking classifiers.

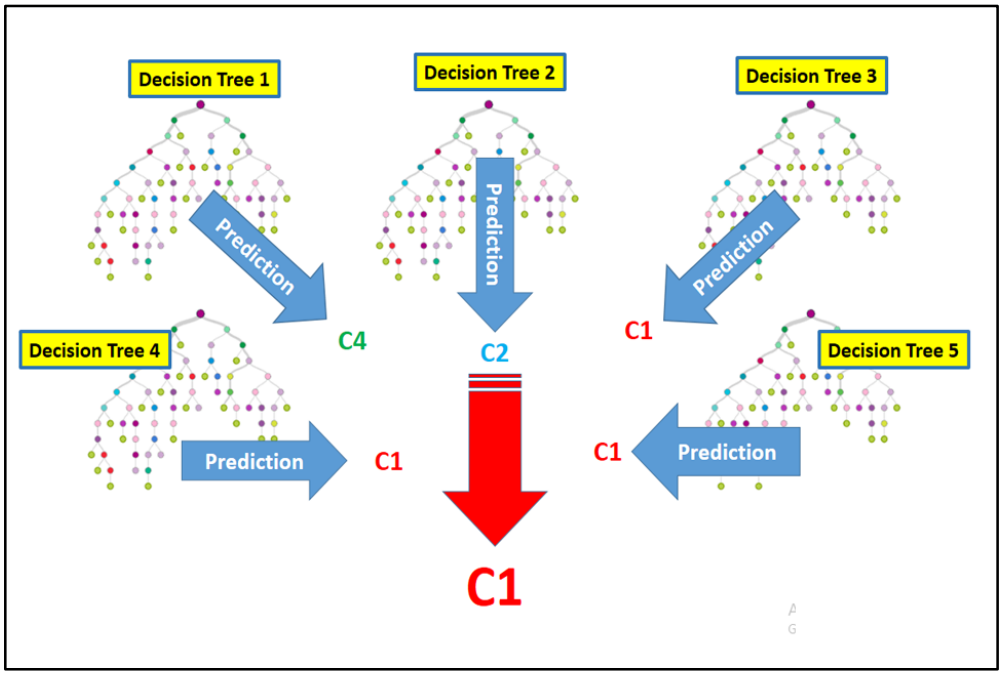


Figure 2. 10: Classifier ranking process (Kaur & Chhabra, 2014)

One can sum up the Classification process as its algorithm goal to sort and group records into classes. To put it differently, it is actually predicting.

2.9 Cross Validation

The results of the classifier are summed up in 3 parameters, derived from the confusion matrix, and they are:

a. Precision
b. Recall
c. F-measure

The definition of a Confusion matrix is the result of performing classification. The confusion matrix depicts the 4 probabilities:

a- The Class is True:
   i. Prediction is Positive (TP)
   ii. Prediction is Negative (TN)
b- The result is False:
   i. Prediction is Positive (FP)
   ii. Prediction is Negative (FN)

		Prediction	
		Positive	**Negative**
Class	**True**	TP	TN
	False	FP	FN

Table 2. 2: Cross validation

Table 2.2 is confusion matrix 2x2, showing how the four results from the classification process are derived from it.

i. Accuracy: may be defined as the correct prediction divided by the total predictions.

$$\textbf{accuracy} = \frac{\textbf{correct predictions}}{\textbf{total predictions}} = \frac{\textbf{TP} + \textbf{TN}}{\textbf{TP} + \textbf{TN} + \textbf{FP} + \textbf{FN}}$$

ii. Precision: one can calculate precision through this ratio and it shows how many events of the positive class were in fact positive.

$$\textbf{precision} = \frac{\textbf{TruePositive}}{\textbf{totalPositive}} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}}$$

iii. Recall: this ratio calculates the positive events that have been predicted correctly as positive:

$$\text{Recall} = \frac{\textbf{True Positive}}{\textbf{True Positive} + \textbf{False Negative}} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

iv. F-measure: known as F1, expressed as the weighted average of the precision and recall.

$$\textbf{F\_measure} = \frac{2 \cdot \textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

2.10 SAFAR Framework

SAFAR is a platform dedicated to Arabic natural language processing (ANLP). It is an open source platform, and it follows the basic systems and modules, to provide an integrated development environment (IDE), which includes the resources needed

for different ANLP treatments, basic levels of the language, especially Arabic (such as formation, grammar and semantics), and ANLP applications as well.

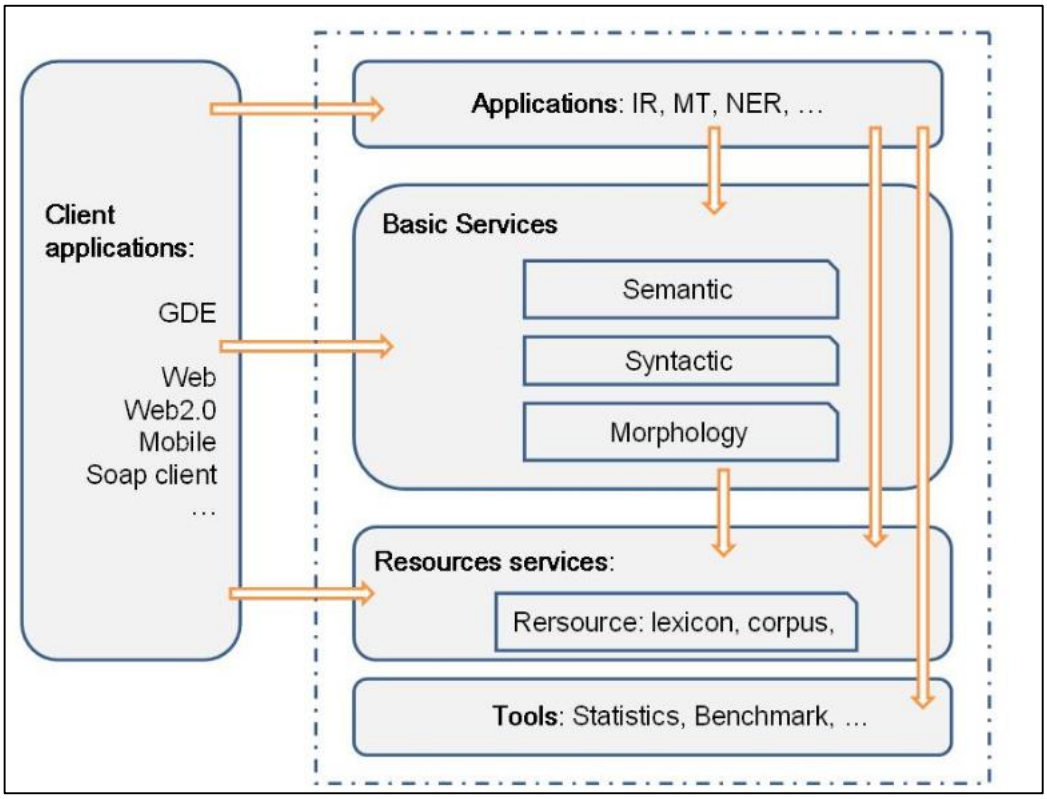


Figure 2. 11: SAFAR architecture (SAFAR, 2013)

As shown in Figure 2.11 the construction of the Safar tool consists of several layers each layer developed as a set of interfaces using Java reusable application programming, and included tools, which consist of a set of technical services (statistical functions, division of sentences, test tools , Etc.), and also contained resource services, which provide linguistic advice to resources such as dictionaries. And NLP services, which contain several layers which are morphology, semantics, and syntax. As for the applications, they contain high-level applications and finally the blind, which are used in the direct use of the services layer (SAFAR, 2013).

2.11 Posit Framework

According to Weir (2007), the Posit Textual Analysis Toolset generates 27 features from any input text, including frequency data, in addition to Part-of-Speech (POS) tagging. Posit contains these profilers:

POS Profiler: Performs analysis of a given text corpus to derive statistics on the POS characteristics of that Text, among which are types/tokens, number of sentences, average sentence, word length, common nouns, and proper nouns, verb tenses and total frequency of each POS token or type.

Vocabulary Profiler: Based upon the statistical data output by the POS Profiler, the Vocabulary Profiler can determine the relative frequency of occurrence for vocabulary items in the selected corpus. These frequency data may be compared to a reference set of frequency data (derived from the British National Corpus) to pinpoint unusual word occurrences or individual terms, the use of which is likely to prove unfamiliar to English readers. The module is also able to determine n-gram frequencies, which can compare word and n-gram frequencies of a text to a reference frequency list.

Readability Profiler: This software component, (under development) will focus on text readability, based upon the statistical analyses from the POS Profiler and the frequency data from the vocabulary profiler. Word commonality (Weir & Ritchie, 2006) and average collocation frequency (Weir & Anagnostou, 2007).

1. **Posit text profiling tools as applied on English text**

    i. The main Posit analysis gives a detailed analysis of data on three levels:
        a. Summary analysis,
        b. Aggregated analysis, and
        c. Individual post-level analysis.

    ii. Posit actions are invoked at the command line, using one or more of the following operations:
        a. pos_all.sh – invokes the main Posit analysis and generates most statistics on target text;
        b. ngram.sh – invokes a n-gram frequency analysis (for n=1 to 4) on target text;
        c. pos_ngram.sh - invokes a part-of-speech n-gram frequency analysis (for n=1 to 4) on target text;

d. Results – a comprehensive set of analysis results is output to a ‘results’ folder for any target text.

The Posit software is designed to operate under Unix. Posit is designed to generate quantitative analyses of text. To this end, three levels of detail are output for any analyzed text sample. These are (i) summary, (ii) aggregated and (iii) pos-level. The summary level gives an overview of detail. This includes the total number of word types and tokens, type-token ratio, number of sentences, average sentence length, number of characters and average word length. Counts of ten separate parts-of-speech are listed for tokens and types. The summary file contains 27 features which used by the classifier later in our approach (see Appendix A) to see an example summary output for the text file after posit processing.

At the aggregate level of detail, Posit provides counts of ten aggregated parts-of-speech. These are listed by specific part-of-speech. Finally, at the part-of-speech level of detail, Posit lists the frequency of individual word occurrence within its part-of-speech, for every word in the original text.

Posit is a Unix command-line system that comprises a series of individual programs. These include the main analysis facility (pos_all.sh), an n-gram analysis program (ngram.sh) and a part-of-speech n-gram analysis program (pos_ngram.sh).

2.12 Literature Review

Once you have finished reading the first and second chapters, you will have a sufficient perception of the research. You will also have a slight idea of all basic terms in this research; beginning with extremism and means used by people in this area. This Chapter will talk about some of the approaches through which the Arabic texts were classified, which is the basic idea of this research. It will talk about the most relevant of them and its results.

2.12.1 Sentiment Analysis of Arabic Text (Opinion Mining)

Aldayel and Azmi (2016) carried out a study on sentiment analysis that connected various domains of study such as NLP, computational linguistics, and text mining (Aldayel & Azmi, 2016). It concerns the extraction of the given information from textual data. It may be called sentiment analysis or opinion mining (Pang & Lee, 2008) they used the Twitter API to collect Twitter data from a specific domain in a specific language. Preprocessing was done by the removal of irrelevant information, tweet cleaning and other preprocessing techniques. The classification technique is based on a Lexicon-based classifier. To extract features used in the classification process, they used the term frequency inverse document frequency (TF-IDF) weighting scheme on the n-grams (1-2-3 gram) and selected the features that have frequencies greater than a certain threshold.

They used two measures to evaluate the classification process:

- The error rate (percentage of misclassification twists).
- Accuracy Rate (percentage of correctly classified twists).

The Twitter API for Arabic data collection was used. The data was then passed through data cleaning and attribute extraction using 1-2-gram statistical processing. This is to prepare the data to obtain the feature vector for the main purpose of research, i.e., classification. The machine learning classifiers used are Naive Bayes (NB), and Support Vector Machines (SVM). They apply both classifiers twice. First, they apply both classifiers on features extracted based on unigrams. Then, use the features extracted based on bigram statistics (Amira Shoukry; Ahmed Rafea, 2012).

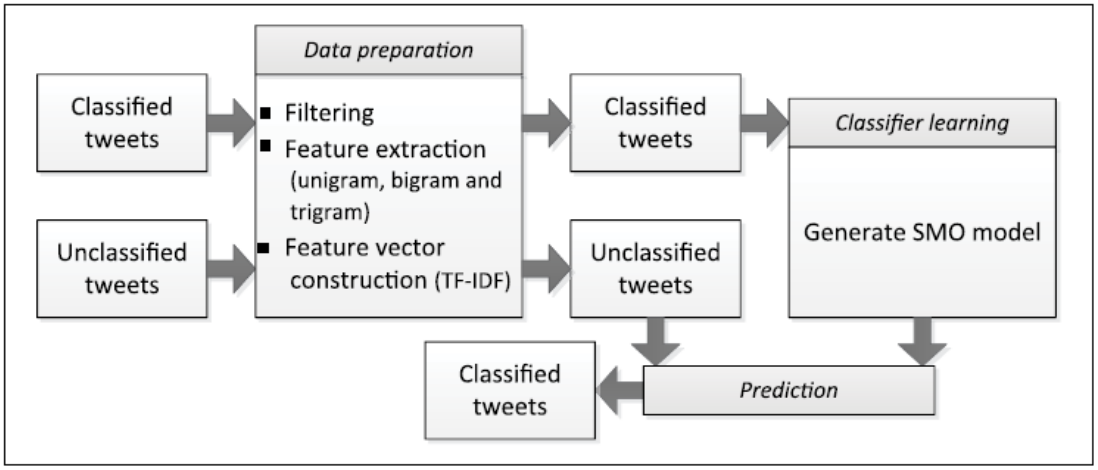


Figure 2. 12: Twitter classification block diagram (Amira Shoukry; Ahmed Rafea, 2012)

The SVM classifier was employed as the research classifier and the data collection used the Twitter API. Data cleaning and normalizing, with stemming, and stop words removed was applied to make data suitable for feature extraction. The data sets were organized using 1- Unigrams, 2- Bigrams + Unigrams and 3- Unigrams + Bigrams + Trigrams (Shoukry A and Rafea A., 2012).

The SVM classifier was applied before after applying each stage of the preprocessing to test its effect on the system's performance. Sentiment analysis studies vary in pre-treatment techniques, analysis methods, and review design. Some have used the supervised method, others the unsupervised learning method. A multi-level technique based on semantic orientation (lexical classifier to handling unnamed tweets) and ML (SVM classifier) was suggested by Aldayel and Azmi (2016) to identify the polarity of Arabic tweets. The biggest challenge of this mixed approach, however, is to deal with the application of Twitter in dialectical Arabic.

Moraes, Valiati and Neto (2013) compared the execution of SVM (support vector machines) and NN (neural networks) at document-level sentimental Arabic analysis. They have found that NN execution is better than SVM on the same records (Moraes, Valiati, & Neto, 2013).

Li and Li (2013) have gauged the objectivity and the truthfulness by utilizing SVM as a method (Li & Li, 2013). Cherif, Madani and Kissi (2015) worked on the execution of three famous techniques (bagging, boosting and random subspace). This

was instituted on five algorithms, which are (Naive Bayes, Maximum Entropy, Decision Tree, K Nearest Neighbor, and Support Vector Machines) for sentiment categorization. The results showed that the random subspace was more accurate (Cherif, Madani, & Kissi, 2015). Table 2.3 contains some of these studies.

Study	Problem	Methodology	Results
(Aldayel & Azmi, 2016)	Recognizing the polarity of Arabic tweets and the practice of tweeting in dialectical Arabic	Hybrid classifier, Lexical-based classifier, Feature extraction, and Support Vector Machines classifier	The general quality of the obtained results in this study from hybrid classifier quantified by F-measure is 84%, and accuracy is 84.01%
(Cherif et al., 2015)	Evaluating the influence of Arabic grammatical richness on opinion mining accuracy, building a new accurate statistical approach that supports the Arabic language syntactic and grammatical complexity, and analyzing Arabic reviews and comments more accurately	Sentiment classification, Support Vector Machines, a dataset composed of 625 Arabic reviews and opinions of the public obtained from the official website of Trip Advisor	Results obtained were rooted in Support Vector Machines depicted that this method dramatically affects the identification of opinions.
(Duwairi & Qarqaz, 2014)	The effects of stemming feature correlation and n-gram models for Arabic text on sentiment analysis	Support Vector Machines, Naive Bayes, and K-nearest neighbor classifiers	The results of the experiments suggested that choosing the method of preprocessing on the reviews will enhance the performance of the classifiers

Table 2. 3: Research studies conducted on Arabic text mining in terms of sentiment analysis

2.12.2 Classification and Comparing Algorithms on Arabic Text

El Kourdi Bensaid and Rachidi (2004) categorized Arabic documents on the internet automatically by using an NB classifier with ML algorithms to classify soundless Arabic documents to one of five pre-determined classes. The results of the experiments confirmed the effectiveness of the NB classifier. El Koudri utilized groups of 1500 documents under five categories each with 300 text documents. Through 2000 expressions and roots, the precision of the classification varies in-between categories with an average precision overall for the classifiers of 68.78 %. Moreover, the highest performance of categories in these experiments reached 92.8% (El Kourdi, Bensaid, & Rachidi, 2004).

KNN algorithm (K-Nearest Neighbor) is one of the best classifiers for categorizing text documents in English with the SVMs algorithm. This was used by Al-Shalabi Kanaan and Gharaibeh (2006) on Arabic language for texts classification. They utilized the DF (Document Frequency) technique to extract the main words and minimize dimensions. The results proved that the KNN is suitable to categorize Arabic documents (Al-Shalabi, Kanaan, & Gharaibeh, 2006).

Maximum Entropy (ME) was applied by El-Halees (2015) and Sawaf, Zaplo and Ney (2001) to categorize Arabic news articles. El-Halees pre-processes data, utilizing natural language processing methods such as tokenizing, stemming, and part of speech then uses the maximum entropy method to categorize Arabic documents. The best-reported accuracy was 80.41% and 62.7% when using statistical methods by Sawaf without morphological analysis (El-Halees, 2015; Sawaf, Zaplo, & Ney, 2001).

Al-Zoghby, Eldin, Ismail and Hamza (2007) proposed a novel system that was developed to determine association rules using similarity measurements based on the derivation of the Arabic language. It also offered the advantage of using the "Frequent Closed Item sets" (FCI) concept when extracting the association rules instead of "Frequent Item sets" (FI) (Al-Zoghby, Eldin, Ismail, & Hamza, 2007).

2.13 Summary

In this chapter, many concepts and related terms were presented, which give the reader a complete understanding of the most ambiguous terms in this research. It also talked about Arabic language – the main topic of the research – and highlighted many of the most important strengths and difficulties in it. It talked about the automatic classification of texts and the most famous models used in this regard. The next chapter will talk about (literature review) related to some of the curricula that have addressed Arabic language texts processing, how it classified these texts and the approach used by these studies and the results obtained by the researchers. So in our Background and Literature Review chapters, the following topics have been reviewed:

- The difficulty in differentiating the use of each word, sentence and idea of the author in the Arabic language.

- The optimum algorithms for text mining in the Arabic language to automate the process of extremist website recognition.
- The different classification algorithms used in text mining that have different treatment approaches.
- The confusion matrix and how it helps to build classification process measures and metrics.
- The decision tree and how to create a forest of decision trees that have randomly selected know to form Random Forest.
- The quantitative Analysis, for text mining to obtain deferent unbiased, objective datasets to use with classification algorithms to differentiate pro-terrorism sites. Many studies have been discussed in this research; all of them are interested in the classification of Arabic texts. The next chapter of this study will be about the methodology used in this research.

CHAPTER THREE: METHODOLOGY

3.1 Overview

This chapter introduces the methodology proposed in this thesis, containing, research design, methods, tools, data preparation, compiling and building of the Arabic corpus, and covering the classification methodology. This chapter details the workflow process of our research. The proposed system was accomplished through several different stages. The first of these stages was data collection; the data were preliminary data, due to the lack of previous studies with the same type of current study.

At the start of the process for addressing the objectives of this research, data were collected from various sources and classified manually, with the aid of specialists. Insight on this manual classification and the specialists' role is provided in the data collection section. The second stage was creating the Arabic Corpus and data processing of its content through the Posit and SAFAR frameworks. The third stage was training the system classifier (WEKA API) on a set of text files pre-classified manually by specialists in the data collection stage. The last stage was to confirm the validation of the system by testing it against a separate test dataset. Figure 3.1 shows the four methodological steps in this process.

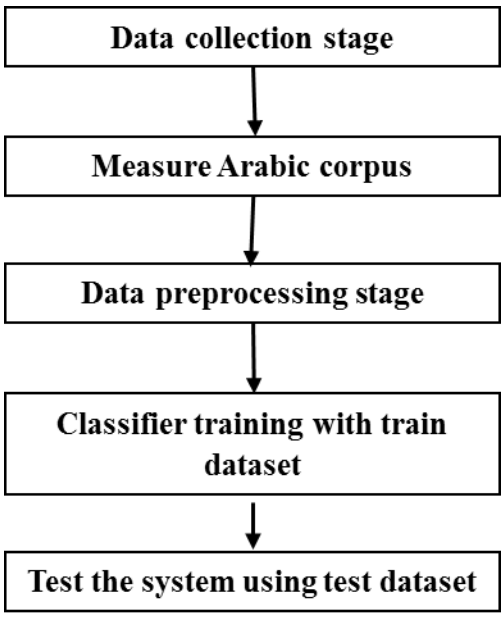


Figure 3. 1: Proposed methodological design

3.2 Methodological Approach

In every research, the methodological approach explains the nature of data being used by the researcher. The most commonly used methodological approaches are categorized as a qualitative, quantitative or mixed approach. The qualitative methodological approach shows that the nature of the information collected for the study is fact-based or non-numeric. However, the quantitative methodological approach ensures that the data collected for the research is comprised of numbers and quantities, which can be statistically tested or calculated for gathering research findings (Kumar, 2019). In contrast, qualitative data is non-numeric; therefore, it is interpreted for obtaining research results. Thirdly, the mixed methodological approach refers to the usage of both qualitative as well as quantitative data for data analysis (Ledford & Gast, 2018). As far as the present research is concerned, we have selected a quantitative method that we considered suitable for numerical data analysis. The use of this approach seems to be justified because it features selection, categorizing, identifying classes and measuring. According to Fletcher (2017), using quantitative data in research can facilitate in generating authentic and most relevant research findings because it helps in statistically testing and estimating the most accurate results.

In this research, the selected methodological approach is suitable for categorizing different Arabic-speaking website pages with unscrupulous intentions and questionable language. The use of this approach is also considered to be effective because it can help in representing data and findings in the form of numbers and graphs which are easy to explain and interpret. For this research, we have selected this approach according to the need for analyzing text data, extracting parts-of-speech features - as an example, verb or noun counts in certain text - so that we can test or validate the experiments directed toward our classification objective.

Considering these facts, it can be assumed that the selected methodological approach is the most appropriate method for the current research. This methodological approach can be further justified based on the work completed by Wiek & Lang (2016)

who argued that the quantitative methodological approach could turn out to be effective for research because it focuses on objective measurement as well as statistical, numerical and mathematical analysis. The selection of this approach will ultimately allow us to present the statistical basis for our conclusions, along with graphical illustrations.

3.3 Data collection

Data collection and data preprocessing play a major role in research. Data is viewed as an important source of information and knowledge. It can form the basis of the research results and findings. The term data collection is used to define the process being used or followed by the researcher for aggregating and assembling desired information. This information can be used for generating research outcomes (Hofmann, 2013). In like manner, the term data pre-processing is used to describe a cleaning process which is considered an important step. In simple words, data preprocessing is a data refining technique that seeks to eliminate noise and obvious aberrations from the raw data.

3.3.1 Gathering data

The proposed system depends on the analysis of an Arabic dataset but a specific one; it must contain text data for encouraging extremism, anti-extremism and neutral data, in Arabic language. Since such an existing resource proved elusive, we had to develop our own means of gathering such a dataset (using tools like sketch engine). To this end, we used a 'seed list' of Arabic words and sentences in an input list box and the sketch engine will fetch around one million words per search. The data range was the most likely used words for extremism websites, tweets and any social media website, e.g., the Arabic equivalent of " kill the disbeliever and you enter heaven".

Through this process, we collected more than 7000 Arabic text files and processed them to form the downloaded corpus of individual files, in which everyone represents pro-extreme text, with associated id and URL. The same approach was followed for Anti-extremism and neutral data.

3.3.2 Verification

In order to verify our grouped collection, five people specializing in different jobs, checked the corpus files, and managed to manually categorize the text files into three categories (Pro, Anti and Neutral). This manual stage in the process took six months to complete.

To validate our original categories, we compared the results of the five specialists, and if at least four of them agreed to classify a specific file, this was included under the agreed category. In this way, we extracted about 5000 files that were manually reviewed for the three categories mentioned above, in order to build our own corpus specifically for the present research.

The process of collecting the content from different sites proved to be more difficult than we had expected. The first challenge was the lack of sites containing words belonging to one of the three categories (anti-neutral-pro). In other words, we did not find sites containing a simple set of words that carried a clear unified content. Almost all sites contain a mixture of different words, which are language combinations used to convey the concept that the writer seeks to communicate to the reader. We found that sites belonging to terrorist organizations use extremist words, and at the same time, words against terrorism to deceive sympathizers and to attract new members and followers. Most of the extreme sites use words to deceive the reader, giving the impression that they defend the truth and stand as a wall to protect society from the forces of darkness. These words that extremists use belong to the other party to inspire the recipient to adopt their corrupt thoughts.

On the other hand, writers who oppose extremist ideologies and terrorist groups have to use the rhetoric of extremism to inform the recipient of the extent to which extremist groups have been brainwashed and misled by those who do not know who they are. Even the neutral sites did not avoid terms that can be used by the advocates of extremism in their suspicious calls, as well as other terms used to combat extremist thought. All of the above shows the challenge faced when collecting files from different sites and the difficulty encountered in classifying these files manually. Moreover, it wasn't straightforward to reclassify a file from one category to another,

especially in light of the development of the algorithm used for the process of classification.

The second challenge was the rarity of sites that can openly be classified as inciting terrorism or calling for extreme ideas, because of governmental efforts to stop such sites, delete extremist content, and arrest the offenders. Also, it is improbable that these sites would declare themselves as extremist sites. In addition to that, most newspapers with extreme ideas that incite terrorism have been taken out of circulation or put out of business. Many suspicious accounts have been removed from social network sites, and writers who espouse these ideas have been banned from publishing their hate-spewing articles. This led to the scarcity of publications that openly incite terrorism in Arabic, and the difficulty of establishing a base for starting the research, and the need to increase our efforts to provide the necessary material to feed our classification process.

As a measure to reduce the problems we encountered when choosing search keywords to distinguish between different sites, we used a word sequence (N-Gram) instead of individual keywords. The result from data collection using this strategy is a large text file containing tens of thousands of words separated by HTML tags that contain the site URL.

3.4 Research design

The research design is explained as the methodological purpose behind carrying out research. The researcher must select a suitable research design because it can facilitate an effective data collection and analysis strategy. The most commonly used research designs are descriptive, exploratory or explanatory. The design of an investigation is selected based on several underlying factors including the research problem/question. In research, the descriptive design is used to simply describe a research problem without statistically testing any variable or relationship. On the other hand, the explanatory design of the research is used in cases when the researcher is testing a relationship between predefined variables. However, the exploratory design of the study is presumed to be experiment-based because it can help in generating new knowledge and information regarding unidentified variables (Nahar et al., 2019). The

research described in this thesis has opted for experimentations. The use of this design is suitable because it helps in generating authentic and relevant research data. The collection of authentic research information can affect the overall credibility and authenticity of obtained research results and findings. This can lead us to the quantitative approaches.

Quantitative research is the systematic experimental research of phenomena that are recorded and monitored through statistical, mathematical or mathematical techniques. Quantitative research is used to develop theories of phenomena and the mathematical relationships they represent. The measurement process is an important aspect of quantitative research to find the link between empirical observation and the mathematical relationship of the study (Given, 2008).

This quantitative research is used in the form of numbers that can be analyzed by the researcher using statistics to reach an unbiased result, and it can be generalized. While qualitative research produces only information about the cases studied only, and therefore is used to test the validity of general conclusions only (hypotheses). Therefore, we find that qualitative research studies the depth of specific experiences, to describe and explore the meaning through texts or visual data (Hunter & Leahey, 2008). The quantitative methodology that was appropriate for research was chosen in terms of digital data analysis, feature selection, classification, category classification and measurement.

The quantitative approach assists in representing data by numbers and graphs. This method was successful according to the need to analyze textual data, and extract part of the speech features as an example of verbs or names in a text, so that we can test or agree to experiments that will be explained later. As we mentioned earlier, this thesis aims to design a methodology based on certain algorithms, to extract quantitative information from the Arabic text file and then classify it into three types (extreme, non-extremist, neutral). Since all previous researches (in Arabic), were based on a qualitative rather than quantitative approach, we did not find enough data in the literature to support this type of methodology. In this thesis, we were compelled to collect his information and classify it manually to build the algorithm, train and test it correctly. This methodology is designed in several stages, as shown in the following

figure. Figure 3.2 shows the main process to perform text classification.

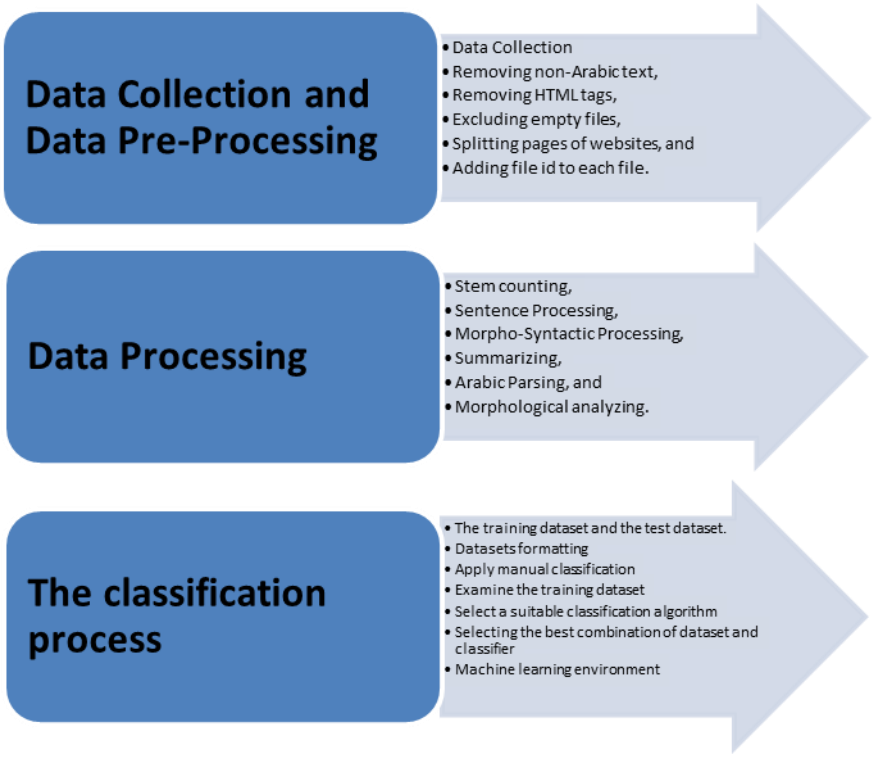


Figure 3. 2: Main process to perform text classification

As shown in the previous figure, the system design can be divided into three main phases. The stage of data collection and pre-processing, where data was collected manually from more than 7000 text files in the Arabic language from various sources and it contained terrorist, anti-terrorist, neutral data. These files were classified manually by specialists. After extracting the files that fit the subject of this thesis, they were pre-processed in several steps to be suitable for the next stage. The stage of data processing where data will be summarized, extracting its features through specific programs, will be discussed later. The last stage is the data classification stage where data is classified into our three categories (extremism, anti-extremism, and neutral data) using classification software (WEKA). After all this, the system will be trained through the training set, and then it will be examined through the test set.

3.5 Research methods

As we explained earlier, the research has been divided into several stages, and in each stage certain methods and tools have been deployed. The following sections

will explain these methods and equipment.

3.5.1 Data collection and preprocessing stage

Our research starts with data collection from sites. The data collected should be a mixture of locations that considered to be antiterrorism, pro-terrorism, and neutral sites to ensure balanced datasets for training and test datasets. Next to collecting datasets is to perform preprocessing for the data that include but not limited to:

- Removing non-Arabic text,
- Removing HTML tags,
- Excluding empty files,
- Splitting pages of websites, and
- Adding file ID to each file.

3.5.2 Data analysis stage

The step following data preprocessing is to apply text analysis toolkits to derive detailed information on the Arabic file content. The result of this process is to generate summary files containing all numeric, quantitative information about the Arabic text files. In addition, an N-Gram file is created to be used for the classification process and prediction calculations.

The main competition among the different data processing tools available lies in the number of distinct features that can be extracted from Arabic text. The more features, the more quantitative information, and, potentially, the more precise will be the classification. We should note the need for an Arabic language expert working side-by-side with the developer to review and audit the results coming out of each tool, to make sure they are semantically correct.

The main operations on the Arabic text should include the following:

- Stem counting,
- Sentence Processing,

- Morpho-Syntactic Processing,
- Summarizing,
- Arabic Parsing, and
- Morphological analyzing.

We used the Posit and SAFAR tools, which gave more than 58 features together, to create Summary files. Once we get summary and N-Gram files we are ready for data classification. The classification process is divided into two main steps; the first step is to ensure that the training data set, which is manually classified, will produce a high-quality model for future use. Then, in the second step, we can test and create the model.

3.5.3 Data classification stage

The next step is to use the model file that is created in the first step for classification of the unseen dataset to calculate a prediction for each file individually. To calculate prediction and to construct a confusion matrix, we store the extracted quantitative data as well as N-Gram data in a suitable format for training and test datasets. The classification is done by studying the attribute parameters during the training phase, and then considers the hidden files in order to predict the class for each new data item. The classification process can be divided to two main steps. Figure 3.3 (above) shows cross validation and model creation. Figure 3.4 (below) shows the classification process.

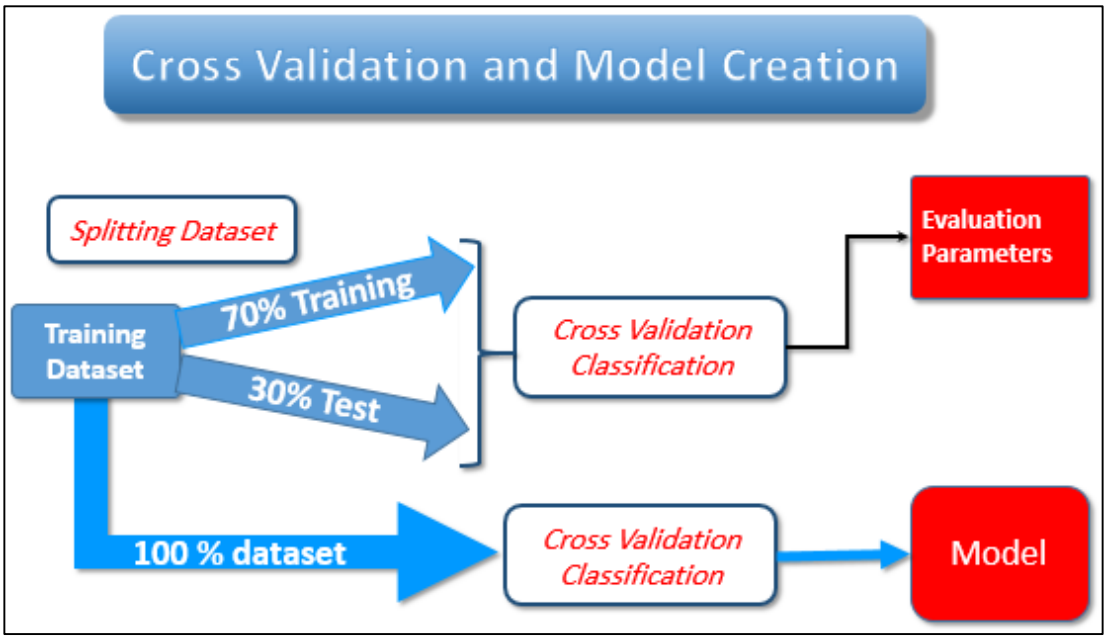


Figure 3. 3: cross validation and model creation

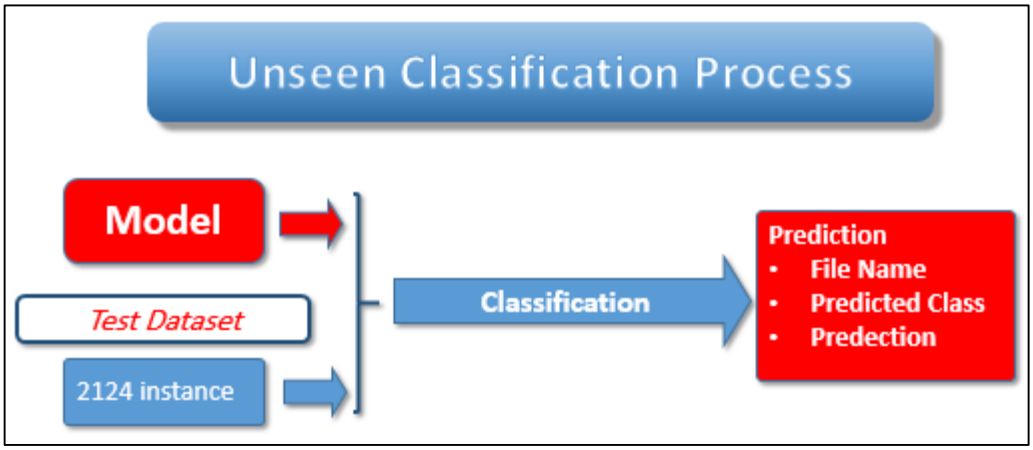


Figure 3. 4: classification process

To perform the classification process for the unseen dataset, we follow these detailed steps:

- Divide the collected corpus into a training dataset and test (unseen) dataset.
- Put all information collected that relates to each file into a suitable format for the classification process (ARFF file format).
- Manually classify the data samples by a high-qualified person for the training dataset. The purpose of the training dataset is to create a classification model used subsequently in the classification of the unseen dataset.
- Examine the training dataset for the quality of classification. We divide the data

into 70% and 30% subsets. We use 70% datasets as a self-training dataset and 30% as self-test datasets.

- Explore the use of different algorithms for classification. To choose the most suitable classification algorithm, we study many classifiers that can be listed under different classification concepts. The results are not selected based on the classifier only but also depend on dataset combinations of the two text analysis toolkits and the use of N-Grams generated by both text analysis toolkits.
- Select the best combination of dataset and classifier, based upon the precision, Recall, and F-Measure.

  We selected WEKA (machine learning environment) as a basis for our classification work because it is rich with a classification environment with attribute processing like attribute selection and a rich library of machine learning algorithms. Moreover, it has an API to be used to throw user-made applications.
- The programming language selected for creating the user interface is JAVA. It can utilize WEKA API to produce an efficient application that can fulfill all research requirements, including classification, and put the results in a suitable form for analysis.

3.6 Research tools

3.6.1 Posit Toolset

According to Weir (2007), the Posit Textual Analysis Toolset generates 27 features, including frequency data, in addition to Part-of-Speech (POS) tagging. It contains several profilers:

POS Profiler: Performs analysis of a given text corpus to derive statistics on the POS characteristics of that text, among which are types/tokens, number of sentences, average sentence, word length, common nouns, and proper nouns, verb tenses and total frequency of each POS token or type.

Vocabulary Profiler: Based upon the statistical data output by the POS Profiler, the Vocabulary Profiler can determine the relative frequency of occurrence for vocabulary items in the selected corpus.

Readability Profiler: This software component, (under development) will focus on text readability, based upon the statistical analyses from the POS Profiler and the frequency data from the vocabulary profiler. Word commonality (Weir & Ritchie, 2006) and average collocation frequency (Weir & Anagnostou, 2007).

3.6.2 SAFAR

The SAFAR (Software **A**rchitecture **F**or **AR**abic language processing) program is a platform dedicated to natural Arabic language processing (ANLP). It is an open source program, cross-platform, modules, and provides an integrated development environment (IDE) that includes: resources for different treatments, ANLP, basic language level units, especially those for the Arabic language, namely morphology, syntax, semantics, and finally requests for ANLP. This program was used in the stage of data processing in the proposed system, as it worked on extracting quantitative information from text file data.

3.6.3 WEKA API

The text document classification problem is a special case of a supervised data mining issue. To solve the problem of categorizing a text document, some steps are required. Common steps are: feature extraction, feature selection, rating and visualization. WEKA is a framework that helps categorize any text document after extracting its features. WEKA's began as a Java library to help implement data mining applications and was later developed to avoid the need for Java programming skills. WEKA components are also available visually within the "WEKA Knowledge Flow Environment". The most important feature of this framework is the ability to test different visual styles without programming capabilities.

This framework was used in the proposed system in order to classify the processed text data into three types (terrorist, anti-terrorist, and neutral). As a supervised learning approach, this needed training on pre-classified dataset.

3.7 Sample

WEKA API classifier needs training in a pre-categorized dataset to learn how to differentiate between our three categories (pro-terrorism, anti-terrorism, neutral)

and the distinct features and words for each category. In this system, a train data set of 300 files of textual data containing the three categories was used to train the classifier, and then this was tested to check its accuracy and effectiveness.

3.8 Validity and Reliability

Validity and reliability were taken into consideration in each step of the system design and implementation. First, collecting data, which was done by using several different sources (neutral sources, sources supporting terrorism, and anti-terrorism sources), whether it is on the Internet, social media, and elsewhere.

Manual data classification stage was performed by five different specialized people. Once classified in a category by at least four people the file is considered classified and added to the Corpus, otherwise it is removed from the Corpus group. In the next stage, a program was used to process the Arabic texts.

The next stage was to train the classifier through a group consisting of 300 text files of the three types. Finally, to ensure the validity of the training, progress and design of the program, the program was tested through a test dataset consisting of 200 text files of the three categories. The 200 files were completely correctly categorized, which demonstrated the validity of the proposed program's work for this message.

3.9 Summary

From the foregoing description, we see that our quantitative approach is best suited to our research. The next chapter will provide a more detailed account of our implementation.

CHAPTER FOUR: IMPLEMENTATION

4.1 overview

This chapter will be the core and the Practical application of the study. In which the clarification of what will be done during this study in terms of data collecting, organizing, and preparing. How the data was manually classified to get a ready-to-study corpus, and what methods were used. This chapter also describes the configurations of Posit and SAFAR that were required to deal with the Arabic language. In addition to how this research used them to extract the properties which are the basis for the Arabic text classification. Later in this chapter, the results will be explained how it converted into a format that WEKA can identify and deal with. Finally, it describes how to benefit from WEKA and its capabilities to classify Arabic texts based on the information extracted by Posit and SAFAR.

4.2 Data Collection

4.2.1 Corpus Building and compiling

A web-crawler was used to browse the web and the dark web, to collect and retrieve Arabic language pages, concerning extremism and related topics. The retrieved pages were analyzed and the links from those pages followed recursively. The best tool found and tested for this purpose was Crawler 4j, which is a Java API independent platform that has proven to be a reliable toolset and especially beneficial because it also contains a parser. Via the visited site domain and web URL, all retrieved data files were automatically labelled for future classification.

In order to deploy this tool in collecting a new Arabic extremist dataset, additional effort had to be made using this tool. Many websites contained data that was potentially related to our target, but this tool requires that URLs be identified beforehand. About 175 MB (1865 files) of related data have been collected. Figure 1 in Appendix A shows sample of the crawler code, while Appendix B includes some codes.

The initially collected data proved to be smaller in quantity than it hoped. The majority of known Arabic extremist websites, including their Twitter feed and Facebook accounts, are banned and have been shut down by the relevant authorities.

Researcher had to search further into the open web and the dark web, but such searches could expose the researcher to dangerous consequences from both governments and hackers. To overcome this issue, a paid VPN called Speedify has been subscribed, to make us anonymous and untraceable as far as possible

Further means of retrieving Web-based data was secured by subscribing to a specialized facility called Sketch Engine. This Website, enabled us to search by words, websites, and URLs through the open web, gather the data, compile the text into sentences. It allowed downloads of the generated corpus in text or vertical files (A vertical file is a text file where each token (or word) is on a separate line) format. That's mean it gave the user more facilities to work (see figure 2 in Appendix A).

The search keywords as an example ("الجهاد ضد الكفار", "داعش", "مقاومة التطرف", "الجهاد الإسلامي ضد الحكام العرب" , "مناهضة الحكم الكافر", "جبهة النصرة") are collected from internet web sites related to the main categories, which are:

- Pro-terrorism,
- Anti-terrorism, and
- Neutral.

The file was forwarded to the java tool, which split the resultant file into as many files as the number of sites collected, by Sketch Engine. Now we had many text files supposed to belong to the category related to the search word collection. This was the result of an internet search for common words linked to the selected category. According to the nature of the decided research to use a variety of the text files collecting from internet. with different categories (Pro-terrorism, Anti-terrorism, and Neutral)

Initially about 7000 files were collected which were manually classified. The first phase of supervised machine learning is to make manual classification to ensure that the learning and test text files are correctly classified; this provides a 'ground truth'

and should yield better accuracy of classification. Manual classification may solve the problem of the wrong categorization during data collection from websites, and partially solve the problem of repeated site content. After the initial classification and to achieve high accuracy of the data, the manual classification for data samples was achieved by giving the sample data to five different people with higher education levels but belonging to various fields of knowledge (accountant - social sciences scholar- Kindergarten teacher – Ph.D. in Arabic Linguistics – Translator). A contract has been made with these advisers to read the files one by one and make the classification depending on the content of each file. After receiving their responses, all the results were compared, and approved the classification of files for which the informants had unanimously agreed. Files that lacked this unanimity were excluded from the data set.

The result of this process was 5,124 text files, manually classified in 3 folders: pro, anti, and neutral. |In addition, the data was cleaned to remove HTML, any rubbish data, and every line with a sentence in XML is tagged, with the first line of the file's given URL tag. To do this, a Java class was programmed to read the whole text corpus and split it into files according to </files> tag embedded in the Sketch Engine compilation. To retrieve data from the dark web, a browser engine was installed that contains a simple VPN that permitted us to search Onion websites which is a specific web file extension special used in dark web pages and not available in ordinary normal internet websites.

The Arabic extremist ideas are truly hidden within the text, and a try to develop means by which to detect them was performed. The research continued to crawl more Arabic data from the open and dark web. The corpus downloaded from the Sketch Engine after compilation is one big text file, formatted with XML tags.

Figure 4. 1: Corpus before splitting

Figure 4.1 shows the data extracted from Sketch Engine. This significant corpus needs to be split into files to represent each original HTML webpage, then cleansed and normalized. It also needs to be referenced in a database by ID to keep its origins after normalization and processing. The Text normalization is the process of transforming text into a single certain form to be able to stored or processed and is used when converting text to speech. Numbers, dates, acronyms, and abbreviations are non-standard "words" that need to be pronounced differently depending on context" as by (Sproat, 2001) to split the corpus a java splitter class, was programmed, which reads the corpus line by line, and at the specific tags performs the file splitting (see Appendix A).

The manual or visual inspection of the Arabic Text files is an essential step in the classification procedure. These results were in content across our three categories: Anti, Pro, and Neutral. Due to the large number of files and the likelihood of human error, some files could be classified into two or more categories. This issue has raised the need for some semi-automated processes to help in discovering instances of mis-classification.

Our solution to this problem involved two main components:

1. A database (named corpusDB.db).
2. A Java code (named corpusDatabase.java).

The database is composed of one table that contains information about all the files, including the file ID, the file location, and other information. The Java code's role is to loop through all the files, extract the required data, and insert them into the database. After that, an SQL query was made to discover any repetition in the files. Having file names provides the additional step, namely, locating that file and visually re-inspecting it to check whether it is in the correct folder. If not, it is deleted manually from the wrong folder. Referencing the files in a database is the next step with all the data that enables identifying the files automatically.

A SQLite database engine has been used to create the database (see Appendix A). This has been chosen because of its portability, as it is a server-less database, and because of its cross-platform features, which will facilitate using it under Windows and Linux alike. DB Browser for SQLite has been used as the database browser.

One table has been created (see Appendix A) to contain all the data attributes collected from each file. All files are formatted in the same way so that the first line always holds the URL origin of the file and the file id. Figure 4.2 shows the file formatting before extracting the attributes to the database table.

Figure 4. 2: text file format

Java code has been written using Eclipse with jdk 1.8 and the sqlite-jdbc-3.2.1 library to extract the required data from each file, i.e., ID, file name, URL, first line and the manual classification of each file, and insert them into the database.

The java program loops through the three folders with the three file categories (anti, pro, and neutral) using a function called “looper” to scan the folder; moreover, within this function, a call to another function called “insert” is made to insert the extracted data into the database. For approximately 5,100 files, the code took about 4 minutes to finish and the resulting database table is shown in figure 4.3.

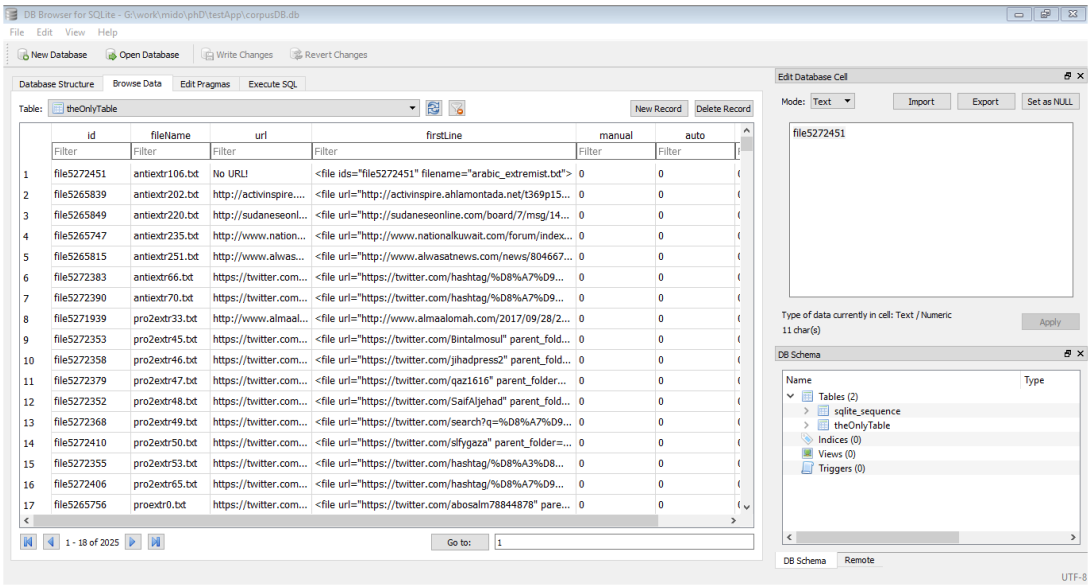


Figure 4. 3: The resulting database table

As shown in the figure 4.3 the file id, file name, URL, and the first line are extracted from each file and inserted as fields in the table. There is a field named “manual”; that holds an integer value of:

- 0 for pro,
- 1 for anti and
- 2 for neutral.

The field named “auto” is intended for future use when automatic analysis is conducted in WEKA classification experiments for these files. The results will be compared to the number in the “manual” field to calculate the percentage accuracy of the automatic classification process.

Also, it is worth mentioning that the table will help us, in the preparation phase, to locate redundancy in the corpus files, if any does exist, and to remove the repeated files easily. This is done by using a SQL code to group by id all the files that may have

different names but the same content. The result is a list of file names for the files that have the same ID, and this is used to delete these duplicates.

4.3 Data Preprocessing

The essential differences between Arabic language and English meant that changes were required in configuration for the software tools being targeted at our Arabic data.

4.3.1 Working with Posit

This was applied particularly to the Posit toolset that was designed originally to analyses English language texts. The following details the changes required for our Arabic context.

1. The first one is to modify the Posit toolset to be capable of POS tagging Arabic words and extract n-grams from Arabic sentences, whilst keeping the original 27 powerful Posit text features.

2. The second requirement is to use an Arabic-specific package, a tool named SAFAR v2, which can be used for POS and morphological analysis on Arabic datasets that are sourced from the web and produce 31 features.

Modifying Posit text profiling tools for use on Arabic text

i. Using Arabic tagger:

The first modification to Posit is to make it deal with Arabic rather than English parts-of-speech. This requires a process on the text using an Arabic tagger (the Stanford NLP POS-tagger) instead of the Lapos (English language) tagger from the original Posit tools package. The Stanford postagger-3.8.0 package that was applied for tagging the Arabic corpus includes a shell script named Stanford-postagger.sh that can be used to call the Arabic model.

ii. Using Arabic models:

The second modification that made to Posit to make it deal with Arabic text was to use Arabic language models in conjunction with the Arabic tagger. Integrating the Stanford tagger with Posit could then work successfully. Figure 5.15 shows the Arabic Models that are provided with the Stanford tagger (StanfordNLPGroup, 2019).

iii. Modifying Code:

Some changes to the Posit code have been made to allow it work with Arabic text and, as noted previously, Posit is a Unix command-line system that comprises a series of individual programs.

Firstly, by applying some changes on the file pos_all.sh, which is a shell script file used for calling every intermediate shell command in order of execution. These shell commands perform data preprocessing and functionality to analyze corpus files and stream out a summary of all the part-of-speech features needed for later processing. The changed tagger to the Arabic tagger, to be able to process Arabic text. These figures highlight the changes made to invoke the Stanford Arabic tagger (see Appendix A).

iv. Results

After applying the amended Posit to the Arabic files, it has been successfully generated the results for each text file. The main need result to use in our approach is the summary file. The summary file contains 27 features, which were used later in the classification phase. An example Arabic text file before processing by Posit and after the Arabic text has been processed by Posit is shown in Appendix A. This details 27 features extracted from the original text.

v. N-grams

The other modification to Posit was to ensure that the n-gram calculations were effective on Arabic Text. In the fields of linguistics and probability, an n-gram is a contiguous sequence of n words or items from a given text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. When the items are words, n-grams may also be called shingles. The size of n can be 1,

referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram".

In Appendix A two figures shows the original ngram.sh in Posit and highlights changes made to the modified ngram.sh. Specifically, the cat command was used to trim and clean Arabic text and add the result to a temporary file, in order to apply the n-gram coding on a cleaned text. The original code removed any numeric and non-alphabetic letters, but this approach is not useful in our case as all Arabic letters are non-alphabetic, so the code was changes to trim all alphanumeric letters and special characters, leaving only the unrecognized letters and non-English letters. Thereby, after trimming any noise, the result is a successful clean file of Arabic text tokens that the n-gram program can process effectively.

4.3.2 Working with SAFAR V2 in windows environment

1- Summary Generator

In this section, the use of the SAFAR tool will be described as an alternative means to Posit for preparing a dataset to be used as the training dataset. The Posit textual analysis offers 27 distinct features involving frequency data and POS tagging while SAFAR offers 31 distinct features as shown in figure 5.43.

Two classes in the SAFARV2 package for morphology analysis have been tested; BAMA and ALKHALIL. The results of the ALKHALIL class have been visually inspected and proven to generate more accurate details than BAMA class, and in turn, more accurate summaries than that in the BAMA.

A Java application, SummaryGenInteractive.java, has been written to generate the following outputs for each Arabic text file:

i. Summary file: containing all the Arabic features extracted from the file as shown in figure 5.43 and it includes the 31 features.
ii. 2-gram file.
iii. 3-gram file.
iv. 4-gram file.

The execution of this operation involves the following procedural steps:

i. Reading all the input files by selecting the folder containing them.
ii. Generating an output folder to hold two processed files (Split-out file, Normalized file) for each input file:
- Split-out file: with all the recognized sentences in XML format.
- Normalized file: with all the sentences removed all noise and non-Arabic characters.

iii. Generating an output folder to hold the required output files (n-gram, and summary)
iv. Generating the helping files.
v. Generating the summary files.
vi. Generating n-gram files.

- **GUI coding and explanation**

This section shows how all these details of SAFAR operate by starting with the summary generator.

i- Summary Generator:

Reading the input files. In the figure below this step show how the Java application allow you to generate the input files and select them from your files.

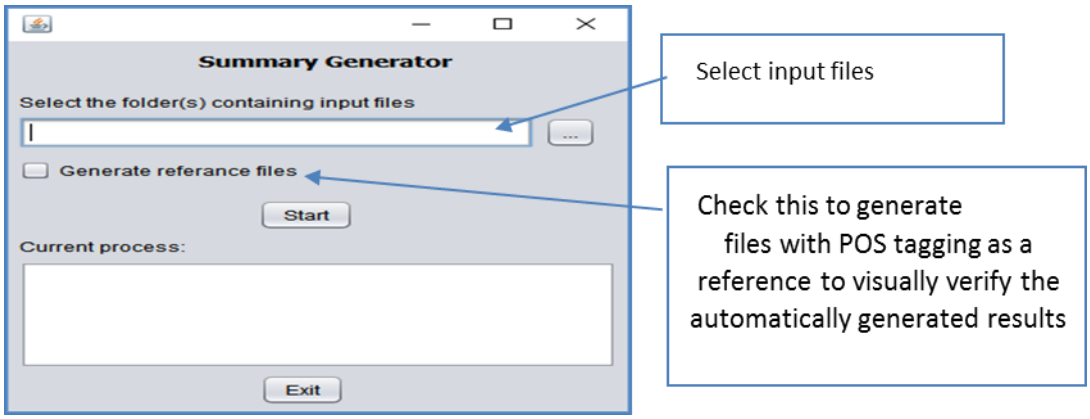


Figure 4. 4: Reading input for the Summary Generator.

A pop-up file dialog will arise, from which was selected our input files, the desired files were selected, then analysis will be started and the output generated.

After analysis, the output generated, a number of files processed were displayed on the screen. In the “output” folder, a folder will be created for each of the input folders. And in each folder, there will be a folder for each of the input files.

Since there is a need to loop through all the files in all the selected folders, this is done with a function called "looper" that accepts an array of file names and the folder where they reside. The operation of the looper function is clarified in Figure 4.5.

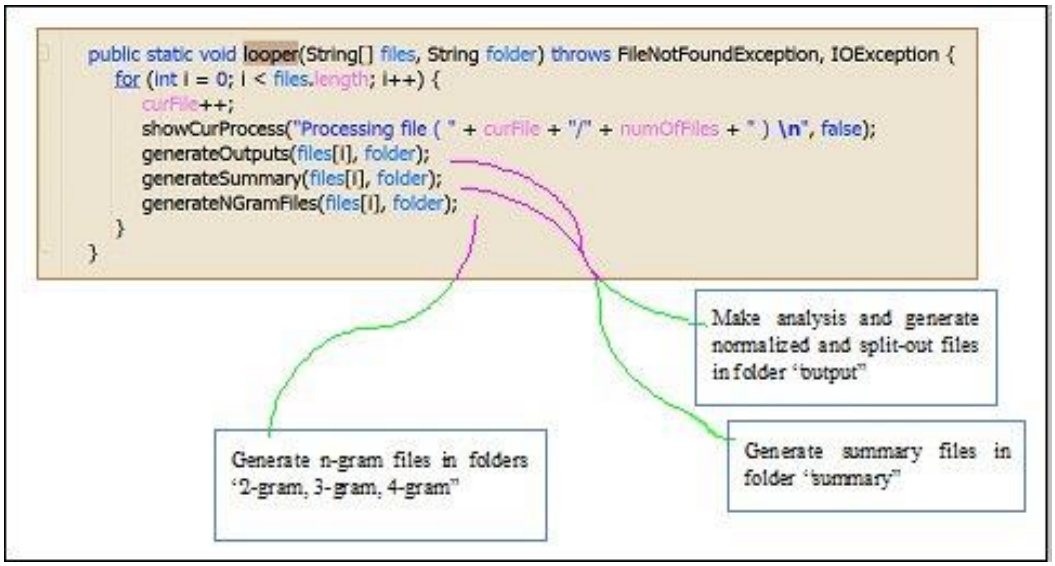


Figure 4. 5: The function of looper

The function “generate outputs” is responsible for generating the “split-out” file and the “normalized” file. The generated files will be named after the original name of the file, for easy referencing in the code.

The split-out file is a result of a method in SAFAR Sentence Splitter class called split () that generates an XML file with the sentences of the file, each appearing as an element in the file.

This file is accepted as a parameter in another function that generates the normalized file using a function called normalize (), which is a method in SAFAR normalize class. Figure 4.6 shows this process.

Figure 4. 6: Process of summary generator

After finishing (generate Outputs), generate Summary is called. This is where all the quantitative analysis results are stored as features in the summary files.

The POS noun analysis, for example, is made on each word in a file that has been identified as a noun using a class named Noun Morphology Analysis. This analysis provides a method called get(Pos) that is used for counting the number of occurrences of each noun type in the file.

An example of a generated summary file called: summary_antiextr13.txt with the features and their counts showed in Appendix A. For each file, after generating the summary, the 2, 3, and 4-gram files are generated using the function called generate N-gramFiles

4.3.3 Summary of Data Pre-Processing

As seen in Appendix A, results from both Posit and SAFAR toolsets are summary files and n-gram files 2, 3, 4-gram files. Before going forward, many features resulting were found from the two pre-processing tools (Posit and SAFAR) seem to be the same (at least in property name) but, due to different processing algorithms, the results are not identical, with slight differences that can be cited as Posit, it is not giving some features as singular, plural, feminine, muscular, modaf, mansob and majror count which is some grammatical features not found in English language but SAFAR V2 package could offer it as its specialized in Arabic language grammar processing. This helps to analyze each file with different algorithms. The final results show that the two different approaches support each other and help us to achieve higher model accuracy in the classification phase.

4.4 WEKA Implementation

a Java project was created to perform all the tasks required for data classification. GUI (Graphical User Interface) was implemented because it is easy to

use and simple to understand. In the following, it illustrated the user interfaces and the steps to fulfil the tasks required for the classification.

4.4.1 GUI used to Implement WEKA

A. GUI-A: Database Preparation, and N-Gram Calculation

(GUI-A) interface allows the user to prepare the database and conduct the n-gram calculations. The operations resulting from use of this interface are listed below.

- Add file names to the database according to manual classification for the three classes {anti, pro, neutral}.
- N-Gram reference data are storing in the database (table names: 2_ref, 3_ref, 4_ref).
- Calculate the 9 n-gram values average values (low/high/average for 2-3-4 N-Gram) and store it in database (table: ngram, sgram).
- This process is done for Posit and SAFAR data in separate tables. All result data are stored in SQLite database for use in ARFF file creation.

B. GUI-B: ARFF file creation

The second interface (GUI-B) is used to create the required ARFF file formats for use in WEKA. The operations resulting from use of this interface are listed below.

- Create ARFF files containing all possible combinations of Posit/SAFAR with/without N-Gram and mixing Posit+SAFAR attributes to create ARFF files according to user selection.
- Create all required subsections of ARFF files for classification (70% & 30% Datasets –100% dataset – unseen datasets for each dataset.

Each single click adds more than one file, each file contains a dataset to be used in the classification process and model building. The datasets required are 70% training dataset, and 30% test dataset as well as other datasets required for the supervised unseen tests. Naming is done according to the file content. A complete list of file names and file descriptions is given in Table 4 (below).

	ARFF Files	Number of Instances	Purpose of Dataset
POS	pos_70.arff	2100	70% training sample
	pos_30.arff	900	30%test sample
	pos_100.arff	3000	Create Module
	pos_test_class.arff	2124	Supervised Test
POS + N-Gram	posGram_70.arff	2100	70% training sample
	posGram_30.arff	900	30%test sample
	posGram_100.arff	3000	Create Module
	posGram_test_class.arff	2124	Supervised Test
SAFAR Toolset	safar_70.arff	2100	70% training sample
	safar_30.arff	900	30%test sample
	safar_100.arff	3000	Create Module
	safar_test_class. Arff	2124	Supervised Test
SAFAR+N-Gram	safarGram_70.arff	2100	70% training sample
	safarGram_30.arff	900	30%test sample
	safarGram_100.arff	3000	Create Module
	safarGram_test_class.arff	2124	Supervised Test
POS + SAFAR	total_70	2100	70% training sample
	total_30	900	30%test sample
	total_100	3000	Create Module
	total_test_class	2124	Supervised Test
POS + SAFAR + POS N-Gram	total_pGram_70	2100	70% training sample
	total_pGram_30	900	30%test sample
	total_pGram_100	3000	Create Module
	total_pGram_test_class	2124	Supervised Test
POS + SAFAR + SAFAR N-Gram	total_sGram_70	2100	70% training sample
	total_sGram_30	900	30%test sample
	total_sGram_100	3000	Create Module
	total_sGram_test_class	2124	Supervised Test
POS + SAFAR + Total N-Gram	total_psGram_70	2100	70% training sample
	total_psGram_30	900	30%test sample
	total_psGram_100	3000	Create Module
	total_psGram_test_class	2124	Supervised Test

Table 4. 1: complete file names according to file description

c. GUI-C: WEKA classification

The third interface (GUI-C) is used to invoke the classification process on the prepared data. This Java interface employs the WEKA API to apply a classifier on the selected dataset and output the file prediction. This interface (GUI-C) supports the following operations:

- Apply selected classifier algorithm on selected dataset
- Show (30%-70%) cross-validation test results.

- Generate model file using (100%) dataset.
- Using a Model file to classify unseen test dataset with or without N-Gram attributes.
- Optionally show prediction details (per file).

4.4.2 ARFF file

In order to use the summary file information (the output of the Posit and SAFAR text analysis toolkits) in classification, this output has to be in ARFF file format (Attribute Relation Format File) that is suitable for WEKA framework processing.

- ARFF files have two distinct sections. The first section is the Header information, which is followed by the Data information. The ARFF file format is as follows:

- The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types (Paynter, Trigg, Frank, & Kirkby, 2008).

- The data section is made up of individual lines; each line contains comma-separated values of the attributes arranged according to their order in the header section.

- **Create ARFF Files**

  The detail in the Attribute-Relation File Format (ARFF) file consists of 3 parts:

  - @relation: represents dataset name.
  - @attribute: indicates the name and data type of features extracted from the document.
  - @data: each instance is held in a single line, with each line separated by a carriage return.

As noted, attribute values for each instance are delimited by commas and must appear in the order that they were declared in the header section, i.e., the data

corresponding to the nth @attribute declaration is always the nth field of the attribute (Al-Zoghby et al., 2007).

Create ARFF file

1- Source text file:

The dataset in the form of an ARFF file is created from the summary files that result from applying the Posit and SAFAR tools. Each summary file represents the result of calculating all parameter values extracted from each source web site.

2- Creating an ARFF file using summary file information includes the following steps:
   - Collecting summary file names.
   - Read the file information line by line.
   - Split line string into sets of attribute and value pairs.
   - Form result ARFF file and save the file.
   - Split ARFF files into standard 70 % for later training and 30 % for later testing and the results.

The ARFF creation step is processed in three stages. In the first stage, data is collected from files considered as “anti” files into the dataset. The second stage, collects data from files considered as “pro” files into the dataset and the third stage collects information from files considered as “neutral” files into the dataset. Finally, these details are all saved into the ARFF file.

Summary files and 2,3 and 4 N-Gram are arranged in folders so that they can be accessed by JAVA code. The Java code is designed to read summary files from both the Posit and the SAFAR data files path. The steps of reading are as follows:

- Read folder files as a list of files and store the result in a database.
- Access all files, one by one, to collect data form file inside.
- Read file content as lines and parse each line for attribute and value pairs.
- Add all available attributes to an array of attributes and values.
- Connect to the SQLite database to get the last 9 n-gram attributes for each file and add them to a long array.

- As soon as the data is collected from all files, it starts to build the ARFF file using data stored in a long array after updating the last attribute by class value (pro, anti, neutral).
- The ARFF Saver method is used to save in ARFF format.

4.4.3 Prepare N-Gram Data

As a result of using the Posit and SAFAR tools, it have been derived 2-gram,3-gram, and 4-gram files, which contain n-gram details. This n-gram information indicates the frequency of occurrence of each sequence of words in the given text.

The first step to extract useful information from the n-gram file is to recognize the file structure. Each file name indicates the type of n-gram files 2, 3 and 4. The next step is the file inside; this consists of multiple lines, as many as n-gram items (tokens), extracted from the text. Each line contains the n-gram information with the n-gram tokens. The second part of the line is the frequency of occurrence of the n-gram token.

The next step is to gain information about the prediction of "How often the token is used in the text relative to the Arabic Language," or, in other words, how to calculate the probability of using the token in the text. To calculate this probability, it need a reference to compare the frequency of a particular token to the frequency of the occurrence of the same token in the language. Of course, the larger our n-gram reference files, the higher the plausibility of the prediction values, and therefore, the better the prediction for any token.

The reference n-gram is text files called Arabic_news_plainText which consists of 1200 Arabic news files from different aspects of news (), each line contains phrase contexts, and frequency is separated by a colon ":".

Reference File	Token count	File size
2-Gram	1,737,701	52,822 KB
3-Gram	3,040,427	121,694 KB
4-Gram	3,565,402	179,607 KB

Table 4. 2: n-gram Reference files properties. (sourceforge, 2017)

The idea of collecting n-gram information is to add reference n-gram information into a database and then read 2-3-4 gram files into an array. The text array is scanned, item by item, to get the token used to search the reference database (Table 4.2) for corresponding reference frequency.

The file size is one of the faced challenges for the following reasons:

i. Having 5124 files.
ii. There are three files (for 2-3-4 Gram information) associated with each text file.
iii. Each N-Gram file consists of 1000 – 4000 Token.
iv. Each token has to be searched in the corresponding reference file2-3-4 Reference Gram files.
v. Each search consumes from a few seconds up to minutes, when using traditional mechanical hard disk HDD to store database files and the 2,3- and 4-Gram files.

This means that the time required to collect token text frequency and reference frequency and store results (low average, high average, and overall average) in the database may take a few months. This is the time needed to calculate data required to add n-gram attributes to the ARFF file (9 attribute fetchers are planned to be stored in the database) .The taken steps in this experiment have managed to accelerate the process by speedup read/write process, exploiting the advanced properties of the SQLite database:

i. **Studying the use of SSD (solid-state hard disk):**

Using SSD rather than HDD enhances the performance and reduces processing time. Appendix A has a comparison table between mechanical and solid-state disk drive. However, this is not quite enough to significantly enhance performance. In the end, SSD did not used, but went for faster storage; it will be covered in the next step.

ii. **Studying the use of RAM disk**

This technology is an effective way to increase the read/ write speed. Using a RAM disk enhances the read/write performance, so that the time consumed by the processing does not depend on disk access, but on its database access and search algorithm. As a result, performance is enhanced by 18.61%. This, however, is not quite

enough to help the overall process. The free version of RAMDISK have been used, a program that takes a portion of system memory and uses it as a disk drive. The more RAM the computer has, the larger the RAMDISK it can create (Diffen, 2019). Complete installation and uses are available on the RAMDISK site. it did not benefit from read/write speed enhancement because most of the time was consumed in the search process but not in the reading /write process.

iii. **Studying database access performance:**

One of the essential topics of database access performance is the index. Every table may add to the index (AUTO_INCREMENT PRIMARY KEY). This index is useful for selecting item or items from the table. However, it is not searchable. When a query on the text column was executed, a way to avoid searching through every row to find the intended one was found.

Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows. The larger the table is, the more time this process takes. If the table has an index for the columns in question, MySQL can quickly determine the relevant point in the middle of the data file, without having to look at the entire range of data. This is much faster than reading every row sequentially (RAMDisk, 2019).

After creating a Text-Index, the search process becomes an easy one as its duration significantly decreases by 99.61%. Table 4.3 shows the improvement of using text index.

SUM of processing duration		
HDD	SDD	Text Index
3,994,000	3,249,000	12,670
Enhancement Percentage %		
	SSD	Text Index
Enhancement	18.6 %	99.61 %

Table 4. 3: Processing duration trials to reach the optimized processing time

Concerning the percentage of performance enhancements for N-Gram Low-High Average calculations, a SQLite official site informs us that Text-Index Indexes are special lookup tables the database search engine can use to speed up data retrieval.

An index is a pointer to data in a table. An index in a database is very similar to an index at the back of a book (Mysql, 2019).

The enhancement result is archived by efficient use of the database through utilizing the Text-Index, which applies various algorithms to reduce search time. The result values of processing duration to calculate low-average, high- average and overall-average using Java code can be found Appendix A.

The prediction of a particular was determined for N-Gram data provided by Posit. The frequency of occurrence of a specific sequence of the word does not present information about text type (Pro, Anti or neutral) unless this frequency is compared to the typical Arabic language use for this particular token. This gives us more information about how common or usual the use of particular 2-3-4 Gram is for the given token. To do that, an equation was applied that described by MichaelP. Oakes, in his paper “Text Categorization: Automatic Discrimination Between US and UK English using the Chi-Square Test and High Ratio Pairs.” (SQLite, 2019).

E = row total x column total / grand total

Where:

Row total: token frequency + reference frequency.

Column total: total token frequency.

Grand total: total (token frequency + reference frequency).

Table 4.4 shows an example of calculating the prediction for token occurrence.

2 Gram	Text Freq.	Reference Freq.	Row Total	Expected value
الأخرى التي	4	16859	16861	4.065667323071977
في العراق	2	8020	8022	0.6447775589760077
	6	24879	24883	

Table 4. 4: Example for calculating the prediction for token occurrence

Another similar table named “n-gram” is used to store 2-3-4 Gram values for N-Gram information for data, provided by SAFAR.

The average frequency was calculated of the reference file by the following simple equation:

$$\text{Average Reference Frequency} = \frac{\sum Frequency\ of\ Token\ Frequency}{Token\ Count}$$

The result is as follows:

avgRef2 = 4

avgRef3 = 2

avgRef4 = 2

The prediction value is used to calculate three average values for each file, which are low-average, high-average, and overall-average. The low average is the average of the sum of E for all values of E below or equal to the reference n-gram average. Similarly, the high-average is the average sum of E for all values of E above the reference n-gram average. The overall-average is the average of all values of E. As soon as this value was got, it is added to n-gramDB.db in n-gram. The rows represent n-gram information about all 2-3-4 n-gram average values corresponding to each file.

4.5 Summary

This chapter details the important applications in this study, how the basic data in this study was collected, how data was prepared for the study, and how Posit and SAFAR were used to extract important data that will benefit our study in the classification process. Also, a discussion of how to handle the results of our analyses to work with WEKA in the classification stage. The next chapter will describe how to deal with these programs, how these were applied in this experiment and what the obtained results were.

CHAPTER FIVE: EXPERIMENTS

5.1 Overview

This chapter details the experiments that were carried out on several datasets. The first part (70% training dataset and 30% test dataset) of our experiments was carried out to ensure the quality of manual classification. To create the model file, we use 100% (70%+30%) of the training data. Once a model file was obtained which carries all of the information derived from the training process for 100% of the training dataset, which was used to classify the unseen dataset.

The results from the manual classification process are the production of ARFF files for each dataset, for the Posit or SAFAR datasets including average ratios for 2-3-4 grams. The data classification process where applied different algorithms to each of our various datasets to get the classification prediction of each file in any dataset.

5.2 Experimental setup

In order to run this experiment, two main stages were followed. The first is collecting data from the internet, preprocessing, and extracting quantitative information, and storing these details in summary files. The second stage is to create a classification model to use into classifying unseen data samples.

There are two environments employed in our experiments:

1. Linux: Kali-Linux was used for data collection and preprocessing.
2. MS-Windows: Windows 10 was used for running the classification process.

In support of this effort, Java JDK 8 must be installed to provide the necessary libraries for the classification process and WEKA APIs for Windows must be installed.

Several issues were encountered in this practical phase of our work. First, a problem was encountered with Arabic text not being displayed and having insufficient memory for processing ARFF files. This problem could be fixed by changing some environment variables in the Run WEKA file. Specifically, the File Encoding was set

to utf-8 instead of Cp1256 in order to have the WEKA framework recognize and display an Arabic character set where needed. Increasing the assigned runtime memory for Java, served to address the ARFF file processing issue. These changes ensured that WEKA ran without further problem (Figure 5.1).

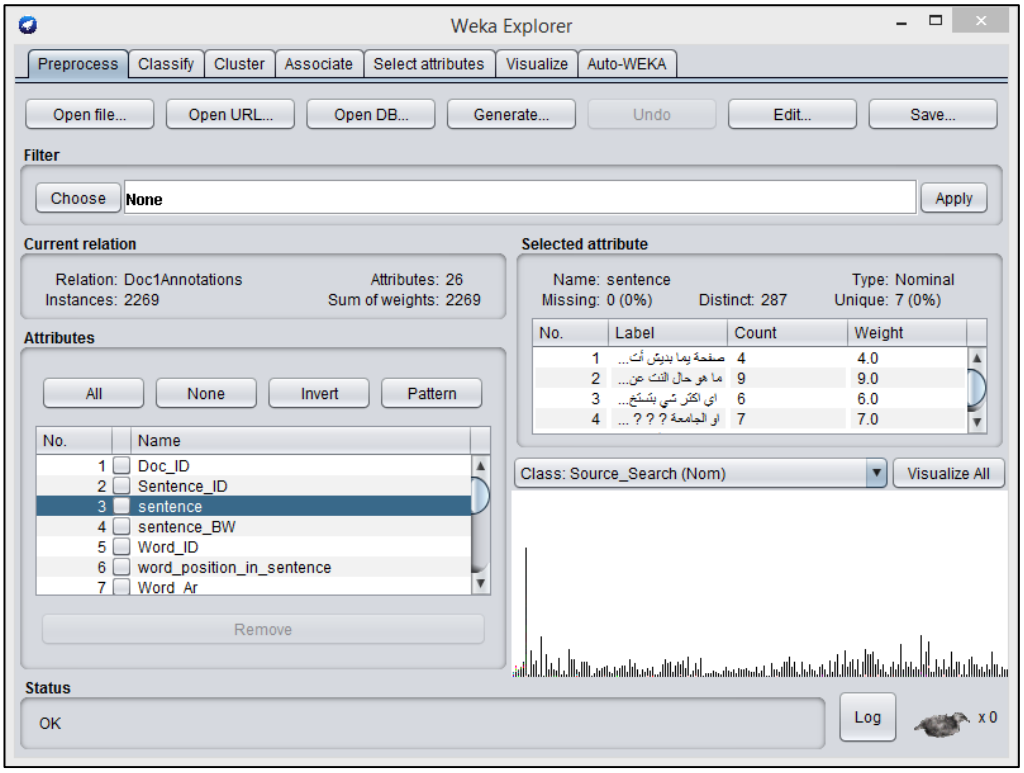


Figure 5. 1: results of changing some environment of Weka

5.3 WEKA

"WEKA is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code by adding WEKA API reference to the JAVA application. WEKA API is a library prepared by WEKA. It is a set of functions and procedures that allow the creation of applications, which access these features directly. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes" (Oakes, 2019).

It is a free machine learning software that provides a user interface and reference library (to be used from within Java code). It also provides algorithms for machine learning, including attribute selection filters and classification algorithms. WEKA was used for

selecting experiments and applying classification algorithms to the dataset. Then came applying the resulting model to approve the precision and accuracy of the model.

To be able to apply classification algorithms using WEKA (Waikato Environment for Knowledge Analysis) on both Posit and SAFAR results, the result should be formatted or arranged in a suitable way for processing. This format is called the ARFF file format (Attribute Relation File Format), and it is used by WEKA. This text file describes a list of instances sharing a set of attributes.

5.4 Our approach

The results from all steps or phases help us to produce quite enough data in a suitable format to produce file prediction. Every phase produces results that are required by the next phase. The collection process produces text files that contain all text information from the corresponding web page. Unfortunately, these text files contain HTML tags and other languages text (e.g., English). The second phase cleans all superfluous other language text and HTML tags. Next, all quantitative information were derived using the Posit and SAFAR toolkits. This quantitative information is used to produce different datasets, each in ARFF file format, to be used for the classification process by producing ARFF files, it is ready for the classification process to get the main results for our research, and to calculate the file class prediction (anti-pro-neutral) for each file in the test dataset using different classification algorithms.

After creating an ARFF file that is used to supply WEKA with input and test data, this automatically finds the data model (prediction pattern) to be able to determine data class (anti-pro-neutral) in the future. The other part of the data is used for the test set (unseen data) against the data model to ensure proper operation of the data prediction.

The classification process takes place by establishing recognition patterns to be able to make predictions about newly provided data (unseen data). The data available for building the recognition pattern is divided into two parts, the training sample for machine learning and the other sample for testing the recognition pattern created during the machine learning phase.

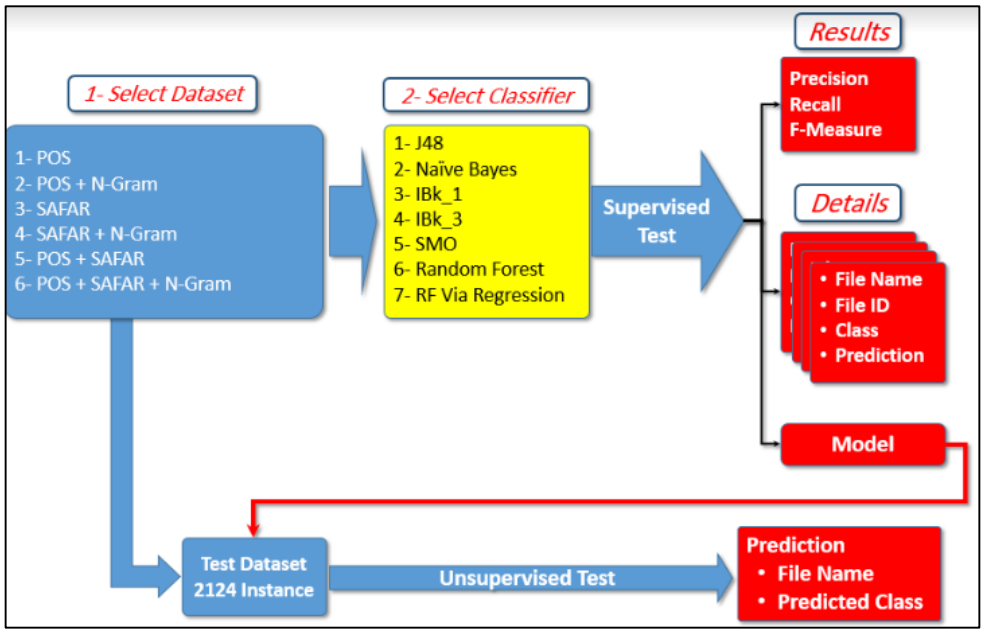


Figure 5. 2: Classifier-Dataset Selection and prediction calculation Diagram

Each of the two text analysis tools creates two sets of files; summary files that contain information about words inside the text and the n-gram files. To begin the machine learning, three pairs of sample/test ARFF files were prepared. The first group is created by utilizing the results of the summary file of Posit toolset (after making necessary customization to the toolset to suit the Arabic language). The second group uses the summary files result from the SAFAR toolset. The third group utilizes both Posit and SAFAR. each of the three groups in two phases will be treated, the first one is to classify the data without n-gram information, and the second phase is to classify for all information, including n-gram information. Figure 5.3 illustrates the different calculation methods.

The application of the classifier is made by dividing the text file into two main groups; training group used for machine learning and test group for validating the recognition pattern resulting from the machine learning process. The ratio of training data to test data is 4:1. The Posit data has 27 attribute values extracted from summary files. The SAFAR data has 31 attribute values extracted from summary files.

Next, the same classifier to the other ARFF files were applied, i.e., files containing the 27 attributes from Posit summary files and 31 attributes from SAFAR

summary files, plus nine attributes representing the 2, 3, and 4 n-gram information average values (low-average, high-average, and overall-average).

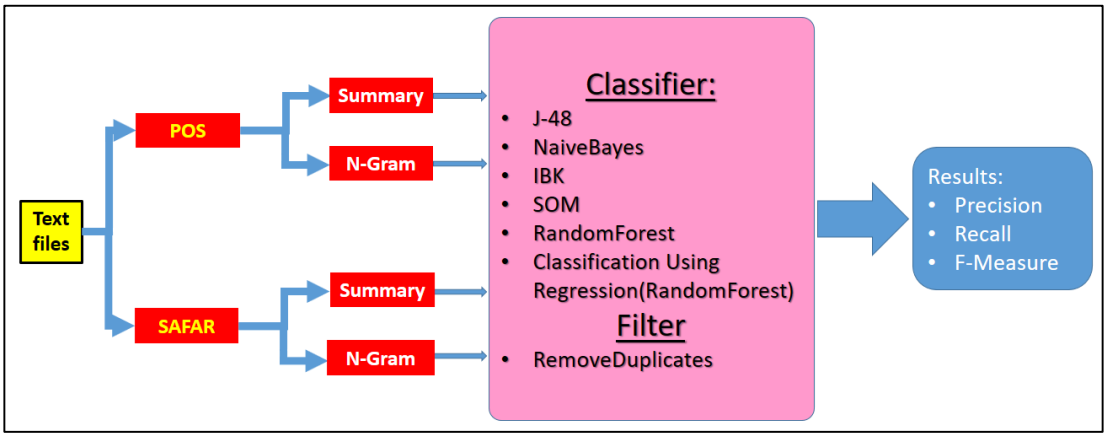


Figure 5. 3: Work Flow for different calculation methods

The different values of the 2 ,3 and 4 N-Gram average are calculated for each text file as:

$$Low\_average = \frac{\sum N\_Gram Values}{Number\ of\ N\_Gram}$$ Where N-Gram value <= Ref Average.

$$High\_average = \frac{\sum N\_Gram Values}{Number\ of\ N\_Gram}$$ Where N-Gram value > Ref Average.

$$Overall\_average = \frac{\sum N\_Gram Values}{Number\ of\ N\_Gram}$$ For all values of N-Gram values.

Many algorithms have been tried during the early stages to discover the best classifier, side by side, with the commonly used classifier for text processing. For this purpose, the following classifier was used to examine the most suitable algorithm for our machine learning.

1. J48 classifier,
2. Naïve Bayes,
3. K-Nearest Neighbor (KNN), for which an investigation of two values of K were performed (where k=1 and k=3). The value of k is the number of neighbors to query to make a prediction,

4. Support Vector Machines (SVM), and it have been found that the best parameter setting is the one that produces the hypothesis with the lowest VC—Dimension. This allows for a fully automatic parameter tuning without expensive cross-validation,

5. Random Forest, and

6. Classification by Regression (Using Random Forest classifier).

5.5 Summary

In this chapter, research experiments have been described to cover all combinations of applying selected classifiers on different datasets. The calculation of n-gram ratios was added as additional parameters to the summary file parameters derived from the Posit and SAFAR text analysis toolkits. In the next chapter, a discus will be performed of the collected results from classification process for training dataset cross validation and the results obtained from applying the classification model on unseen datasets.

CHAPTER SIX: RESULTS AND DISSCUSION

6.1 Overview

Classification is a data management process used to sort or categorize a dataset to distinct types. We apply different algorithms to a manually classified dataset (known as the training dataset) to produce classification requirements model then apply this model, containing dataset requirements, to unseen data (known as the test dataset) to calculate a prediction for each item. This chapter contains the analysis of our approach and results**.**

6.2 Analysis

The results can be summarized as file type prediction (anti-pro-neutral) and a confusion matrix that is used to calculate parameters (Precision – Recall- F.Measures) describing how accurate the classification is. The classifiers algorithms used to obtain results shown in Table 6.1 :

	Classifier
1	J48
2	Naïve Bayes
3	k-Nearest Neighbors (kNN3)
4	SVM
5	Random Forest
6	Classification by Regression (Random Forest classifier)

Table 6. 1: classifiers to obtain results

The datasets are arranged so that classifiers were applied on the summary data of each toolset used for the analysis of Arabic text files. The two toolsets used to generate summary files are in Table 6.2 .

	TOOLSET	Abbreviation
1	parts-of-speech Text Profiling Toolset	POSIT
2	Software Architecture for Arabic Language Processing	SAFAR

Table 6. 2: Tool sets to generate the summary files

Each summary file divided into two subsets:

- 3000 files used to create the Model and 70%-30% test.
- The rest of the files used for supervised labeled and unseen datasets classification process.

Four ARFF files were generated for each case; the first three files are created by using the manually selected files, which are the 70% training-set, 30% test-set, and the 100% dataset consisting of the 30%+70% files. The last ARFF files are the remaining 2124 files formed to suit the purpose of supervised datasets classification.

The used training-set and test-set to ensure proper operation of the manual selection, as soon as the best value of 70-30 classification precision was gotten, the Model file was generated, used as a classifier for Unseen data.

The model describes the information that is used when trying to deal with the new dataset. In a simple spam detection scenario, the algorithm determines which words seem to point to spam and which do not, by looking at annotated emails. The lists of words then form the model.

When receiving a new email, this won't be compared with other real emails; instead, the new email's words will be considered and checked against the model (word lists) to see whether they seem to indicate a spam mail or not. This way, your data will be independent from the training data. As a result, you will have a piece of knowledge that tries to model the whole "spam vs. non-spam"-reality, the model creation is done for each classifier in a separate file and stored in the ". /modelOut/" folder.

Eight files per classifier were created as follows (j48 and random forest were used, for example in Table 6.3:

	Dataset	J48	Random Forest
1	Posit	1j48.Model	1randomForest.model
2	Posit + N-GRAM	2j48.Model	2randomForest.model
3	SAFAR	3j48.Model	3randomForest.model
4	SAFAR + N-GRAM	4j48.Model	4randomForest.model
5	Posit + SAFAR	5j48.Model	5randomForest.model
6	Posit + SAFAR + Posit N-GRAM	6j48.Model	6randomForest.model
7	Posit + SAFAR + SAFAR N-GRAM	7j48.Model	7randomForest.model
8	Posit+ SAFAR + (Posit+SAFAR) N-GRAM	8j48.Model	8randomForest.model

Table 6. 3: Sample of Model file name

6.2.1 Output Specifications

Applying a model as the classifier is done in the last stage by means of GUI-C. Therefore, the model is a file carrying prediction criteria used to classify new unseen data. The corresponding model for each dataset will be used. Every single click is used to do several operations and display the results before going forward to the next step.

The three-classification processes are:

i. 70-30% Classification cross validation,
ii. 100% Classification and module creation,
iii. Supervised unseen dataset classification.

The final evaluation figure results from the classification process are summarized in three parameters, which are:

i. Accuracy.
ii. Precision,
iii. Recall, and
iv. F-measure.

All three results are derived from the confusion matrix.

A confusion matrix (Kohavi and Provost, 1998) contains information about actual and predicted classifications carried out by a classification system. The

performance of such systems is commonly evaluated using the data in the matrix. The following Table 6.4 shows a generic confusion matrix for a two-class classifier.

The entries in the confusion matrix have the following meaning in the context of our study:

a- The class is True:
   1. TP is the number of correct predictions that an instance is positive.
   2. TN is the number of incorrect predictions that an instance is negative.

b- The class is False:
   1. FP is the number of the wrong of predictions that an instance positive.
   2. FN is the number of correct predictions that an instance is negative

		Prediction	
		Positive (correct)	**Negative (Incorrect)**
Class	**True**	TP	TN
	False	FP	FN

Table 6. 4: The entries in the confusion matrix

This confusion matrix 2x2, which is used to derive the three results from the classification process.

- Precision: this helps to calculate precision as the ratio of how many events of the positive class were positive.

$$\text{precision} = \frac{\text{True Positive}}{\text{total Positive}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall: is defined as sensitivity, and this ratio represents the ratio of the positive events predicted correctly as positive:

$$\text{recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- F-measure: (known as F1), is calculated as the weighted average of the precision and recall.

$$\text{F\_measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

6.3 Experiments Results

6.3.1 Results 1 (Posit dataset)

In this test, the used ARFF file created depending on summary files resulting from the Posit dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	**Test dataset**
1	70–30% CLASSIFICATION.	pos_70. Arff	Seen dataset: pos_30.arff
2	100 % Classification	pos_100.arff	Seen dataset: pos_100.arff
3	Supervised Classification.	1j48.mode 1bayes.mode 1iBk_3.mode 1smo.mode 1randomForest.mode 1randomForestR.mode	Unseen dataset: pos_test_class.arff

Table 6. 5: Posit classification datasets

List of attributes (count = 27):

1) Total words (tokens)_pos
2) Particle_types_pos
3) Total unique words (types)_pos
4) Determiner_types_pos
5) Type/Token Ratio (TTR)_pos
6) Interjection_types_pos
7) Number of sentences_pos
8) Nouns_pos
9) Average sentence_pos length (ASL)_pos
10) Verbs_pos
11) Number of characters_pos
12) Prepositions_pos
13) Average word length (AWL)_pos
14) Adjectives_pos
15) Verb_types_pos
16) Determiners_pos
17) Noun_types_pos
18) Particles_pos
19) Adverb_types_pos
20) Possessive pronouns_pos
21) Adjective_types_pos
22) Personal pronouns_pos
23) Preposition_types_pos
24) Adverbs_pos
25) Possessive_pronoun_types_pos
26) Interjection_pos
27) Personal_pronoun_types_pos

The classification results are listed below Table 6.6, where it can be noticed when using Posit dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

	70-30 %.			100 %			Unseen		
	Prec.	Recall	F-Meas.	Prec.	Recall	F-Meas.	Prec.	Recall	F-Meas.
J48	0.971	0.971	0.971	0.975	0.975	0.975	0.703	0.660	0.672
Naïve-Bayes	0.811	0.611	0.597	0.765	0.622	0.608	0.708	0.386	0.426
KNN k=3	0.904	0.903	0.903	0.951	0.950	0.950	0.650	0.637	0.639
SVM	0.959	0.959	0.959	0.911	0.909	0.909	0.700	0.690	0.690
Rand-Forest	0.992	0.992	0.992	1.000	1.000	1.000	0.797	0.785	0.786
Classification by Regression	0.978	0.978	0.978	0.967	0.966	0.966	0.716	0.711	0.712

Table 6.6 Posit datasets classification results**.**

Figure 6.1 shows test classification results obtained by applying desired classifiers on Posit unseen datasets.

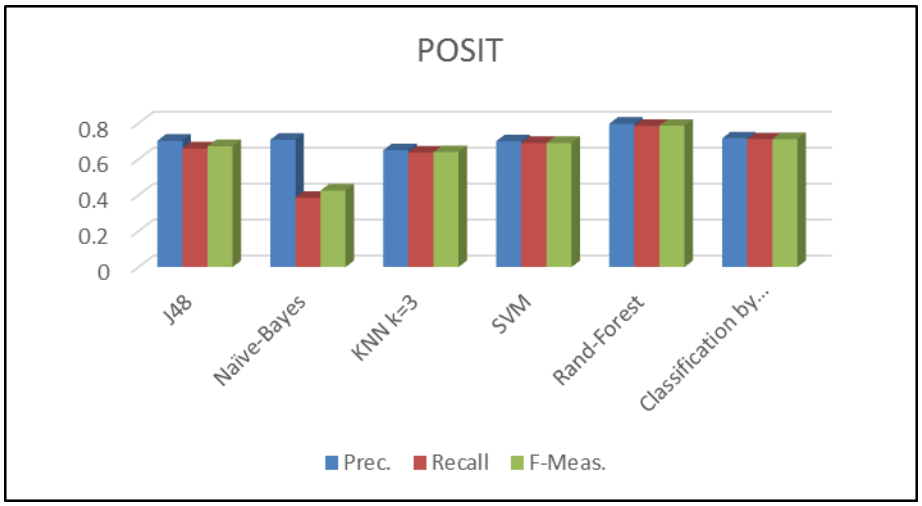


Figure 6. 1: Posit datasets classification results

The classification process for the Posit summary dataset shows the best result achieved by Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset.

However, Classification by Regression using Random Forest took second place for 70%-30% and 100% datasets and the supervised test for the unseen dataset. This is the first achieved result; these results push two classifiers to be the suitable classifiers for Arabic text classification, and they are:

i. Random-Forest.
ii. Classification by Regression using Random-Forest.

6.3.2 Results 2 (Posit + N-Gram dataset)

In this test, the used ARFF file created depending on summary files resulting from Posit + (Posit 2-3-4-Gram) dataset. Each classification process requires training and test datasets.

		Training dataset	Test dataset
1	70–30% Classification.	posGram_70.arff	Seen dataset: posGram_30.arff
2	100% Classification	posGram_100.arff	Seen dataset: posGram_100.arff
3	Supervised Classification.	2j48.mode 2bayes.mode 2iBk_3.mode 2smo.mode 2randomForest.mode 2randomForestR.mode	Unseen dataset: posGram_test_class.arff

Table 6.7 Posit + N-Gram classification dataset

List of attributes (count = 36):

1) Total words (tokens)_pos
2) Verbs_pos
3) Total unique words (types)_pos
4) Prepositions_pos
5) Type/Token Ratio (TTR)_pos
6) Adjectives_pos
7) Number of sentences_pos
8) Determiners_pos
9) Average sentence_pos length (ASL)_pos
10) Particles_pos
11) Number of characters_pos
12) Possessive pronouns_pos
13) Average word length (AWL)_pos
14) Personal pronouns_pos
15) Verb_types_pos
16) Adverbs_pos
17) Noun_types_pos
18) Interjection_pos
19) Adverb_types_pos
20) Gram2low_pos
21) Adjective_types_pos
22) Gram2high_pos
23) Preposition_types_pos
24) Gram2avg_pos
25) Possessive_pronoun_types_pos
26) Gram3low_pos
27) Personal_pronoun_types_pos
28) Gram3high_pos
29) Particle_types_pos
30) Gram3avg_pos
31) Determiner_types_pos
32) Gram4low_pos
33) Interjection_types_pos
34) Gram4high_pos
35) Nouns_pos
36) Gram4avg_pos

The classification results for applying desired classifiers on Posit + n-gram dataset are listed below Table 6.8, where it can be noticed when using Posit dataset+

N-Gram that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

	70-30 %.			100 %			Unseen Labeled		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**
J48	0.835	0.829	0.831	0.960	0.959	0.959	0.671	0.616	0.634
Naïve-Bayes	0.770	0.583	0.534	0.730	0.577	0.515	0.615	0.314	0.326
KNN k=3	0.834	0.828	0.827	0.901	0.900	0.899	0.725	0.634	0.665
SVM	0.780	0.767	0.764	0.768	0.771	0.767	0.743	0.625	0.658
Rand-Forest	**0.899**	**0.899**	**0.899**	**1.000**	**1.000**	**1.000**	**0.778**	**0.654**	**0.696**
Classification by Regression	0.890	0.886	0.886	0.932	0.930	0.930	0.734	0.661	0.690

Table 6.8: Posit + N-Gram datasets classification results.

Figure 6.2 shows test classification results obtained by applying desired classifiers on Posit + N-gram unseen datasets.

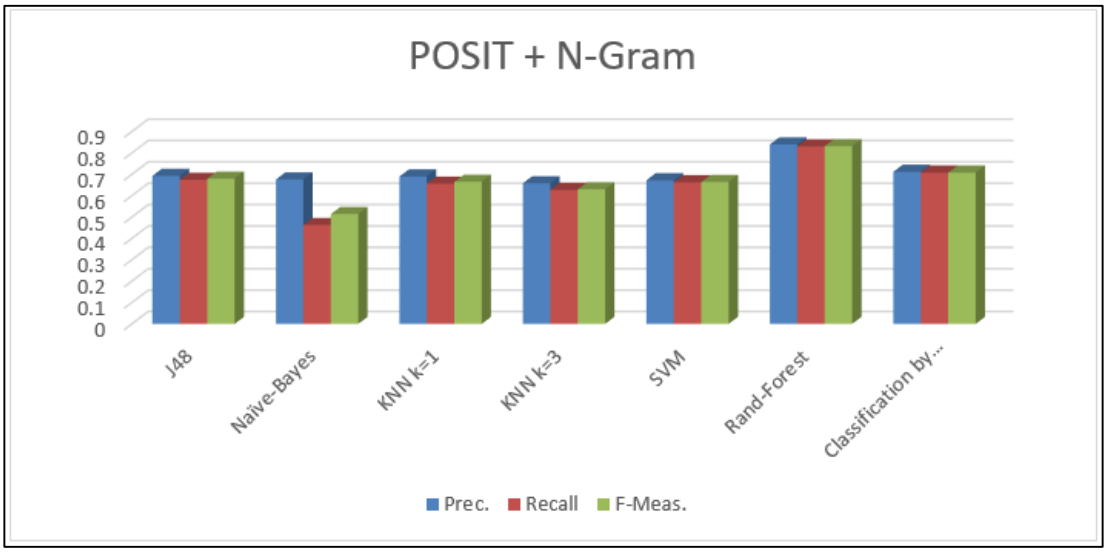


Figure 6. 2: Posit + N-Gram unseen datasets Classification

The classification process for the Posit + 2-3-4 N-Gram summary dataset shows the best results achieved by Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset.

However, Classification by Regression using Random Forest took the second for 70%-30% and 100% datasets and supervised test for the unseen dataset. These results push two classifiers to be the suitable classifiers for Arabic text classification, and they are:

i. Random-Forest.
ii. Classification by Regression using Random Forest.

6.3.3 Results 3 (SAFAR dataset)

In this test, the used ARFF file created depending on summary files resulting from the SAFAR dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	Test dataset
1	70–30% Classification.	safar_70.arff	Seen dataset: safar_30.arff
2	100 % Classification	safar_100.arff	Seen dataset: safar_100.arff
3	Supervised Classification.	3j48.mode 3bayes.mode 3iBk_3.mode 3smo.mode 3randomForest.mode 3randomForestR.mode	Unseen dataset: safar_test_class. Arff

Table 6.9: SAFAR classification datasets

List of attributes (count =31):

1) Number of characters_arab
2) Double Count_arab
3) Total words (tokens)_arab
4) Plural Count_arab
5) Total unique words (types)_arab
6) Masculine Count_arab
7) Number of sentences_arab
8) Feminine Count_arab
9) Type/Token Ratio (TTR)_arab
10) MojarradCount_arab
11) Average sentence length (ASL)_arab
12) MazeedCount_arab
13) Average word length (AWL)_arab
14) LazemCount_arab
15) Verb POS Type Count_arab
16) Mota3adi Count_arab
17) Noun POS Type Count_arab
18) Mota3adi_w_Lazem Count_arab
19) NakiraCount_arab
20) verb_types_arab
21) Ma3rifa Count_arab
22) noun_types_arab
23) ModafCount_arab
24) particle_types_arab
25) Marfo3 Count_arab
26) verbs_arab
27) MansobCount_arab
28) nouns_arab
29) MajrorCount_arab
30) Particl
31) Singular Count_arab

The following table lists the classification results from this experiment.

	70-30 %.			100 %			Unseen Labeled		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**
J48	0.835	0.829	0.831	0.960	0.959	0.959	0.671	0.616	0.634
Naïve-Bayes	0.770	0.583	0.534	0.730	0.577	0.515	0.615	0.314	0.326
KNN k=3	0.834	0.828	0.827	0.901	0.900	0.899	0.725	0.634	0.665
SVM	0.780	0.767	0.764	0.768	0.771	0.767	0.743	0.625	0.658
Rand-Forest	**0.899**	**0.899**	**0.899**	**1.000**	**1.000**	**1.000**	**0.778**	**0.654**	**0.696**
Classification by Regression	0.890	0.886	0.886	0.932	0.930	0.930	0.734	0.661	0.690

Table 6.10: SAFAR datasets classification results

Figure 6.3 shows test classification results obtained by applying desired classifiers on SAFAR unseen datasets, where it can be noticed when using SAFAR dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

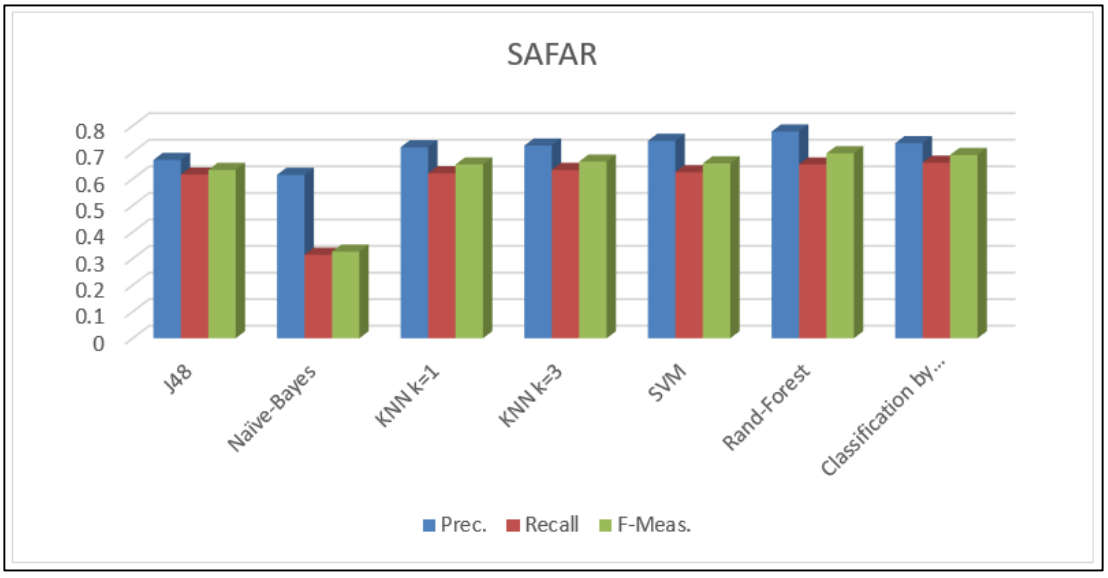


Figure 6. 3: SAFAR unseen datasets Classification

The classification process for the SAFAR summary dataset shows the best result achieved by the Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset.

However, SVM took the second for 70%-30% and 100% datasets and supervised tests for the unseen dataset. These results push two classifiers to be the suitable classifiers for Arabic text classification, and they are:

i. Random Forest.
ii. SVM

6.3.4 Result 4 (SAFAR + N-Gram dataset)

In this test, the used ARFF file created depending on summary files resulting from SAFAR + (SAFAR 2-3-4-Gram) dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	Test dataset
1	70–30% Classification.	safarGram_70.arff	Seen dataset: safarGram_30.arff
2	100 % Classification.	safarGram_100.arff	Seen dataset: safarGram_100.arff
3	Supervised Classification.	4j48.mode 4bayes.mode 4iBk_3.mode 4smo.mode 4randomForest.mode 4randomForestR.mode	Unseen dataset: safarGram_test_class.arff

Table 6.11: SAFAR+N-Gram classification dataset

List of attributes (count = 40):

1) Number of characters_arab
2) MojarradCount_arab
3) Total words (tokens)_arab
4) MazeedCount_arab
5) Total unique words (types)_arab
6) LazemCount_arab
7) Number of sentences_arab
8) Mota3adi Count_arab
9) Type/Token Ratio (TTR)_arab
10) Mota3adi_w_Lazem Count_arab
11) Average sentence length (ASL)_arab
12) verb_types_arab
13) Average word length (AWL)_arab
14) noun_types_arab
15) Verb POS Type Count_arab
16) particle_types_arab
17) Noun POS Type Count_arab
18) verbs_arab
19) NakiraCount_arab
20) nouns_arab
21) Ma3rifa Count_arab
22) particles_arab
23) ModafCount_arab
24) arab_gram2low
25) Marfo3 Count_arab
26) arab_gram2high
27) MansobCount_arab
28) arab_gram2avg
29) MajrorCount_arab
30) arab_gram3low
31) Singular Count_arab
32) arab_gram3high
33) Double Count_arab
34) arab_gram3avg
35) Plural Count_arab
36) arab_gram4low
37) Masculine Count_arab
38) arab_gram4high
39) Feminine Count_arab
40) arab_gram4av

The classification results are listed in the following table:

	70-30 %.			**100 %**			**Unseen Labeled**		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**E-call**	**F-Meas.**
J48	0.944	0.940	0.940	0.980	0.980	0.980	0.673	0.647	0.652
Naïve-Bayes	0.777	0.640	0.626	0.748	0.650	0.632	0.674	0.438	0.495

KNN k=3	0.906	0.903	0.903	0.939	0.937	0.937	0.734	0.657	0.684
SVM	0.906	0.893	0.893	0.905	0.900	0.899	0.717	0.665	0.679
Rand-Forest	0.990	0.990	0.990	1.000	1.000	1.000	0.805	0.771	0.783
Classification by Regression	0.977	0.977	0.977	0.966	0.965	0.965	0.745	0.739	0.742

Table 6.12: SAFAR+ n-gram datasets classification results

Figure 6.4 show test classification results obtained by applying desired classifiers on SAFAR + N-Gram unseen datasets , where it can be noticed when using SAFAR + N-Gram dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

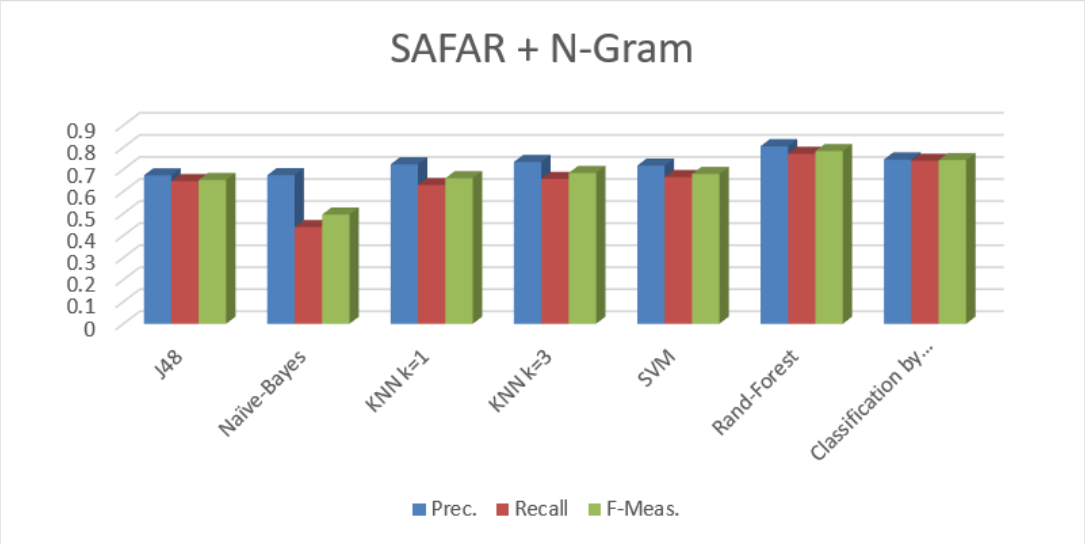


Figure 6. 4: SAFAR + N-Gram datasets supervised classification results

The classification process for SAFAR + N-gram summary dataset shows the best result achieved by Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset.

However, Classification by Regression using Random Forest took the second for 70%-30% and 100% datasets and supervised test for the unseen dataset.

These results present two classifiers as suitable for Arabic text classification, namely:

i. Random Forest

ii. Classification by Regression

6.3.5 Result 5 (Posit+SAFAR dataset)

In this test, the used ARFF file created depending on summary files resulting from Posit+ SAFAR dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	Test dataset
1	70–30% Classification.	total_70.arff	Seen dataset: total_30.arff
2	100 % Classification.	total_100.arff	Seen dataset: total_100.arff
3	Supervised Classification.	51j48.mode 5bayes.mode 5iBk_3.mode 5smo.mode 5randomForest.mode 5randomForestR.mode	Unseen dataset: total_test_class. Arff

Table 6.13: Posit +SAFAR classification dataset

List of attributes (count = 58):

1) Total work (tokens)_pos
2) Total unique words (types)_arab
3) Total unique words (types)_pos
4) Number of sentences_arab
5) Type/Token Ratio (TTR)_pos
6) Type/Token Ratio (TTR)_arab
7) Number of sentences_pos
8) Average sentence length (ASL)_arab
9) Average sentence_pos length (ASL)_os
10) Average word length (AWL)_arab
11) Number of characters_pos
12) Verb POS Type Count_arab
13) Average word length (AWL)_pos
14) Noun POS Type Count_arab
15) Verb_types_pos
16) NakiraCount_arab
17) Noun_types_pos
18) Ma3rifa Count_arab
19) Adverb_types_pos
20) ModafCount_arab
21) Adjective_types_pos
22) Marfo3 Count_arab
23) Preposition_types_pos
24) MansobCount_arab
25) Possessive_pronoun_types_pos
26) MajrorCount_arab
27) Personal_pronoun_types_pos
28) Singular Count_arab
29) Particle_types_pos
30) Double Count_arab
31) Determiner_types_pos
32) Plural Count_arab
33) Interjection_types_pos
34) Masculine Count_arab
35) Nouns_pos
36) Feminine Count_arab
37) Verbs_pos
38) MojarradCount_arab
39) Prepositions_pos
40) MazeedCount_arab
41) Adjectives_pos
42) LazemCount_arab
43) Determiners_pos
44) Mota3adi Count_arab
45) Particles_pos
46) Mota3adi_w_Lazem Count_arab
47) Possessive pronouns_pos
48) Verb_types_arab
49) Personal pronouns_pos
50) Noun_types_arab
51) Adverbs_pos
52) Particle_types_arab
53) Interjection_pos
54) Verbs_arab
55) Number of characters_arab
56) Nouns_arab
57) Total words (tokens)_arab
58) Particles_arab
59)

The following Table 6.14 and Figure 6.5 show the classification results for this experiment, where it can be noticed when using Posit+ SAFAR dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

	70-30 %.			100 %			Unseen Labeled		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**
J48	0.952	0.950	0.950	0.980	0.979	0.979	0.757	0.742	0.745
Naïve-Bayes	0.805	0.611	0.582	0.763	0.612	0.580	0.711	0.380	0.420
KNN k=3	0.942	0.942	0.942	0.958	0.957	0.957	0.728	0.675	0.687
SVM	0.984	0.983	0.983	0.949	0.947	0.947	0.755	0.754	0.754
Rand-Forest	0.993	0.993	0.993	1.000	1.000	1.000	0.872	0.864	0.867
Classification by Regression	0.890	0.886	0.886	0.932	0.930	0.930	0.734	0.661	0.690

Table 6. 14: Posit +SAFAR datasets classification results

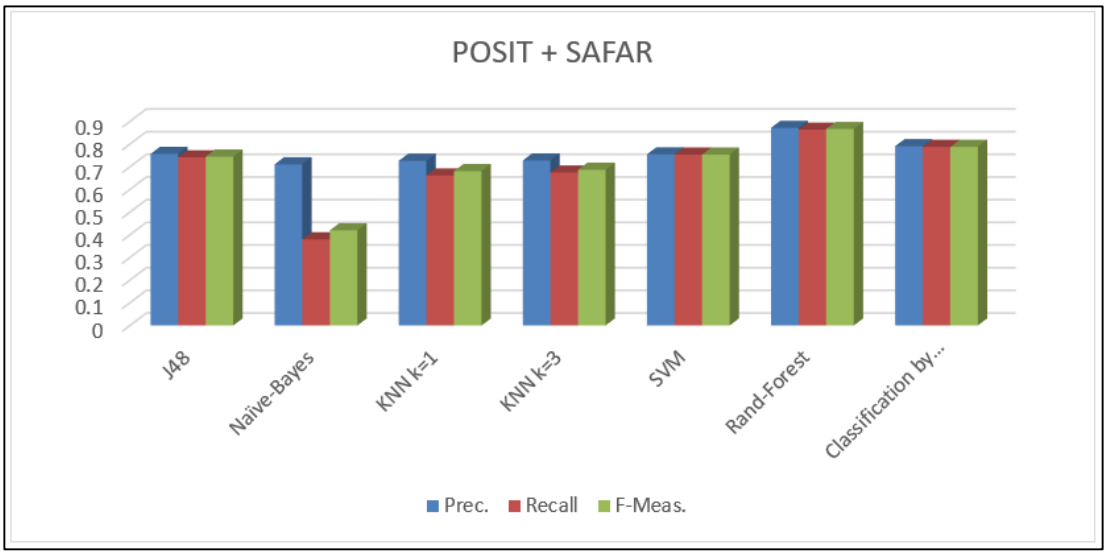


Figure 6. 5: Posit +SAFAR datasets classification results

The classification process for the Posit + SAFAR summary dataset shows the best result achieved by Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset.

However, Classification by Regression using Random Forest took the second for 70%-30% and 100% datasets and supervised test for the unseen dataset. These results push two classifiers to be the suitable classifiers for Arabic text classification, and they are:

i. Random Forest.
ii. Classification by Regression using Random Forest.

Rand Forest continues to maintain the lead with a value less than 90%. Classification by Regression ranks second by a significant margin.

6.3.6 Result 6 (POSIT+SAFAR) + Posit N-Gram dataset

In this test, the used ARFF file created depending on summary files resulting from (Posit+SAFAR) + (Posit 2-3-4-Gram) dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	**Test dataset**
1	70–30% Classification.	total_pGram_70.arff	Seen dataset: total_pGram_30.arff
2	100 % Classification	total_pGram_100.arff	Seen dataset: total_pGram_100.arff
3	Supervised Classification.	6j48.mode 6bayes.mode 6iBk_3.mode 6smo.mode 6randomForest.mode 6randomForestR.mode	Unseen dataset: total_pGram_test_class. Arff

Table 6.15 (Posit+SAFAR) + Posit N-Gram classification dataset

List of attributes (count = 67):

1) Total word (tokens)_pos
2) Verb POS Type Count_arab
3) Total unique words(types)_pos
4) Noun POS Type Count_arab
5) Type/Token Ratio (TTR)_pos
6) NakiraCount_arab
7) Number of sentences_pos
8) Ma3rifa Count_arab
9) Average sentence_pos length (ASL)_pos
10) ModafCount_arab
11) Number of characters_pos
12) Marfo3 Count_arab
13) Average word length (AWL)_pos
14) MansobCount_arab
15) Verb_types_pos
16) MajrorCount_arab
17) Noun_types_pos
18) Singular Count_arab
19) Adverb_types_pos
20) Double Count_arab
21) Adjective_types_pos
22) Plural Count_arab
23) Preposition_types_pos
24) Masculine Count_arab
25) Possessive_pronoun_types_pos
26) Feminine Count_arab
27) Personal_pronoun_types_pos
28) MojarradCount_arab
29) Particle_types_pos
30) MazeedCount_arab
31) Determiner_types_pos
32) LazemCount_arab
33) Interjection_types_pos
34) Mota3adi Count_arab
35) Nouns_pos
36) Mota3adi_w_Lazem Count_arab
37) Verbs_pos
38) Verb_types_arab
39) Prepositions_pos
40) Noun_types_arab
41) Adjectives_pos
42) Particle_types_arab
43) Determiners_pos
44) Verbs_arab
45) Particles_pos
46) Nouns_arab
47) Possessive pronouns_pos
48) Particles_arab
49) Personal pronouns_pos
50) Gram2low_pos
51) Adverbs_pos
52) Gram2high_pos
53) Interjection_pos
54) Gram2avg_pos
55) Number of characters_arab
56) Gram3low_pos
57) Total words (tokens)_arab
58) Gram3high_pos

59) Total unique words (types)_arab
60) Gram3avg_pos
61) Number of sentences_arab
62) Gram4low_pos
63) Type/Token Ratio (TTR)_arab
64) Gram4high_pos
65) Average sentence length (ASL)_arab
66) Gram4avg_pos
67) Average word length (AWL)_arab

The classification results for this experiment are shown in Table 6.16, where it can be noticed when using (Posit+SAFAR) + (Posit 2-3-4-Gram) dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

	70-30 %.			**100 %**			**Unseen Labeled**		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**
J48	0.960	0.960	0.960	0.982	0.982	0.982	0.764	0.752	0.754
Naïve-Bayes	0.805	0.644	0.637	0.759	0.638	0.623	0.691	0.428	0.488
KNN k=3	0.941	0.940	0.940	0.954	0.952	0.952	0.718	0.662	0.679
SVM	0.987	0.987	0.987	0.951	0.949	0.949	0.747	0.745	0.746
Rand-Forest	0.993	0.993	0.993	1.000	1.000	1.000	0.904	0.894	0.896
Classification by Regression	0.977	0.977	0.977	0.973	0.972	0.972	0.797	0.790	0.791

Table 6.16: (Posit+SAFAR) + Posit N-Gram datasets classification results.

Figure 6.6 show test classification results obtained by applying desired classifiers on Posit + SAFAR + (Pos. N-Gram) unseen datasets.

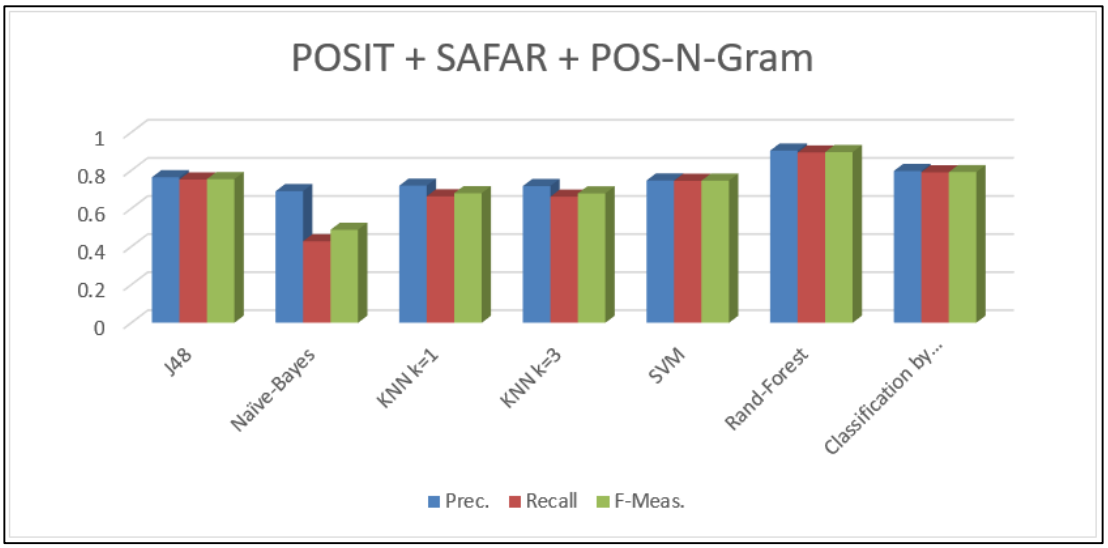


Figure 6. 6: (Posit+SAFAR) + Posit N-Gram datasets classification results

The classification process for (Posit+SAFAR) + Posit N-Gram summary dataset shows the best result achieved by Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset. However, Classification by Regression using the Random Forest took a second place for 70%-30% and 100% datasets and a supervised test for the unseen dataset. These results push two classifiers to be the suitable classifiers for Arabic text classification, which are:

i. Random Forest.
ii. Classification by Regression using Random Forest.

Adding Posit N-Gram attributes to (Posit + SAFAR) attributes enhances precision up to 90%. These results are archived by Random Forest. Classification by Regression precision keeps its values unchanged but holds the second position.

6.3.7 Result 7 (Posit+SAFAR) + SAFAR N-Gram dataset

In this experiment, the used ARFF file created depending on summary files resulting from (Posit+SAFAR) + (SAFAR 2-3-4-Gram) dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	Test dataset
1	70–30% Classification.	total_sGram_70.arff	Seen dataset: total_sGram_30.arff
2	100 % Classification	total_sGram_100.arff	Seen dataset: total_sGram_100.arff
3	Supervised Classification.	6j48.mode 6bayes.mode 6iBk_3.mode 6smo.mode 6randomForest.mode 6randomForestR.mode	Unseen dataset: total_sGram_test_class. Arff

Table 6.17: (Posit+SAFAR) + SAFAR N-Gram classification dataset

List of attributes (count = 67):

1) Total words (tokens)_pos
2) Verb POS Type Count_arab
3) Total unique words (types)_pos
4) Noun POS Type Count_arab
5) Type/Token Ratio (TTR)_pos
6) NakiraCount_arab
7) Number of sentences_pos
8) Ma3rifa Count_arab
9) Average sentence_pos length (ASL)_pos
10) ModafCount_arab
11) Number of characters_pos
12) Marfo3 Count_arab
13) Average word length (AWL)_pos
14) MansobCount_arab
15) Verb_types_pos
16) MajrorCount_arab
17) Noun_types_pos
18) Singular Count_arab
19) Adverb_types_pos
20) Double Count_arab
21) Adjective_types_pos
22) Plural Count_arab
23) Preposition_types_pos
24) Masculine Count_arab
25) Possessive_pronoun_types_pos
26) Feminine Count_arab
27) Personal_pronoun_types_pos
28) MojarradCount_arab
29) Particle_types_pos
30) MazeedCount_arab
31) Determiner_types_pos
32) LazemCount_arab
33) Interjection_types_pos
34) Mota3adi Count_arab
35) Nouns_pos
36) Mota3adi_w_Lazem Count_arab
37) Verbs_pos
38) Verb_types_arab
39) Prepositions_pos
40) Noun_types_arab
41) Adjectives_pos
42) Particle_types_arab
43) Determiners_pos
44) Verbs_arab
45) Particles_pos
46) Nouns_arab
47) Possessive pronouns_pos
48) Particles_arab
49) Personal pronouns_pos
50) Arab_gram2low
51) Adverbs_pos
52) Arab_gram2high
53) Interjection_pos
54) Arab_gram2avg
55) Number of characters_arab
56) Arab_gram3low

57) Total words (tokens)_arab
58) Arab_gram3high
59) Total unique words (types)_arab
60) Arab_gram3avg
61) Number of sentences_arab
62) Arab_gram4low
63) Type/Token Ratio (TTR)_arab
64) Arab_gram4high
65) Average sentence length (ASL)_arab
66) Arab_gram4av
67) Average word length (AWL)_arab

The classification results for this experiment are shown in Table 6.18, where it can be noticed when using (Posit+SAFAR) + (SAFAR 2-3-4-Gram) dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

	70-30 %.			**100 %**			**Unseen Labeled**		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**
J48	0.962	0.962	0.962	0.979	0.979	0.979	0.776	0.768	0.766
Naïve-Bayes	0.799	0.646	0.637	0.764	0.654	0.640	0.692	0.435	0.495
KNN k=3	0.947	0.947	0.946	0.955	0.954	0.954	0.717	0.650	0.667
SVM	0.985	0.984	0.984	0.949	0.947	0.947	0.758	0.756	0.757
Rand-Forest	0.993	0.993	0.993	1.000	1.000	1.000	0.890	0.886	0.887
Classification by Regression	0.968	0.967	0.967	0.972	0.971	0.971	0.786	0.778	0.779

Table 6.18: (Posit+SAFAR) + SAFAR N-Gram datasets classification results

Figure 6.7 show test classification results obtained by applying desired classifiers on Posit + SAFAR (SAFAR N-Gram) unseen datasets.

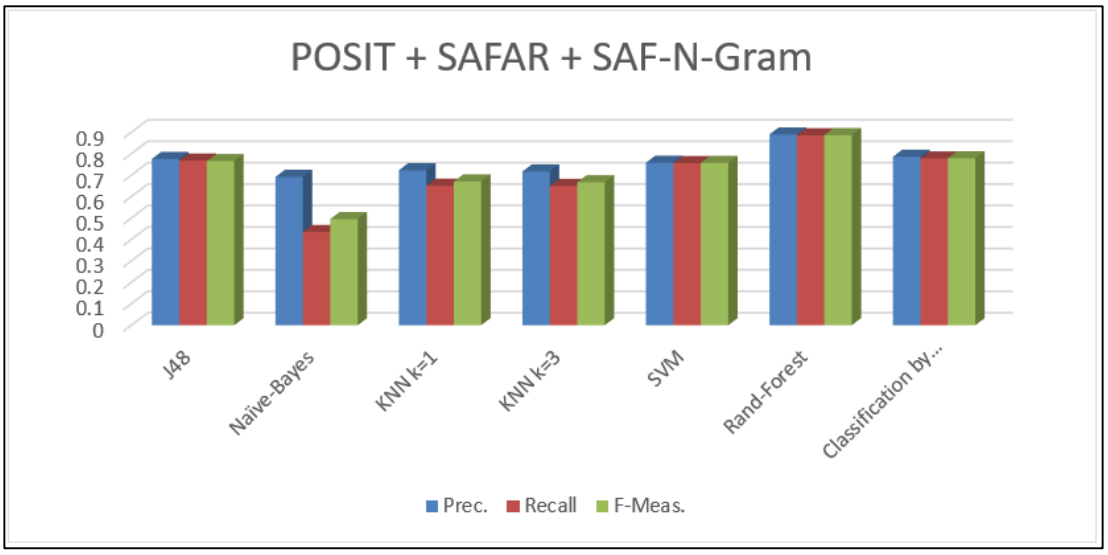


Figure 6. 7: (Posit+SAFAR) + SAFAR N-Gram classification results

The classification process for (Posit+SAFAR) + SAFAR N-Gram summary dataset shows the best result achieved by Random Forest classifier for 70%-30% and 100% datasets and supervised test for the unseen dataset. However, Classification by Regression using Random Forest took the second place for 70%-30% and 100% datasets and supervised test for the unseen dataset. These results push two classifiers to be the suitable classifiers for Arabic text classification, and they are:

i. Random Forest.
ii. Classification by Regression using Random Forest.

The use of the SAFAR N-Gram attributes in addition to (Posit+SAFAR) attributes has led to low precision with each classifier retaining its position. Random-Forest classifier keeps the first position, and classification by Regression stays in the second position.

6.3.8 Result 8 (Posit+SAFAR) + (Posit+SAFAR) N-Gram dataset

In this test, the used ARFF file created depending on summary files resulting from (Posit+SAFAR) + (Posit+SAFAR) N-Gram dataset. Each classification process requires a training dataset and test dataset.

		Training dataset	**Test dataset**
1	70–30% Classification.	total_psGram_70.arff	Seen dataset: total_psGram_30.arff
2	100 % Classification	total_psGram_100.arff	Seen dataset: total_psGram_100.arff
3	Supervised Classification.	6j48.mode 6bayes.mode 6iBk_3.mode 6smo.mode 6randomForest.mode 6randomForestR.mode	Unseen dataset: total_psGram_test_class.arff

Table 6.19: (Posit+SAFAR) + (Posit+SAFAR) N-Gram classification dataset

List of attributes (count = 76):

1) Total words (tokens)_pos
2) ModafCount_arab
3) Total unique words (types)_pos
4) Marfo3 Count_arab
5) Type/Token Ratio (TTR)_pos
6) MansobCount_arab
7) Number of sentences_pos
8) MajrorCount_arab
9) Average sentence_pos length (ASL)_pos
10) Singular Count_arab
11) Number of characters_pos
12) Double Count_arab
13) Average word length (AWL)_pos
14) Plural Count_arab
15) Verb_types_pos
16) Masculine Count_arab
17) Noun_types_pos
18) Feminine Count_arab
19) Adverb_types_pos
20) MojarradCount_arab
21) Adjective_types_pos
22) MazeedCount_arab
23) Preposition_types_pos
24) LazemCount_arab
25) Possessive_pronoun_types_pos
26) Mota3adi Count_arab
27) Personal_pronoun_types_pos
28) Mota3adi_w_Lazem Count_arab
29) Particle_types_pos
30) Verb_types_arab
31) Determiner_types_pos
32) Noun_types_arab
33) Interjection_types_pos
34) Particle_types_arab
35) Nouns_pos
36) Verbs_arab
37) Verbs_pos
38) Nouns_arab

39) Prepositions_pos
40) Particles_arab
41) Adjectives_pos
42) Gram2low_pos
43) Determiners_pos
44) Gram2high_pos
45) Particles_pos
46) Gram2avg_pos
47) Possessive pronouns_pos
48) Gram3low_pos
49) Personal pronouns_pos
50) Gram3high_pos
51) Adverbs_pos
52) Gram3avg_pos
53) Interjection_pos
54) Gram4low_pos
55) Number of characters_arab
56) Gram4high_pos
57) Total words (tokens)_arab
58) Gram4avg_pos
59) Total unique words (types)_arab
60) Arab_gram2low
61) Number of sentences_arab
62) Arab_gram2high
63) Type/Token Ratio(TTR)_arab
64) Arab_gram2avg
65) Average sentence length (ASL)_arab
66) Arab_gram3low
67) Average word length (AWL)_arab
68) Arab_gram3high
69) Verb POS Type Count_arab
70) Arab_gram3avg
71) Noun POS Type Count_arab
72) Arab_gram4low
73) NakiraCount_arab
74) Arab_gram4high
75) Ma3rifa Count_arab
76) Arab_gram4av

Table 6.20 and Figure 6.8 below show the classification results for this test, where it can be noticed when using (Posit+SAFAR) + (Posit+SAFAR) N-Gram dataset that the performance measures were higher in case of the Random Forest classifier, while the least achieved was Naïve-Bayes classifier.

	70-30 %.			**100 %**			**Unseen Labeled**		
	Prec.	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**	**Prec.**	**Recall**	**F-Meas.**
J48	0.949	0.948	0.948	0.982	0.982	0.982	0.763	0.750	0.752
Naïve-Bayes	0.803	0.661	0.658	0.767	0.678	0.672	0.691	0.456	0.514
KNN k=3	0.935	0.934	0.934	0.953	0.952	0.952	0.716	0.651	0.669
SVM	0.982	0.981	0.981	0.950	0.948	0.949	0.771	0.772	0.771
Rand-Forest	0.993	0.993	0.993	1.000	1.000	1.000	0.954	0.953	0.952
Classification by Regression	0.971	0.970	0.970	0.974	0.973	0.973	0.788	0.779	0.780

Table 6. 20: (Posit+SAFAR) + (Posit+SAFAR) N-Gram datasets classification

Figure 6.8 shows test classification results obtained by applying desired classifiers on Posit + SAFAR (Posit + SAFAR N-Gram) unseen datasets.

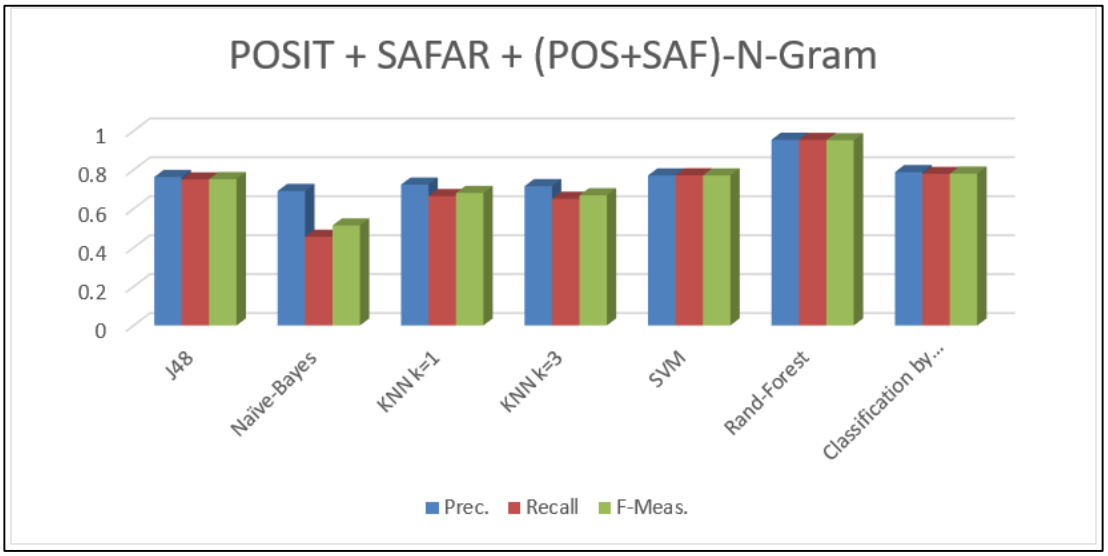


Figure 6. 8: (Posit+SAFAR) + (Posit+SAFAR) N-Gram datasets classification results

The classification process for (Posit+SAFAR) + (Posit+ SAFAR N-Gram) summary dataset shows the best results achieved by Random Forest classifier for 70%-30%, 100%, and Supervised datasets. However, Classification by Regression using Random Forest took the second for 70%-30% and 100% datasets and supervised test for the unseen dataset. Finally, the accuracy exceeded 90% to 95% - such results were reached by the Random-Forest classifier applied on (Posit+SAFAR having 58 attributes) attributes, in addition to (Posit+SAFAR N-Gram having 18 attribute) a total of 76 attributes.

After applying classifiers on different datasets resulting from different toolsets (Posit, SAFAR) either with or without adding N-Gram attributes, all collected results from the different combinations of datasets and these are described in the next section.

6.4 Test Results Summary

A review of unseen dataset test only to summarize the final results for useful information.

6.4.1 Final Posit datasets discussion

Table 6.21 shows Posit dataset classification best results.

Supervised				
		Prec.	Recall	F-Meas.
Posit	Rand-Forest	0.797	0.785	0.786
Posit+ N-Gram	Rand-Forest	0.840	0.831	0.832

Table 6. 21: Posit Classification Results

By adding the N-Gram attributes to the Posit attribute, the Random Forest classifier dominated over other classifiers with considerable value (about 5%) for the supervised dataset.

6.4.2 Final SAFAR dataset discussion

Table 6.22 show Posit dataset classification best results.

Supervised				
		Prec.	Recall	F-Meas.
SAFAR	Rand-Forest	0.778	0.654	0.696
SAFAR + N-Gram	Rand-Forest	0.805	0.771	0.783

Table 6. 22: SAFAR Classification Results

The above results show that less precision of 80% for SAFAR + N-Gram dataset. SAFAR dataset without N-Gram attribute offers lower precision by about 3%.

6.4.3 Posit + SAFAR dataset discussion

Table 6.23 show Posit + SAFAR datasets classification results:

Supervised				
	Classifier	**Prec.**	**Recall**	**F-Meas.**
Posit+SAFAR	Rand-Forest	0.872	0.864	0.867
Posit+SAFAR + (Posit N-Gram)	Rand-Forest	0.904	0.894	0.896
Posit+SAFAR + (SAFAR N-Gram)	Rand-Forest	0.890	0.886	0.887
Posit+SAFAR + (Posit+SAFAR N-Gram)	Rand-Forest	0.954	0.953	0.952

Table 6. 23: Posit+SAFAR best Classification Results

Results show the minimum precision of 87.2% for (Posit+SAFAR) dataset and best precision by using the Posit+SAFAR + (Posit+SAFAR N-Gram) dataset of value 95.4%.

6.4.4 Final Results

Here the best results from all tests and achieved the best performance (Precision = 95%, Recall = 95%, F-Measure = 95%) by applying Random Forest classifier on (Posit+SAFAR) + (Posit+SAFAR) N-Gram. Table 6.24 shows the classifier results sorted in ascending order of performance.

Supervised				
		Prec.	**Recall**	**F-Meas.**
(Posit+SAFAR) + (Posit+SAFAR) N-Gram	Rand-Forest	0.95	0.95	0.95
Posit + SAFAR + (Posit 2-3-4 Gram)	Rand-Forest	0.90	0.89	0.9
Posit + SAFAR + (SAFAR 2-3-4 Gram)	Rand-Forest	0.89	0.89	0.89
Posit + SAFAR	Rand-Forest	0.87	0.86	0.87
Posit + N-Gram	Rand-Forest	0.84	0.83	0.83
SAFAR + N-Gram	Rand-Forest	0.81	0.77	0.78
Posit	Rand-Forest	0.80	0.79	0.79
SAFAR	Rand-Forest	0.78	0.65	0.7

Table 6. 24: All datasets Classification sorted results

6.5 Discussion

Many researchers have tried to obtain the optimum classification algorithm for different languages, especially Arabic. The common toolkit was set up, Posit, to work for the Arabic language. This helped enhance the overall process by finding more than one toolkit to extract meaningful quantum information from Arabic text. The use of N-Gram was another way to process the amount of information used to learn the parameters of attributes and to calculate the prediction of unseen data. WEKA data processing environment is a rich environment for the Random Forest classification algorithm.

Comparison

A similar approach to classification task is reported in the following sections:

1. (Aldayel & Azmi, 2016)

Algorithm used: Hybrid approach classifier for the Arabic language. The approach is based on using lexical classifier for training data for the SVM classifier. Lexical classifier. Used for first step classification to produce the training dataset for model creation. Dataset: 1103 tweets (576 positives, 527 negatives).

Then the SVM classifier was used for classification of an unclassified dataset. The hybrid classifier (Lexical + SVM) produce results as follows:

		Prec.	Recall	F-Meas.	Accuracy
Tweet datasets	**Lexical + SVM**	0.847	0. 838		0. 840

Table 6. 25: Tweet Classification

Our observations:

This research uses a lexical classifier for learning datasets rather than manual classification to apply the SVM classifier to classify the unseen tweets dataset. This combination enhances the overall operation.

$$\textbf{Acc.} = \frac{TP + TN}{TP + TN + FP + FN}$$

Note: Accuracy is determined by:

2. (Shoukry & Rafea, 2012) worked to process tweets to provide their sentiments polarity (positive or negative). SVM and Naïve Bayes (NB) used for both training and classification, one by one. Dataset: 1000 tweets (500 positives, 500 negatives). Results obtained were as follows:

	Classifier	Prec.	Recall	F-Meas.	Accuracy
Tweet datasets (Unigrams and Bigrams)	**SVM**	0.726	0.728	0.726	0.725
Tweet datasets (Unigrams and Bigrams)	**Naïve Bayes**	0.652	0.652	0.652	0.652

Table 6. 26: Sentence Level Arabic Sentiment (1-2 grams) SVM and Naïve Bayes

Our observations:

This research use sentiments classification to produce a learning dataset and supervised test for unseen tweets dataset using two different classification algorithms. SVM gave better results over Naïve Bayes by 7.4%.

3. Its Arabic sentiment considers normalization , stemming, and stop word removal for datasets(during the preprocessing phase) (Shoukry & Rafea, 2012). SVM is used for both training and classification. Dataset: 1000 tweets (500 positives, 500 negative). Results obtained are as follows:

Supervised					
Dataset	**Classifier**	**Accuracy**	**Recall**	**F-Meas.**	**Prec.**
Unigrams - raw tweets	**SVM**	0.740	0.740	0.740	0.740
Unigrams – normalized tweets	**SVM**	0.756	0.756	0.756	0.756
Unigrams + Bigrams + Trigrams stemmed tweets	**SVM**	0.787	0.787	0.787	0.787
Unigrams + Bigrams + Trigrams after stop words removal	**SVM**	0.788	0.788	0.788	0.788

Table 6. 27: Sentence Level Arabic Sentiment (2-3 grams) SVM (normalization, stemming, and stop words removal)

Our observations:

Adding 2-3-4 Gram information enriches the training and test datasets. Preprocessing enhances the classification process by different factors.

4. (El-Halees, 2015)

Algorithm used: combined approach for Arabic language classification in the beginning, the lexicon-based method is used to classify as many documents as possible. The resultant classified documents are used as a training set for the maximum entropy method, which subsequently classifies some other documents. Finally, the k-nearest method used the classified documents from the lexicon-based method and maximum entropy as a training set and classified the rest of the documents (El-Halees, 2015)

Dataset: 949 tweets (415 positives, 534 negatives) belong to "education", "politics" and "sports" categories. Results collected as follows:

	Lexical	Lexical + ME	Lexical +ME + kNN
Accuracy	50.08	60.73	80.29

Dataset	**Classifier**	**Prec.**	**Recall**	**F-Meas.**
Politics Sports Education datasets	Lexical + Maximum Entropy + k-nearest	80.7	79.805	79.895

Table 6. 28: combined approach for Arabic language classification

Our observations:

A combined classification (Lexical + Maximum Entropy + k-nearest) approach enhances classifier accuracy.

Observing the last 4 types of research, the researcher is going forward for the Arabic classification process, which is considered to be an NP-complete problem (nondeterministic polynomial time) [https://www.ics.uci.edu/~eppstein/161/960312.html].

The overview shows that the researcher's results reach an acceptably high level of precision by using different ways of data preprocessing thereby enriching the input data by adding N-Gram or classifying by multiple classifiers. In the following Table 6.28 a comparison of results to reviewed researches results.

			Prec.	Recall	F-Meas.	Accuracy
1	**Tweet datasets**	**Lexical + SVM**	0.847	0. 838		0. 840
2	**Tweet datasets (Unigrams and Bigrams)**	**SVM**	0.726	0.728	0.726	0.725
	Tweet datasets (Unigrams and Bigrams)	**Naïve Bayes**	0.652	0.652	0.652	0.652
3	**Unigrams - raw tweets**	**SVM**	0.740	0.740	0.740	0.740
	Unigrams – normalized tweets	**SVM**	0.756	0.756	0.756	0.756
	Unigrams + Bigrams + Trigrams stemmed tweets	**SVM**	0.787	0.787	0.787	0.787
	Unigrams + Bigrams + Trigrams after stop words removal	**SVM**	0.788	0.788	0.788	0.788
4	**Politics Sports Education datasets**	**Lexical + Maximum Entropy + k-nearest**	80.7	79.805	79.895	
5	**(POSIT+SAFAR) + (POSIT+SAFAR) N-Gram**	**Rand-Forest**	0.95	0.95	0.95	

Table 6. 29: comparing other classification results to our classification methodology

Our approach depends on manual classification for the training dataset (70% + 30% seen dataset) to ensure the best results. Note that the process of manual classification is time consuming, especially if it is carried out on several thousand text files. This is also influenced by the scientific level and culture of those involved in the process of manual classification. After manual classification, the text-processing toolkit was applied in order to build datasets for the training and classification process. Attribute data is extracted by two different toolkits (Posit & SAFAR), which build information obtained from text files.

Unseen Datasets:

The Random Forest algorithm used for creating a classification model employing a carefully and manually classified dataset gives us the best results over other classification techniques.

	Algorithm	Dataset	Prec.	Recall	F-Meas.
1	**Random Forest**	**Posit + SAFAR + (Posit + SAFAR N-Gram)**	**0.95**	**0.953**	**0.952**
2	RF Via Regression	Posit+ SAFAR + (Posit N-Gram)	0.80	0.79	0.791
3	J48	Posit+ SAFAR + (SAFAR N-Gram)	0.78	0.768	0.766
4	SVM	Posit + SAFAR + (Posit + SAFAR N-Gram)	0.77	0.771	0.771
5	IBk_3	SAFAR + N-Gram	0.73	0.657	0.684
6	Naïve Bayes	Posit + SAFAR	0.71	0.38	0.42

Table 6. 30: Random Forest classifier against some other classifiers

The Random Forest algorithm gave the best result against the manually classified dataset (Posit + SAFAR toolkits) and other algorithms with a precision of 0.95. Other algorithms (RF via Regression, J48, SVM, IBk_3, Naïve Bayes) show good results with different datasets, all undergoing manual classification with results of (0.71-0.80).

6.6 Summary

Our discussion shows that using a full details dataset adding all attributes extracted by both Posit and SAFAR techniques and 2-3-4-gram attributes help improve the prediction of an unseen dataset using supervised classification Random Forest classifier. The quantitative approach gave very surprising and impressive results in the classification.

CHAPTER SEVEN: CONCLUSION AND FUTURE WORK

7.1 Conclusion

The impact of sites that encourage terrorism can be minimized, whether they directly express their intentions or indirectly promote extremist ideas to influence those with moderate views or extreme tendencies. Detecting the intentions of these sites cannot be done manually, there are millions of pages that contain articles, comments, and recorded reactions, that would make the task impossible.

Therefore, we have developed a mechanism to predict the message embedded into texts written in Arabic. In this research, we have discussed most of the algorithms used for text processing and have found the best way of predicting anti, neutral, and pro-terrorism sites. Those algorithms can be used to identify the classification of text files collected from different sites to determine whether they are encouraging terrorism or condemning it.

In this research, we have tested several methods of classification to find the best algorithm to classify text files and determine whether they encourage terrorism or not. The first step to undertake file classification is the manual classification process. We have tested the idea of classification based on the overall message of the text .This process leads to fewer predictions since all the files come from different ideologies and are written by different writers. In truth, files have been found to contain a mix of words and statements used, in different contexts, by both sides, anti and pro-terrorism.

a. The steps used to reach optimal results are as follows:

    1. First: Collect text information from different sites with different ideas (Pro – Anti - Neutral).

    2. Second: Use different text processing software (machine learning) to extract text information from the collected text.

b. To do so, we have used two different text processing toolsets:

1. Posit Profiling Toolset: we have used POS for text analysis; for this reason, we have made some modifications to suit the uniqueness of the Arabic language. The Posit toolset targets three related aspects of textual analysis, comprising individual software modules whose operations may be combined.

   i. The first of these modules focuses on parts-of-speech (POS) and performs analysis of a given text corpus to derive statistics on the POS characteristics of that text. This component is known as the POS Profiler.

   ii. The second module of the toolset is the Vocabulary Profiler. Based upon the statistical data output by the POS Profiler, the Vocabulary Profiler can determine the relative frequency of occurrence for vocabulary items in the selected corpus. This frequency data may be compared to a reference set of frequency data (derived from the British National Corpus) to pinpoint unusual word occurrences or individual terms whose use is likely to prove unfamiliar to English readers.

   iii. The third toolset module (presently under development) is the Readability Profiler. This software component will focus on text readability, based upon the statistical analyses from the POS profiler and the frequency data from the vocabulary profiler. In keeping with our research in this area, this module will go beyond the current 'simplistic' readability metrics. It will apply more sophisticated analyses that include factors such as word commonality (Weir & Ritchie, 2007) and average collocation frequency.

2. SAFAR (Software architecture for Arabic language processing): this is the second software toolset that suites the Arabic language.

SAFAR is an integrated platform dedicated to ANLP written in Java. It is cross-platform, modular, extensible, and provides an integrated development environment (IDE). SAFAR brings together all layers of Arabic Natural Language Processing (Jaafar & Bouzoubaa, 2015).

The result from machine learning done by Posit and SAFAR toolsets are a set of files containing the information extracted from the raw text files. The most important parts of the text processing toolsets are Summary and N-Gram files. These two files are used for text mining to build recognition patterns that result from applying a classification algorithm.

The manual classification process is done on the raw text data before Posit/SAFAR processing.

The manual classification is done based on the file content, i.e., based on quantities, side by side, with phrasal combinations (N-Gram attributes). However, we have found that it is not right to use either a quantitative basis by itself or phrasal combinations only. To achieve the best prediction precision results, we have to manually classify the files based on both of these approaches combined.

After manual classification and getting summary files by Posit and SAFAR toolsets, we tested the various algorithms. J48, Naïve-Bayes, KNN k=1, KNN k=3, SVM, Rand-Forest and Classification by Regression are considered for text classification. The research progress is made by applying a classifier on one of the summary datasets.

The Posit only datasets show better precision than SAFAR datasets when using K-Nearest Neighbors, where K= 3. Adding N-Gram attributes associated with each of the datasets (Posit/SAFAR) enhances the SAFAR precision using the Random Forest classifier but has not helped to improve the Posit classification process. Results enhancement is achieved by adding the Posit attributes (27 attributes) to the SAFAR attribute (33 attributes) as one bigger dataset. Adding Posit N-Gram has yielded better results than those obtained by adding SAFAR N-Gram.

Best results reaching 94% precision have been achieved by combining Posit + SAFAR + (18 attributes Posit+ SAFAR N-Gram), a total of 76 attributes. Random Forest excels over all other classifiers during all classification processes, except for Posit only dataset.

- Random Forest classifier parameters are set as follows:

- Bag Size Percent (100);
- Batch Size ("100");
- Break Ties Randomly (true);
- Calc. out Of Bag (false);
- Compute Attribute Importance (false);
- Debug (false);
- Do Not Check Capabilities (false);
- Max Depth (10);
- Num. Decimal Places (2);
- Num. Execution Slots (1);
- Num. Features (0);
- Num. Iterations (25);
- Output out Of Bag Complexity Statistics (false);
- Print Classifiers (true) and
- Seed (1);

The final result reached for classification by regression for the:

1. Summary files created by the Posit toolset (27 attributes).
2. 2, 3, and 4 N-Gram average values produced by Posit toolset (9 attributes).
3. Summary files created by the SAFAR toolset (31 attributes).
4. 2, 3, and 4 N-Gram average values produced by SAFAR toolset (9 attributes).

By using this amount of information provided by Posit , side by side, with SAFAR toolsets (58 attribute) combined with 18 attributes of 2,3 and 4 N-Gram produced by Posit, side by side, with SAFAR toolsets as input for machine learning, we have obtained the best results by applying Random Forest classifier.

7.2 Research Answers

Our research question was mentioned in Section 1.2. The answer for this main question can be broken down to the following:

c. “Can we label different occurrences of the same item (for example, word or

phrase) under various contexts (Pro extremist – Anti extremist –neutral)?"

d. "Can we train robust and discriminative features to uncover any misleading emotional Arabic text, and properly categorize said text according to its underlying message, within a quantitative approach?"

The answer showed the best way for file classification. We found that we can improve and automate the classification process using Random Forest and Random Forest via the Regression classification algorithm which is integrated into JAVA application using the WEKA machine learning environment (WEKA API). The used datasets for unseen data classification are different combinations of data extracted using Posit and SAFAR toolkits and N-Gram attributes.

The answer for this main question can be broken down to the following :

a. Items in text writing (for example, word or phrase) can be labeled under various tags (Pro extremist – Anti extremist –neutral). This makes it hard to distinguish between different classes of context using the automated classification system. The nature of the text makes it difficult reaching the maximum prediction that is equal to one, but it reduces as much as the uncertainty of determining the item class exists.

b. It has been shown from our practical experience that combining different attributes deduced by combined two toolkits for analyzing Arabic text can be used to enhance text categorization using a sufficient set of carefully manually classified files.

To achieve the best results, we have combined two sets of tools known for their ability to analyze the Arabic text or use custom tools for Latin languages after changing the configuration files to be able to recognize the Arabic text. This contribution helps to enhance Arabic text classification.

7.3 Future Work

Future work is based on the increasing diversity of a collection of data from different site categories (sports – politics – social – food – health, etc.) to get alternative ways of writing, also to overcome the lack of sites supporting terrorism, or

having extremist tendencies web sites by getting support from the relevant authorities to provide archival copies of such sites. This would help produce a better model for classification.

Find an algorithm that can use in conjunction with manual sorting to reduce the effort and time required for manual classification. Using techniques like attribute selection will have a better performance especially with datasets with larger n-gram data.

Other future work is to automate the classification process using our produced model and other models for multi-language websites including social media sites, and to propose accepted datasets enhancing the model by re-training to produce a new model file for future use.

REFERENCES

Agarwal, S., & Sureka, A. (2015). Applying social media intelligence for predicting and identifying on-line radicalization and civil unrest oriented threats. arXiv preprint arXiv:1511.06858.

Al-Shalabi, R., Kanaan, G., & Gharaibeh, M. (2006). Arabic text categorization using KNN algorithm. Paper presented at the the Proc. of Int. multi conf. on computer science and information technology CSIT06.

Al-Zoghby, A., Eldin, A. S., Ismail, N. A., & Hamza, T. (2007). Mining Arabic text using soft-matching association rules. Paper presented at the 2007 International Conference on Computer Engineering & Systems.

Aldayel, H. K., & Azmi, A. M. (2016). Arabic tweets sentiment analysis–a hybrid scheme. Journal of Information Science, 42(6), 782-797.

AlGahtani, S., Black, W., & McNaught, J. (2009). Arabic part-of-speech tagging using transformation-based learning. Paper presented at the Proceeedings of the 2nd International Conference on Arabic Language Resources and Tools.

Alhawarat, M., Hegazi, M., & Hilal, A. (2015). Processing the text of the Holy Quran: a text mining study. International Journal of Advanced Computer Science and Applications (IJACSA), 6(2), 262-267.

Alvari, H., Sarkar, S., & Shakarian, P. (2019). Detection of Violent Extremists in Social Media. arXiv preprint arXiv:1902.01577.

Ayadi, R., Maraoui, M., & Zrigui, M. (2016). A Survey of Arabic Text Representation and Classification Methods. Research in Computing Science, 117(12), 51-62.

Af-Wåhlberg, A. (2017). Driver behaviour and accident research methodology: unresolved problems. CRC Press.

Anagnostou, N.K. & Weir, G.R.S. (2007). Average Collocation Frequency as an Indicator of Semantic Complexity. Proceedings of ICTATLL 2007,

Aston, G., Bernardini, S. & Stewart, D. (2004). Corpora and Language Learners. John Benjamins, Amsterdam.

Boukhatem, N. (2014). The Arabic natural language processing: Introduction and challenges. International Journal of English Language & Translation Studies, 2(3), 106-112.

Burnes, & Rouse. (2018). AWS analytics tools help make sense of big data. from https://searchenterpriseai.techtarget.com/definition/machine-learning-ML

Broder, Andrei Z.; Glassman, Steven C.; Manasse, Mark S.; Zweig, Geoffrey (1997). "Syntactic clustering of the web". Computer Networks and ISDN Systems. 29 (8): 1157–1166.

Bird, S. (2006). NLTK: the natural language toolkit. In Proceedings of the COLING/ACL on interactive Presentation Sessions. Association for Computational Linguistics, Morristown, NJ, 69-72.

Bin Hassan, M. H., & Pereire, K. G. (2006). An ideological response to combating terrorism–The Singapore perspective. *Small Wars and Insurgencies*, *17*(4), 458-477.

Clandinin, D. J., Cave, M. T., & Berendonk, C. (2017). Narrative inquiry: a relational research methodology for medical education. Medical Education, 51(1), 89-96.

Ceci, M., Hollmén, J., Todorovski, L., Vens, C., & Džeroski, S. (2017). Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings (Vol. 10535): Springer.

Cherif, W., Madani, A., & Kissi, M. (2015). A new modeling approach for Arabic opinion mining recognition. Paper presented at the 2015 Intelligent Systems and Computer Vision (ISCV).

Dey, A. (2016). Machine learning algorithms: a review. International Journal of Computer Science and Information Technologies, 7(3), 1174-1179.

Diab, M., Hacioglu, K., & Jurafsky, D. (2004). Automatic tagging of Arabic text: From raw text to base phrase chunks. Paper presented at the Proceedings of HLT-NAACL 2004: Short papers.

Diffen. (2019). HDD vs. SSD. from https://www.diffen.com/difference/HDD_vs_SSD

Dunham, M. H. (2006). Data mining: Introductory and advanced topics: Pearson Education India.

Duwairi, R. M., & Qarqaz, I. (2014). Arabic sentiment analysis using supervised classification. Paper presented at the 2014 International Conference on Future Internet of Things and Cloud.

El-Halees, A. M. (2015). Arabic text classification using maximum entropy. IUG Journal of Natural Studies, 15(1).

El Kourdi, M., Bensaid, A., & Rachidi, T.-e. (2004). Automatic Arabic document categorization based on the Naïve Bayes algorithm. Paper presented at the proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages.

El Zein, H. (2015). Is the internet the main medium of transmitting terrorism? Paper presented at the 16th Australian Information Warfare Conference.

Fletcher, A. J. (2017). Applying critical realism in qualitative research: methodology meets method. International journal of social research methodology, 20(2), 181-194.

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In IJcAI (Vol. 7, pp. 1606-1611).

Goddard, C. (2011). Semantic analysis: A practical introduction. Oxford University Press.

Hamzah,S, Mohamed (2012) .Combating terrorism and extremism, and the method of intellectual review.Ministry of the Interior of the Arab Republic of Egypt.

Habash. (2010). Introduction to Arabic natural language processing. Synthesis Lectures on Human Language Technologies, 3(1), 1-187.

Habash, N., & Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. Paper presented at the Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05).

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.

Hofmann, T. (2013). Probabilistic latent semantic analysis. arXiv preprint arXiv:1301.6705.

Jaafar, Y., & Bouzoubaa, K. (2015). Arabic natural language processing from software engineering to complex pipeline. Paper presented at the 2015 First International Conference on Arabic Computational Linguistics (ACLing).

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Paper presented at the European conference on machine learning.

Johansson, F., Kaati, L., & Sahlgren, M. (2017). Detecting linguistic markers of violent extremism in online environments Artificial Intelligence: Concepts, Methodologies, Tools, and Applications (pp. 2847-2863): IGI Global.

Keene, S. D. (2011). Terrorism and the internet: A double-edged sword. *Journal of Money Laundering Control*.

Kaur, G., & Chhabra, A. (2014). Improved J48 classification algorithm for the prediction of diabetes. International Journal of Computer Applications, 98(22).

Kirchner, M. (2010). A framework for detecting anomalies in http traffic using instance-based learning and k-nearest neighbor classification. Paper presented at the 2010 2nd International Workshop on Security and Communication Networks (IWSCN).

Kübler, S., & Mohamed, E. (2012). Part of speech tagging for Arabic. Natural Language Engineering, 18(4), 521-548.

Kumar, R. (2019). Research methodology: A step-by-step guide for beginners. Sage Publications Limited.

Ledford, J. R., & Gast, D. L. (2018). Single case research methodology: Applications in special education and behavioral sciences. Routledge.

Li, Y.-M., & Li, T.-Y. (2013). Deriving market intelligence from microblogs. Decision Support Systems, 55(1), 206-217.

Maamouri, M., & Bies, A. (2004). Developing an Arabic treebank: Methods, guidelines, procedures, and tools. Paper presented at the Proceedings of the Workshop on Computational Approaches to Arabic Script-based languages.

Marie-Sainte, S. L., Alalyani, N., Alotaibi, S., Ghouzali, S., & Abunadi, I. (2018). Arabic natural language processing and machine learning-based systems. IEEE Access, 7, 7011-7020.

Mavengere, N. B. (2013). Information technology role in supply chain's strategic agility. International Journal of Agile Systems and Management, 6(1), 7-24.

Mesleh. (2007). Chi square feature extraction based svms arabic language text categorization system. Journal of Computer Science, 3(6), 430-435.

Mesleh. (2011). Feature sub-set selection metrics for Arabic text classification. Pattern Recognition Letters, 32(14), 1922-1929.

Moraes, R., Valiati, J. F., & Neto, W. P. G. (2013). Document-level sentiment classification: An empirical comparison between SVM and ANN. Expert Systems with Applications, 40(2), 621-633.

Muhammad, A. B. (2012). Annotation of conceptual co-reference and text mining the Qur'an: University of Leeds.

Mysql (Producer). (2019). How MySQL Uses Indexes. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/mysql-indexes.html

Nahar, K. M., Al Eroud, A. F., Barahoush, M., & Al-Akhras, A. M. (2019). SAP: Standard Arabic Profiling Toolset for Textual Analysis. International Journal of Machine Learning and Computing, 9(2).

Ogun, M. N. (2012). Terrorist use of internet: possible suggestions to prevent the usage for terrorist purposes. *Journal of Applied Security Research*, *7*(2), 203-217.

Orngreen, R., & Levinsen, K. (2017). Workshops as a Research Methodology. Electronic Journal of E-learning, 15(1), 70-81.

Oakes. (2019). from www.wlv.ac.uk/~in4326/old/2003_Oakes_RL.pdf

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1-135.

Patil, T. R., & Sherekar, S. (2013). Performance analysis of Naive Bayes and J48 classification algorithm for data classification. International journal of computer science and applications, 6(2), 256-261.

Paynter, Trigg, Frank, & Kirkby (Producer). (2008). Attribute-relation file format (ARFF). Retrieved from http://www. cs. waikato. ac. nz/ml/weka/arff. html

RAMDisk. (2019). RAMDisk. from http://memory.dataram.com/products-and-services/software/ramdisk

Rouse. (2019). AWS analytics tools help make sense of big data

from https://searchsqlserver.techtarget.com/definition/data-mining

Salloum, AlHamad, Al-Emran, & Shaalan. (2018). A survey of Arabic text mining Intelligent Natural Language Processing: Trends and Applications (pp. 417-431): Springer.

Salloum, W., & Habash, N. (2014). ADAM: Analyzer for dialectal Arabic morphology. Journal of King Saud University-Computer and Information Sciences, 26(4), 372-378.

Sawaf, H., Zaplo, J., & Ney, H. (2001). Statistical classification methods for Arabic news articles. Natural Language Processing in ACL2001, Toulouse, France.

Sayad. (2017). Multiple Linear Regression. from https://www.saedsayad.com/mlr.htm

Schutt, R., & O'Neil, C. (2013). Doing data science: Straight talk from the frontline: O'Reilly Media, Inc.

Scrivens, R., Davies, G., & Frank, R. (2018). Searching for signs of extremism on the web: an introduction to Sentiment-based Identification of Radical Authors. Behavioral sciences of terrorism and political aggression, 10(1), 39-59.

Sharaf, A. (2009). The Qur'an annotation for text mining. First year transfer report. School of Computing, Leeds University. December.

Shoukry, A., & Rafea, A. (2012). Sentence-level Arabic sentiment analysis. Paper presented at the 2012 International Conference on Collaboration Technologies and Systems (CTS).

Soman, T., & Bobbie, P. O. (2005). Classification of arrhythmia using machine learning techniques. WSEAS Transactions on computers, 4(6), 548-552.

SQLite (Producer). (2019). The CREATE INDEX Command. Retrieved from https://www.tutorialspoint.com/sqlite/sqlite_indexes.htm

Suthaharan, S. (2016). Machine learning models and algorithms for big data classification. Integr. Ser. Inf. Syst, 36, 1-12.

Van den Bosch, A., Marsi, E., & Soudi, A. (2007). Memory-based morphological analysis and part-of-speech tagging of Arabic Arabic Computational Morphology (pp. 201-217): Springer.

Weir. (2007). The Posit text profiling toolset.

Weir, & Anagnostou. (2007). Exploring newspapers: a case study in corpus analysis. Paper presented at the Proceedings of ICTATLL 2007.

Weir, G. R., Dos Santos, E., Cartwright, B., & Frank, R. (2016). Positing the problem: enhancing classification of extremist web content through textual analysis. Paper presented at the 2016 IEEE International Conference on Cybercrime and Computer Forensic (ICCCF).

Weir, G. R., & Ritchie, C. (2006). Estimating readability with the Strathclyde readability measure. Paper presented at the ICT in the Analysis, Teaching and Learning of Languages, Preprints of the ICTATLL Workshop 2006.

Wiek, A., & Lang, D. J. (2016). Transformational sustainability research methodology. In Sustainability science (pp. 31-41). Springer, Dordrecht.

Appendix A:

Software:

-Computer software, or simply software, is a collection of data or computer instructions that tell the computer how to work. This is in contrast to physical hardware, from which the system is built and actually performs the work. In computer science and software engineering, computer software is all information processed by computer systems, programs and data. Computer software includes computer programs, libraries and related non-executable data, such as online documentation or digital media.

Security:

Security is freedom from, or resilience against, potential harm caused by others. Beneficiaries (technically referents) of security may be of persons and social groups, objects and institutions, ecosystems or any other entity or phenomenon vulnerable to unwanted change. Security mostly refers to protection from hostile forces.

Extremist:

A person who holds extreme political or religious views, especially one who advocates illegal, violent, or other extreme action. Extremist is a person who goes to extremes, especially in political matters. A supporter or advocate of extreme doctrines or practices.

Neutral:

Commonly used to describe a person who doesn't pick sides in disputes, neutral also pegs anything that refuses to be bold. If a person or country adopts a neutral position or remains neutral, they do not support anyone in a disagreement, war, or contest. Neutral also means that not taking part or giving assistance in a dispute or war between others.

Pro Extremist

UNIX:

UNIX is a family of multitasking, multiuser computer operating systems that derive from the original AT&T UNIX, development starting in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others. Initially intended for use inside the Bell System, AT&T licensed UNIX to outside parties in the late 1970s, leading to a variety of both academic and commercial UNIX variants from vendors including University of California, Berkeley (BSD), Microsoft (Xenix), IBM (AIX), and Sun Microsystems (Solaris).

POS: Part of speech:

Part-of-speech tagging (POS tagging), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

Tagger:

A piece of software that labels words in a document according to their grammatical and syntactic role.

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of-speech taggers described in these papers.

Stanford Tagger:

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of-speech taggers described in these papers. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. The tagger was originally written by Kristina Toutanova.

Classification:

Classification is a process related to categorization, the process in which ideas and objects are recognized, differentiated and understood. It is a systematic arrangement in groups or categories according to established criteria.

WEKA: Waikato Environment for Knowledge Analysis

WEKA is tried and tested open source machine learning software that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It is widely used for teaching, research, and industrial applications, contains a plethora of built-in tools for standard machine learning tasks, and additionally gives transparent access to well-known toolboxes.

GUI: Graphical user interface

The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), which require commands to be typed on a computer keyboard.

HDD: Hard disk drive

A hard disk drive is an electro-mechanical data storage device that uses magnetic storage to store and retrieve digital information using one or more rigid rapidly rotating disks (platters) coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a moving actuator arm, which read and write data to the platter surfaces. Data is accessed in a random-access manner, meaning that individual blocks of data can be stored or retrieved in any

order and not only sequentially. HDDs are a type of non-volatile storage, retaining stored data even when powered off.

SSD: Solid-state drive

A solid-state drive (SSD) is a solid-state storage device that uses integrated circuit assemblies to store data persistently, typically using flash memory, and functioning as secondary storage in the hierarchy of computer storage. It is also sometimes called a solid-state device or a solid-state disk, although SSDs lack the physical spinning disks and movable read-write heads used in hard drives ("HDD") or floppy disks.

ISIS: Intel System Implementation Supervisor

ISIS is an operating system for early Intel microprocessors like the 8080. It was originally developed by Ken Burgett under the new management of Bill Davidow for the Intel Microprocessor Development System starting in 1975, and later adopted as ISIS-II for systems with floppy drives.

Twitter:

Twitter is an American microblogging and social networking service on which users post and interact with messages known as "tweets". Registered users can post, like, and retweet tweets, but unregistered users can only read them. Users access Twitter through its website interface, through Short Message Service (SMS) or its mobile-device application software ("app").

Social media:

Social media is interactive computer-mediated technologies that facilitate the creation or sharing of information, ideas, career interests and other forms of expression via virtual communities and networks. Social media is the collective of online communications channels dedicated to community-based input, interaction, content-sharing and collaboration. Websites and applications dedicated to forums, microblogging, social networking, social bookmarking, and wikis are among the different types of social media.

Appendix B:

Initialization Variables

Glob.java

```
package init;

import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.Add;
import weka.filters.unsupervised.attribute.Remove;

public class Glob {

//	public final static int learnCount = 210;
//	public final static int testCount = 90;

//	public final static int learnCount = 490;
//	public final static int testCount = 210;

	public final static int learnCount = 700;
	public final static int testCount = 300;
	public final static int classCount = 50000;

	public final static String driveC = "G:\\weka_project\\";

	public final static String CONNECTION_STRING = "jdbc:sqlite:" + driveC + "ngramDB.db";

	public final static String summPathPos = driveC + "posData\\summary_pos\\";
	public final static String summPathSafar = driveC + "safarData\\summary_safar\\";

	////////////////////////////////////////////////////////////////////////////////////////////////////
///////
	public static String userTraining = "";
	public static String userTest = "";

	// Module Data Set
	public static Instances moduleInsts;

	// Classification Data Set
	public static Instances classificationInsts;

	// Training Data Set
	public static Instances trainInsts;

	// Self Test Data Set
```

```
	public static Instances selfTestInsts;

	// Test Data Set
	public static Instances testInsts;

	public static void init(Integer executionIndex) {
		try {
			String strTrain = "", strTest = "", strSelfTest = "", strModule = "",
strClassification = "";
			switch (executionIndex) {
			case 1:
				strTrain = "1-PosTraining.arff";
				strSelfTest = "1-PosTraining_t.arff";
				strModule = "1-PosTraining_m.arff";
				strClassification = "1-PosTraining_c.arff";
				strTest = "2-PosTest.arff";
				break;
			case 2:
				strTrain = "3-PosNgram.arff";
				strSelfTest = "3-PosNgram_t.arff";
				strModule = "3-PosNgram_m.arff";
				strClassification = "3-PosNgram_c.arff";
				strTest = "4-PosNgramTest.arff";
				break;
			case 3:
				strTrain = "5-SafarTraining.arff";
				strSelfTest = "5-SafarTraining_t.arff";
				strModule = "5-SafarTraining_m.arff";
				strClassification = "5-SafarTraining_c.arff";
				strTest = "6-SafarTest.arff";
				break;
			case 4:
				strTrain = "7-SafarNgram.arff";
				strSelfTest = "7-SafarNgram_t.arff";
				strModule = "7-SafarNgram_m.arff";
				strClassification = "7-SafarNgram_c.arff";
				strTest = "8-SafarNgramTest.arff";
				break;
			case 5:
				strTrain = "9-TotalTraining.arff";
				strSelfTest = "9-TotalTraining_t.arff";
				strModule = "9-TotalTraining_m.arff";
				strClassification = "9-TotalTraining_c.arff";
				strTest = "10-TotalTest.arff";
				break;
			case 6:
				strTrain = "11-TotalPosNgram.arff";
				strSelfTest = "11-TotalPosNgram_t.arff";
```

```
                strModule = "11-TotalPosNgram_m.arff";
                strClassification = "11-TotalPosNgram_c.arff";
                strTest = "12-TotalPosNgramTest.arff";
                break;
        case 7:
                strTrain = "13-TotalSafarNgram.arff";
                strSelfTest = "13-TotalSafarNgram_t.arff";
                strModule = "13-TotalSafarNgram_m.arff";
                strClassification = "13-TotalSafarNgram_c.arff";
                strTest = "14-TotalSafarNgramTest.arff";
                break;
        case 8:
                strTrain = "15-TotalAllNgram.arff";
                strSelfTest = "15-TotalAllNgram_t.arff";
                strModule = "15-TotalAllNgram_m.arff";
                strClassification = "15-TotalAllNgram_c.arff";
                strTest = "16-TotalAllNgramTest.arff";
                break;
        }

        DataSource source = null;
        if (executionIndex == 9) {
                source = new DataSource(userTraining);
        } else {
                source = new DataSource(driveC + "arffOut\\" + strTrain);
        }
        trainInsts = source.getDataSet();

        int classIndex = trainInsts.numAttributes() - 1;
        trainInsts.setClassIndex(classIndex);
        ////////////////////////////////////////////////////////

        // Model Data Set
        if (executionIndex == 9) {
                source = new DataSource(userTest);
        } else {
                source = new DataSource(driveC + "arffOut\\" + strModule);
        }

        moduleInsts = source.getDataSet();
        moduleInsts.setClassIndex(classIndex);
        ////////////////////////////////////////////////////////

        // Module Data Set
        if (executionIndex == 9) {
                source = new DataSource(userTest);
        } else {
                source = new DataSource(driveC + "arffOut\\" + strClassification);
```

```
                }

                classificationInsts = source.getDataSet();
                classificationInsts.setClassIndex(classIndex);
                ////////////////////////////////////////////////////////

                // Self Test Data Set
                if (executionIndex == 9) {
                        source = new DataSource(userTest);
                } else {
                        source = new DataSource(driveC + "arffOut\\" + strSelfTest);
                }

                selfTestInsts = source.getDataSet();
                selfTestInsts.setClassIndex(classIndex);
                ////////////////////////////////////////////////////////

                // Test Data Set
                if (executionIndex == 9) {
                        source = new DataSource(userTest);
                } else {
                        source = new DataSource(driveC + "arffOut\\" + strTest);
                }

                testInsts = source.getDataSet();
                // testInsts = RemoveFilter(testInsts, new
                // String[]{"-R",String.valueOf(testInsts.numAttributes()-1)});
                testInsts.setClassIndex(classIndex);

        } catch (Exception e) {
                e.printStackTrace();
        }
}

public static Instances RemoveFilter(Instances insts, String[] rem) {
        Remove remove = new Remove();
        try {
                remove.setOptions(rem);
                remove.setInputFormat(insts);
                Instances retInstances = Filter.useFilter(insts, remove);
                return retInstances;
        } catch (Exception e) {
                e.printStackTrace();
        }
        return null;
}

public static Instances AddFilter(Instances insts, String[] rem) {
```

```
		Add add = new Add();
		try {
			add.setOptions(rem);
			add.setInputFormat(insts);
			Instances retInstances = Filter.useFilter(insts, add);
			return retInstances;
		} catch (Exception e) {
			e.printStackTrace();
		}
		return null;
	}

	public static double[] calculateMean(double[] m) {

		double[] ret = new double[3];
		double min = 1, max = 0;

		for (int i = 0; i < m.length; i++) {

			if (min > m[i])
				min = m[i];

			if (max < m[i])
				max = m[i];

			ret[0] += m[i];
		}
		ret[0] = (ret[0]) / (m.length);
		ret[1] = min;
		ret[2] = max;

		return ret;
	}

	public static double[] calculateCount(double[] m) {

		double[] ret = new double[6];

		for (int i = 0; i < m.length; i++) {

			if (m[i] < 0.5)
				ret[0]++;

			if (m[i] >= 0.5 && m[i] <= 0.6)
				ret[1]++;

			if (m[i] > 0.6 && m[i] <= 0.7)
				ret[2]++;
```

```
                        if (m[i] > 0.7 && m[i] <= 0.8)
                                ret[3]++;

                        if (m[i] > 0.8 && m[i] <= 0.9)
                                ret[4]++;

                        if (m[i] > 0.9 && m[i] <= 1)
                                ret[5]++;
                }
                return ret;
        }

        public static double calculateStandardDeviation(double[] sd) {

                double sum = 0;
                double newSum = 0;
                double[] newArray = new double[sd.length];

                for (int i = 0; i < sd.length; i++) {
                        sum = sum + sd[i];
                }
                double mean = (sum) / (sd.length);

                for (int j = 0; j < sd.length; j++) {
                        newArray[j] = ((sd[j] - mean) * (sd[j] - mean));
                        newSum = newSum + newArray[j];
                }
                double squaredDiffMean = (newSum) / (sd.length);
                double standardDev = (Math.sqrt(squaredDiffMean));

                return standardDev;
        }
}
```

SafarFiles.java

```
package init;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

class SafarFiles {

	public final static String driveC = "G:\\weka_project\\";

	public final static String sDir1 = "F:\\safar_project\\safarData\\sourceAnti30\\";
	public final static String dDir1 = "F:\\safar_project\\TextFolder\\textAnti30\\";

	public final static String sDir2 = "F:\\safar_project\\safarData\\sourceAnti70\\";
	public final static String dDir2 = "F:\\safar_project\\TextFolder\\textAnti70\\";

	public final static String sDir3 = "F:\\safar_project\\safarData\\sourceAntiTest\\";
	public final static String dDir3 = "F:\\safar_project\\TextFolder\\textAntiTest\\";

	public final static String sDir4 = "F:\\safar_project\\safarData\\sourceNeutral30\\";
	public final static String dDir4 = "F:\\safar_project\\TextFolder\\textNeutral30\\";

	public final static String sDir5 = "F:\\safar_project\\safarData\\sourceNeutral70\\";
	public final static String dDir5 = "F:\\safar_project\\TextFolder\\textNeutral70\\";

	public final static String sDir6 = "F:\\safar_project\\safarData\\sourceNeutralTest\\";
	public final static String dDir6 = "F:\\safar_project\\TextFolder\\textNeutralTest\\";

	public final static String sDir7 = "F:\\safar_project\\safarData\\sourcePro30\\";
	public final static String dDir7 = "F:\\safar_project\\TextFolder\\textPro30\\";

	public final static String sDir8 = "F:\\safar_project\\safarData\\sourcePro70\\";
	public final static String dDir8 = "F:\\safar_project\\TextFolder\\textPro70\\";

	public final static String sDir9 = "F:\\safar_project\\safarData\\sourceProTest\\";
	public final static String dDir9 = "F:\\safar_project\\TextFolder\\textProTest\\";

	private final static String mainDir = "F:\\safar_project\\TextFolder\\";

	private static String tmpFile = "";
	private static File sourceFolder = null;
	private static File[] sourceFiles = null;
	private static Path from = null;
	private static Path to;
```

```
	public static void main(String[] args) {

		movesFiles(sDir1, dDir1);
		movesFiles(sDir2, dDir2);
		movesFiles(sDir3, dDir3);
		movesFiles(sDir4, dDir4);
		movesFiles(sDir5, dDir5);
		movesFiles(sDir6, dDir6);
		movesFiles(sDir7, dDir7);
		movesFiles(sDir8, dDir8);
		movesFiles(sDir9, dDir9);
	}

	private static void movesFiles(String _from, String _to) {

		sourceFolder = new File(_from);
		sourceFiles = sourceFolder.listFiles();
		to = Paths.get(_to);

		for (int i = 0; i < sourceFiles.length; i++) {

			tmpFile = sourceFiles[i].getName().toString();

			from = Paths.get(mainDir + tmpFile);

			try {
				Files.move(from, to.resolve(from.getFileName()));
			} catch (IOException e) {
				System.out.print(tmpFile);
				System.out.print(_to);
				System.out.print("");
			}
		}
	}
}
```

User Interface

A_NGramCalcGUI.java

```
package userInterface;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTextPane;
import CalcNgram.CalcPosNgram;
import CalcNgram.CalcSafarNgram;
import arffCreator.ArffPosClass;
import arffCreator.ArffSafarClass;
import arffCreator.ArffTotal;

import java.awt.Font;
import javax.swing.JButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

import java.awt.Insets;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;

public class A_NGramCalcGUI {

	private JFrame frame;
	private static JTextArea txtOut = new JTextArea();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {

				try {
					A_NGramCalcGUI window = new A_NGramCalcGUI();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
```

```
	/**
	 * Create the application.
	 */
	public A_NGramCalcGUI() {
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.getContentPane().setBackground(Color.BLUE);
		frame.setBounds(100, 100, 450, 518);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.getContentPane().setLayout(null);

		JTextPane txtpnStepsOfOperation = new JTextPane();
		txtpnStepsOfOperation.setMargin(new Insets(5, 10, 10, 5));
		txtpnStepsOfOperation.setFont(new Font("Tahoma", Font.BOLD, 11));
		txtpnStepsOfOperation.setText("Steps of Operation:\r\n1- Extract all received RAR
files to the path C:\\weka_project.\r\n2- Insure that you are using copy of empty database.\r\n3-
Start the 3 interface in sequence:\r\n	a- NGram >> apply pos ngram button >> apply sgram
button\r\n	b- ArffCreator >> select all buttons >> check N-Gram >> select all buttons.\r\n	c-
WekaCalc >> select the desired input Arff files >> select the desired classifiers >> select the execution
button.\r\n4- collect results by [copy] or [save] buttons.");
		txtpnStepsOfOperation.setBounds(0, 0, 434, 155);
		frame.getContentPane().add(txtpnStepsOfOperation);

		JButton btnPos = new JButton("POSIT");
		btnPos.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {

				//1
				try {
					CalcPosNgram.initBD();
					CalcPosNgram.calcNgram(false);
					CalcPosNgram.calcNgram(true);
				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
		});
		btnPos.setFont(new Font("Tahoma", Font.BOLD, 11));
		btnPos.setBounds(25, 166, 89, 23);
		frame.getContentPane().add(btnPos);

		JButton btnSafar = new JButton("SAFAR");
```

```
btnSafar.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		try {
			CalcSafarNgram.calcNgram();
		} catch (SQLException ex) {
			ex.printStackTrace();
		}
	}
});
btnSafar.setFont(new Font("Tahoma", Font.BOLD, 11));
btnSafar.setBounds(124, 166, 89, 23);
frame.getContentPane().add(btnSafar);

JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(10, 198, 415, 270);
frame.getContentPane().add(scrollPane);

txtOut.setMargin(new Insets(5, 8, 5, 8));
scrollPane.setViewportView(txtOut);
txtOut.setColumns(10);

JButton btnRunAll = new JButton("POS + SAFR");
btnRunAll.setBackground(new Color(255, 255, 255));
btnRunAll.setForeground(Color.RED);
btnRunAll.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {

		//1
		try {
			CalcPosNgram.initBD();
			CalcPosNgram.calcNgram(false);
			CalcPosNgram.calcNgram(true);
		} catch (SQLException e) {
			e.printStackTrace();
		}
		//2
		try {
			CalcSafarNgram.calcNgram();
		} catch (SQLException ex) {
			ex.printStackTrace();
		}
		//3
		ArffPosClass.BtnCalc1(true, false, false);
		ArffPosClass.BtnCalc1(true, false, true);
		ArffPosClass.BtnCalc1(true, true, false);
		//4
		ArffPosClass.BtnCalc1(false, false, false);
		ArffPosClass.BtnCalc1(false, false, true);
```

```
                                ArffPosClass.BtnCalc1(false, true, false);
                                //5
                                ArffSafarClass.BtnCalc1(true, false, false);
                                ArffSafarClass.BtnCalc1(true, false, true);
                                ArffSafarClass.BtnCalc1(true, true, false);
                                //6
                                ArffSafarClass.BtnCalc1(false, false, false);
                                ArffSafarClass.BtnCalc1(false, false, true);
                                ArffSafarClass.BtnCalc1(false, true, false);
                                //7
                                ArffTotal.BtnCalc1(1, false, false);
                                ArffTotal.BtnCalc1(1, false, true);
                                ArffTotal.BtnCalc1(1, true, false);
                                //8
                                ArffTotal.BtnCalc1(2, false, false);
                                ArffTotal.BtnCalc1(2, false, true);
                                ArffTotal.BtnCalc1(2, true, false);
                                //9
                                ArffTotal.BtnCalc1(3, false, false);
                                ArffTotal.BtnCalc1(3, false, true);
                                ArffTotal.BtnCalc1(3, true, false);
                                //10
                                ArffTotal.BtnCalc1(0, false, false);
                                ArffTotal.BtnCalc1(0, false, true);
                                ArffTotal.BtnCalc1(0, true, false);
                        }
                });
                btnRunAll.setFont(new Font("Tahoma", Font.BOLD, 11));
                btnRunAll.setBounds(284, 166, 128, 23);
                frame.getContentPane().add(btnRunAll);
        }

        public static void PrintOut(String txt) {

                txtOut.append(String.valueOf(txt)+"\n");

        }

        public static void PrintOut2(String txt) {

                txtOut.append(String.valueOf(txt)+"\n");

        }
}
```

B_ArffCreatorGUI.java

```
package userInterface;

import java.awt.Color;
import java.awt.Desktop;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.Window.Type;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.StringSelection;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;

import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

import arffCreator.ArffPosClass;
import arffCreator.ArffSafarClass;
import arffCreator.ArffTotal;
import java.awt.Insets;

public class B_ArffCreatorGUI {

	private JFrame frmCreateArffFile;
	private static JTextArea txtOut = new JTextArea();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					B_ArffCreatorGUI window = new B_ArffCreatorGUI();
					window.frmCreateArffFile.setVisible(true);
				} catch (Exception e) {
```

```
                                        e.printStackTrace();
                                }
                        }
                });
        }

        /**
         * Create the application.
         */
        public B_ArffCreatorGUI() {
                initialize();
        }

        int i = 0;

        /**
         * Initialize the contents of the frame.
         */
        private void initialize() {
                frmCreateArffFile = new JFrame();
                frmCreateArffFile.setForeground(new Color(0, 0, 0));
                frmCreateArffFile.setType(Type.UTILITY);
                frmCreateArffFile.setTitle("Create ARFF File");
                frmCreateArffFile.getContentPane().setBackground(new Color(0, 0, 255));
                frmCreateArffFile.setBounds(100, 100, 310, 540);
                frmCreateArffFile.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frmCreateArffFile.getContentPane().setLayout(null);

                JLabel lblSafar = new JLabel("\u221A");
                lblSafar.setVisible(false);
                lblSafar.setForeground(Color.BLACK);
                lblSafar.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 16));
                lblSafar.setBounds(10, 10, 17, 23);

                JLabel lblPos = new JLabel("\u221A");
                lblPos.setVisible(false);
                lblPos.setForeground(new Color(0, 0, 0));
                lblPos.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 16));
                lblPos.setBounds(10, 10, 17, 23);

                JLabel lblPosSafar = new JLabel("\u221A");
                lblPosSafar.setVisible(false);
                lblPosSafar.setForeground(Color.BLACK);
                lblPosSafar.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 16));
                lblPosSafar.setBounds(10, 10, 17, 23);
                txtOut.setMargin(new Insets(5, 8, 8, 5));

                txtOut.setBounds(10, 196, 275, 229);
```

```
txtOut.setWrapStyleWord(true);
JScrollPane scroll = new JScrollPane(txtOut);
scroll.setBounds(10, 188, 275, 268);
frmCreateArffFile.getContentPane().add(scroll);

JPanel panel = new JPanel();
panel.setLayout(null);
panel.setBackground(new Color(0, 255, 102));
panel.setBounds(10, 11, 275, 40);
frmCreateArffFile.getContentPane().add(panel);
panel.add(lblPos);

JButton btnPos = new JButton("POS");
JCheckBox chkPos = new JCheckBox("POS N-Gram");

btnPos.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		lblPos.setVisible(true);
		lblSafar.setVisible(false);
		lblPosSafar.setVisible(false);
		btnPos.setText("POS ... in progress");

		if (chkPos.isSelected()) {
			//////////////////// ngram, test, model
			//70% + 30%
			ArffPosClass.BtnCalc1(true, false, false);
			//100%
			ArffPosClass.BtnCalc1(true, false, true);
			// UnSupervised + Supervised
			ArffPosClass.BtnCalc1(true, true, false);
		} else {
			ArffPosClass.BtnCalc1(false, false, false);//3*700
			ArffPosClass.BtnCalc1(false, false, true);//3*1000
			ArffPosClass.BtnCalc1(false, true, false);//3*
		}
		btnPos.setText("POS");
	}
});
btnPos.setBounds(35, 10, 117, 23);
panel.add(btnPos);

chkPos.setBounds(158, 10, 111, 23);
panel.add(chkPos);

JPanel panel_1 = new JPanel();
panel_1.setLayout(null);
panel_1.setBackground(new Color(102, 255, 255));
panel_1.setBounds(10, 57, 275, 40);
```

```
frmCreateArffFile.getContentPane().add(panel_1);
panel_1.add(lblSafar);

JButton btnSafra = new JButton("SAFAR");
JCheckBox chkSafar = new JCheckBox("SAFAR N-Gram");
btnSafra.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {

		lblPos.setVisible(false);
		lblSafar.setVisible(true);
		lblPosSafar.setVisible(false);
		btnPos.setText("SAFAR ... in progress");

		if (chkSafar.isSelected()) {
			////////////////////// ngram, test, module
			ArffSafarClass.BtnCalc1(true, false, false);
			ArffSafarClass.BtnCalc1(true, false, true);
			ArffSafarClass.BtnCalc1(true, true, false);
		} else {
			ArffSafarClass.BtnCalc1(false, false, false);//3*700
			ArffSafarClass.BtnCalc1(false, false, true);
			ArffSafarClass.BtnCalc1(false, true, false);
		}
	}
});
btnSafra.setBounds(35, 10, 117, 23);
panel_1.add(btnSafra);

chkSafar.setBounds(158, 10, 111, 23);
panel_1.add(chkSafar);

JPanel panel_2 = new JPanel();
panel_2.setLayout(null);
panel_2.setBackground(new Color(255, 153, 255));
panel_2.setBounds(10, 105, 275, 72);
frmCreateArffFile.getContentPane().add(panel_2);
panel_2.add(lblPosSafar);

ButtonGroup radGrp = new ButtonGroup();
JButton btnPosSafar = new JButton("POS + SAFAR");
JCheckBox chkPosSafar = new JCheckBox("N-Gram");
JRadioButton radPos = new JRadioButton("POS");
radPos.setSelected(true);
JRadioButton radSafar = new JRadioButton("SAFAR");
JRadioButton radPosSafar = new JRadioButton("POS + SAFAR");
radGrp.add(radPos);
radGrp.add(radSafar);
radGrp.add(radPosSafar);
```

```
		btnPosSafar.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {

				lblPos.setVisible(false);
				lblSafar.setVisible(false);
				lblPosSafar.setVisible(true);
				if (chkPosSafar.isSelected()) {
					if (radPos.isSelected()) {
						////////////////////// ngram, test, module
						//70%    //30%
						ArffTotal.BtnCalc1(1, false, false);
						//100%
						ArffTotal.BtnCalc1(1, false, true);
						//UnSupervised + //Supervised
						ArffTotal.BtnCalc1(1, true, false);
					}
					if (radSafar.isSelected()) {
						ArffTotal.BtnCalc1(2, false, false);
						ArffTotal.BtnCalc1(2, false, true);
						ArffTotal.BtnCalc1(2, true, false);
					}
					if (radPosSafar.isSelected()) {
//						////////////////// ngram, test, model
						ArffTotal.BtnCalc1(3, false, false);
						ArffTotal.BtnCalc1(3, false, true);
						ArffTotal.BtnCalc1(3, true, false);
					}
				} else {
					ArffTotal.BtnCalc1(0, false, false);
					ArffTotal.BtnCalc1(0, false, true);
					ArffTotal.BtnCalc1(0, true, false);
				}
			}
		});
		btnPosSafar.setBounds(35, 10, 117, 23);
		panel_2.add(btnPosSafar);

		chkPosSafar.setBounds(158, 10, 111, 23);
		panel_2.add(chkPosSafar);

		radPos.setBounds(10, 42, 70, 23);
		panel_2.add(radPos);

		radSafar.setBounds(86, 42, 70, 23);
		panel_2.add(radSafar);
		radPosSafar.setBounds(162, 42, 105, 23);
		panel_2.add(radPosSafar);
```

```
		JButton btnSave = new JButton("Save");
		btnSave.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {

				JFileChooser fileChooser = new JFileChooser();
				int retval = fileChooser.showSaveDialog(btnSave);
				if (retval == JFileChooser.APPROVE_OPTION) {
					File file = fileChooser.getSelectedFile();
					if (file == null) {
						return;
					}
					if (!file.getName().toLowerCase().endsWith(".txt")) {
						file = new File(file.getParentFile(), file.getName() +
".txt");
					}
					try {
						txtOut.write(new OutputStreamWriter(new
FileOutputStream(file), "utf-8"));
						Desktop.getDesktop().open(file);
					} catch (Exception e) {
						e.printStackTrace();
					}
				}
			}
		});
		btnSave.setBounds(205, 467, 80, 23);
		frmCreateArffFile.getContentPane().add(btnSave);

		JButton btnCopy = new JButton("Copy");
		btnCopy.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {

				String myString = txtOut.getText();
				StringSelection stringSelection = new StringSelection(myString);
				Clipboard clipboard =
Toolkit.getDefaultToolkit().getSystemClipboard();
				clipboard.setContents(stringSelection, null);

			}
		});
		btnCopy.setBounds(10, 467, 70, 23);
		frmCreateArffFile.getContentPane().add(btnCopy);

		JButton btnClear = new JButton("Clear");
		btnClear.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				txtOut.setText("");
			}
```

```
                });
                btnClear.setBounds(90, 467, 70, 23);
                frmCreateArffFile.getContentPane().add(btnClear);
        }

        public static void PrintOut(String txt) {

                txtOut.append(String.valueOf(txt) + "\n");

        }

}
```

C_ClassifierGUI.java

```
package userInterface;

import java.awt.EventQueue;

import javax.swing.JFrame;
import java.awt.Color;
import java.awt.Desktop;

import javax.swing.JPanel;
import javax.swing.JRadioButton;

import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.awt.event.ActionEvent;
import javax.swing.border.TitledBorder;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

import init.Glob;
import wekaCalc.FinalCalc;

import javax.swing.UIManager;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.StringSelection;
import javax.swing.JLabel;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;

public class C_ClassifierGUI {

	private JFrame frmCreateApply;
	private static JTextArea txtOut = new JTextArea();

	private static JFileChooser fileChooser = new JFileChooser();
	private JCheckBox chkEnhanceDetails;
```

```
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					C_ClassifierGUI window = new C_ClassifierGUI();
					window.frmCreateApply.setVisible(true);
					FileFilter filt = new FileNameExtensionFilter("ARFF File",
"arff");
					fileChooser.setFileFilter(filt);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public C_ClassifierGUI() {
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frmCreateApply = new JFrame();
		frmCreateApply.setTitle("CREATE & APPLY CLASSIFICATION MODULE");
		frmCreateApply.getContentPane().setBackground(Color.BLUE);
		frmCreateApply.getContentPane().setLayout(null);

		JPanel panel = new JPanel();
		panel.setBackground(new Color(255, 255, 204));
		panel.setBounds(10, 8, 192, 675);
		frmCreateApply.getContentPane().add(panel);
		panel.setLayout(null);

		ButtonGroup radGrp = new ButtonGroup();

		JRadioButton radPOS = new JRadioButton("POSIT Toolkit");
		radPOS.setBackground(new Color(255, 255, 204));
		radPOS.setFont(new Font("Tahoma", Font.BOLD, 11));
		radPOS.setBounds(16, 4, 105, 23);
		radPOS.setForeground(new Color(0, 128, 128));
```

```
                panel.add(radPOS);

                JRadioButton radSAF = new JRadioButton("SAFAR Toolkit");
                radSAF.setFont(new Font("Tahoma", Font.BOLD, 11));
                radSAF.setBackground(new Color(255, 255, 204));
                radSAF.setBounds(16, 27, 105, 23);
                radSAF.setForeground(Color.BLUE);
                panel.add(radSAF);

                JRadioButton radPOSSAF = new JRadioButton("POS + SAFAR Attributes");
                radPOSSAF.setSelected(true);
                radPOSSAF.setFont(new Font("Tahoma", Font.BOLD, 11));
                radPOSSAF.setBackground(new Color(255, 255, 204));
                radPOSSAF.setBounds(16, 50, 161, 23);
                radPOSSAF.setForeground(Color.RED);
                panel.add(radPOSSAF);

                JRadioButton radUser = new JRadioButton("User");
                radUser.setBounds(16, 138, 49, 23);
                radUser.setFont(new Font("Tahoma", Font.BOLD, 10));
                radUser.setBackground(new Color(255, 255, 204));
                panel.add(radUser);

                radGrp.add(radPOS);
                radGrp.add(radSAF);
                radGrp.add(radPOSSAF);
                radGrp.add(radUser);

                JCheckBox chkNgram = new JCheckBox("N-Gram");
                chkNgram.setForeground(Color.RED);
                chkNgram.setSelected(true);
                chkNgram.setFont(new Font("Tahoma", Font.PLAIN, 14));
                chkNgram.setBackground(new Color(255, 255, 204));
                chkNgram.setBounds(16, 163, 161, 23);
                panel.add(chkNgram);

                JPanel panel_1 = new JPanel();
                panel_1.setBackground(new Color(255, 255, 204));
                panel_1.setBorder(new TitledBorder(
                                new TitledBorder(UIManager.getBorder("TitledBorder.border"), "",
TitledBorder.LEADING, TitledBorder.TOP,
                                                null, new Color(0, 0, 255)),
                                "", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(51, 204,
255)));
                panel_1.setBounds(16, 80, 161, 35);
                panel.add(panel_1);
                panel_1.setLayout(null);
```

```
                ButtonGroup radGrpT = new ButtonGroup();
                ButtonGroup radGrpC = new ButtonGroup();

                JRadioButton radP = new JRadioButton("P");
                radP.setFont(new Font("Tahoma", Font.BOLD, 10));
                radP.setBackground(new Color(255, 255, 204));
                radP.setBounds(6, 7, 45, 23);
                panel_1.add(radP);

                JRadioButton radS = new JRadioButton("S");
                radS.setFont(new Font("Tahoma", Font.BOLD, 10));
                radS.setBackground(new Color(255, 255, 204));
                radS.setBounds(53, 7, 45, 23);
                panel_1.add(radS);

                JRadioButton radPS = new JRadioButton("P+S");
                radPS.setForeground(new Color(255, 0, 0));
                radPS.setFont(new Font("Tahoma", Font.BOLD, 10));
                radPS.setBackground(new Color(255, 255, 204));
                radPS.setSelected(true);
                radPS.setBounds(100, 7, 55, 23);
                panel_1.add(radPS);

                radGrpT.add(radP);
                radGrpT.add(radS);
                radGrpT.add(radPS);

                JPanel panel_2 = new JPanel();
                panel_2.setLayout(null);
                panel_2.setBorder(new TitledBorder(
                                new TitledBorder(UIManager.getBorder("TitledBorder.border"), "",
TitledBorder.LEADING, TitledBorder.TOP,
                                                null, new Color(0, 0, 255)),
                                "", TitledBorder.LEADING, TitledBorder.TOP, null, new Color(51, 204,
255)));
                panel_2.setBackground(new Color(204, 255, 204));
                panel_2.setBounds(16, 188, 161, 194);
                panel.add(panel_2);

                JRadioButton radJ48 = new JRadioButton("J48");
                radJ48.setFont(new Font("Tahoma", Font.BOLD, 11));
                radJ48.setBackground(new Color(204, 255, 204));
                radJ48.setBounds(6, 4, 149, 23);
                panel_2.add(radJ48);

                JRadioButton radNaivebayes = new JRadioButton("NaiveBayes");
                radNaivebayes.setFont(new Font("Tahoma", Font.BOLD, 11));
                radNaivebayes.setBackground(new Color(204, 255, 204));
```

```
radNaivebayes.setBounds(6, 31, 149, 23);
panel_2.add(radNaivebayes);

JRadioButton radIBK_1 = new JRadioButton("IBk_1");
radIBK_1.setFont(new Font("Tahoma", Font.BOLD, 11));
radIBK_1.setBackground(new Color(204, 255, 204));
radIBK_1.setBounds(6, 58, 149, 23);
panel_2.add(radIBK_1);

JRadioButton radIBk_3 = new JRadioButton("IBk_3");
radIBk_3.setFont(new Font("Tahoma", Font.BOLD, 11));
radIBk_3.setBackground(new Color(204, 255, 204));
radIBk_3.setBounds(6, 85, 149, 23);
panel_2.add(radIBk_3);

JRadioButton radRF = new JRadioButton("Random Forest");
radRF.setSelected(true);
radRF.setFont(new Font("Tahoma", Font.BOLD, 11));
radRF.setBackground(new Color(204, 255, 204));
radRF.setBounds(6, 139, 149, 23);
panel_2.add(radRF);

JRadioButton radCVR = new JRadioButton("RF Via Regression");
radCVR.setFont(new Font("Tahoma", Font.BOLD, 11));
radCVR.setBackground(new Color(204, 255, 204));
radCVR.setBounds(6, 165, 149, 23);
panel_2.add(radCVR);

radGrpC.add(radJ48);
radGrpC.add(radNaivebayes);
radGrpC.add(radIBK_1);
radGrpC.add(radIBk_3);
radGrpC.add(radRF);
radGrpC.add(radCVR);

JRadioButton radSMO = new JRadioButton("SMO");
radSMO.setBounds(6, 112, 149, 23);
panel_2.add(radSMO);
radSMO.setFont(new Font("Tahoma", Font.BOLD, 11));
radSMO.setBackground(new Color(204, 255, 204));
radGrpC.add(radSMO);

JButton btnClear = new JButton("Clear");
btnClear.setMargin(new Insets(2, 4, 2, 4));
btnClear.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                txtOut.setText("");
        }
```

```
		});
		btnClear.setBounds(71, 560, 50, 23);
		panel.add(btnClear);

		JButton btnCopy = new JButton("Copy");
		btnCopy.setMargin(new Insets(2, 4, 2, 4));
		btnCopy.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {

				String myString = txtOut.getText();
				StringSelection stringSelection = new StringSelection(myString);
				Clipboard clipboard =
Toolkit.getDefaultToolkit().getSystemClipboard();
				clipboard.setContents(stringSelection, null);

			}
		});
		btnCopy.setBounds(16, 560, 50, 23);
		panel.add(btnCopy);

		JButton btnCalculatePredection = new JButton("Calculate Predection");
		btnCalculatePredection.setFont(new Font("Tahoma", Font.BOLD, 12));
		btnCalculatePredection.setBounds(16, 469, 161, 61);
		panel.add(btnCalculatePredection);

		JButton btnSave = new JButton("Save");
		btnSave.setMargin(new Insets(2, 4, 2, 4));
		btnSave.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {

				int retval = fileChooser.showSaveDialog(btnSave);
				if (retval == JFileChooser.APPROVE_OPTION) {
					File file = fileChooser.getSelectedFile();
					if (file == null) {
						return;
					}
					if (!file.getName().toLowerCase().endsWith(".txt")) {
						file = new File(file.getParentFile(), file.getName() +
".txt");
					}
					try {
						txtOut.write(new OutputStreamWriter(new
FileOutputStream(file), "utf-8"));
						Desktop.getDesktop().open(file);
					} catch (Exception e) {
						e.printStackTrace();
					}
				}
```

```
		}
	});
	btnSave.setBounds(127, 560, 50, 23);
	panel.add(btnSave);

	JLabel lblTraining = new JLabel("Training");
	lblTraining.setForeground(new Color(211, 211, 211));
	lblTraining.setFont(new Font("Tahoma", Font.BOLD, 11));
	lblTraining.setBounds(88, 121, 50, 14);
	panel.add(lblTraining);

	JLabel lblTest = new JLabel("Test");
	lblTest.setFont(new Font("Tahoma", Font.BOLD, 11));
	lblTest.setForeground(new Color(211, 211, 211));
	lblTest.setBounds(144, 121, 33, 14);
	panel.add(lblTest);

	JButton btnTraining = new JButton("...");
	btnTraining.setEnabled(false);
	btnTraining.setBounds(88, 138, 33, 23);
	panel.add(btnTraining);
	btnTraining.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent arg0) {

			int result = fileChooser.showOpenDialog(btnTraining);
			if (result == JFileChooser.APPROVE_OPTION) {
				try {
					File f = fileChooser.getSelectedFile();
					Glob.userTraining = f.getAbsolutePath();
					lblTraining.setForeground(Color.RED);
				} catch (Exception ex) {
					ex.printStackTrace();
				}
			}
		}
	});
	btnTraining.setMargin(new Insets(2, 4, 2, 4));

	JButton btnTest = new JButton("...");
	btnTest.setEnabled(false);
	btnTest.setBounds(144, 138, 33, 23);
	panel.add(btnTest);
	btnTest.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {

			int result = fileChooser.showOpenDialog(btnTest);
			if (result == JFileChooser.APPROVE_OPTION) {
```

```
                                try {
                                        File f = fileChooser.getSelectedFile();
                                        Glob.userTest = f.getAbsolutePath();
                                        lblTest.setForeground(Color.RED);
                                } catch (Exception ex) {
                                        ex.printStackTrace();
                                }
                        }
                }
        });
        btnTest.setMargin(new Insets(2, 4, 2, 4));

        JCheckBox chkDisplayDetails = new JCheckBox("70/30 Details");
        chkDisplayDetails.setFont(new Font("Tahoma", Font.BOLD, 11));
        chkDisplayDetails.setBackground(new Color(255, 255, 204));
        chkDisplayDetails.setBounds(16, 389, 161, 23);
        panel.add(chkDisplayDetails);

        JCheckBox chkTrainingDetails = new JCheckBox("Training Predic. Details");
        chkTrainingDetails.setFont(new Font("Tahoma", Font.BOLD, 11));
        chkTrainingDetails.setBackground(new Color(255, 255, 204));
        chkTrainingDetails.setBounds(16, 415, 161, 23);
        panel.add(chkTrainingDetails);

        JCheckBox chkTestDetails = new JCheckBox("Test Predic. Details");
        chkTestDetails.setFont(new Font("Tahoma", Font.BOLD, 11));
        chkTestDetails.setBackground(new Color(255, 255, 204));
        chkTestDetails.setBounds(16, 439, 161, 23);
        panel.add(chkTestDetails);

        JCheckBox chkUpdateModule = new JCheckBox("Update Model");
        chkUpdateModule.setSelected(true);
        chkUpdateModule.setForeground(Color.RED);
        chkUpdateModule.setFont(new Font("Tahoma", Font.BOLD, 12));
        chkUpdateModule.setBackground(new Color(255, 255, 204));
        chkUpdateModule.setBounds(16, 535, 161, 23);
        panel.add(chkUpdateModule);

        JPanel panel_3 = new JPanel();
        panel_3.setBackground(Color.RED);
        panel_3.setBounds(16, 590, 161, 63);
        panel.add(panel_3);

        JCheckBox chkMoveSelf = new JCheckBox("Self");
        JCheckBox chkMoveCross = new JCheckBox("Cross");
        chkMoveSelf.addChangeListener(new ChangeListener() {
                public void stateChanged(ChangeEvent e) {
                        if (chkMoveSelf.isSelected()) {
```

```
                        chkMoveCross.setEnabled(false);
                        chkDisplayDetails.setSelected(true);
                        chkTestDetails.setSelected(true);
                        chkTrainingDetails.setSelected(true);
                } else {
                        chkMoveCross.setEnabled(true);
                        chkDisplayDetails.setSelected(false);
                        chkTestDetails.setSelected(false);
                        chkTrainingDetails.setSelected(false);
                }
        }
});
chkMoveSelf.setForeground(Color.WHITE);
chkMoveSelf.setFont(new Font("Tahoma", Font.BOLD, 11));
chkMoveSelf.setBackground(Color.RED);
panel_3.add(chkMoveSelf);

JLabel lblA = new JLabel("A");
JLabel lblP = new JLabel("P");
JLabel lblN = new JLabel("N");

chkMoveCross.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
                if (chkMoveCross.isSelected()) {
                        chkMoveSelf.setEnabled(false);
                        chkDisplayDetails.setSelected(true);
                        chkTestDetails.setSelected(true);
                        chkTrainingDetails.setSelected(true);
                } else {
                        chkMoveSelf.setEnabled(true);
                        chkDisplayDetails.setSelected(false);
                        chkTestDetails.setSelected(false);
                        chkTrainingDetails.setSelected(false);
                }
        }
});
chkMoveCross.setForeground(Color.WHITE);
chkMoveCross.setFont(new Font("Tahoma", Font.BOLD, 11));
chkMoveCross.setBackground(Color.RED);
panel_3.add(chkMoveCross);

JButton btn100 = new JButton("Complete 100%");
btn100.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
                int res = 0;
                res = FinalCalc.Complete100("anti");
                lblA.setText(res + "");
                res = FinalCalc.Complete100("pro");
```

```
                    lblP.setText(res + "");
                    res = FinalCalc.Complete100("neutral");
                    lblN.setText(res + "");
            }
});
panel_3.add(btn100);
btn100.setFont(new Font("Tahoma", Font.BOLD, 12));

JCheckBox chkEnhance = new JCheckBox("Enhance");
panel_3.add(chkEnhance);
chkEnhance.setForeground(Color.WHITE);
chkEnhance.setFont(new Font("Tahoma", Font.BOLD, 11));
chkEnhance.setBackground(Color.RED);

lblA.setBounds(16, 655, 46, 14);
panel.add(lblA);

lblP.setBounds(71, 655, 46, 14);
panel.add(lblP);

lblN.setBounds(131, 655, 46, 14);
panel.add(lblN);

chkEnhanceDetails = new JCheckBox("Enhance Details");
chkEnhanceDetails.setVisible(false);
chkEnhanceDetails.setFont(new Font("Tahoma", Font.BOLD, 11));
chkEnhanceDetails.setBackground(new Color(255, 255, 204));
chkEnhanceDetails.setBounds(16, 465, 161, 23);
panel.add(chkEnhanceDetails);

JScrollPane scroll = new JScrollPane();
scroll.setBounds(212, 8, 462, 675);
frmCreateApply.getContentPane().add(scroll);
txtOut.setFont(new Font("Monospaced", Font.PLAIN, 14));
txtOut.setMargin(new Insets(5, 8, 5, 8));
txtOut.setBackground(new Color(224, 255, 255));

scroll.setViewportView(txtOut);
txtOut.setColumns(10);
btnCalculatePredection.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

                    boolean details7030 = chkDisplayDetails.isSelected();
                    boolean detailsTraining = chkTrainingDetails.isSelected();
                    boolean detailsTest = chkTestDetails.isSelected();
                    boolean detailsEnhance = chkEnhanceDetails.isSelected();

                    boolean NGram = chkNgram.isSelected();
```

```
                                boolean moveSelf = chkMoveSelf.isSelected();
                                boolean moveCross = chkMoveCross.isSelected();
                                boolean moveEnhance = chkEnhance.isSelected();
                                boolean UpdateModule = chkUpdateModule.isSelected();

                                int[] classifier = new int[7];

                                if (radJ48.isSelected())
                                        classifier[0] = 1;

                                if (radNaivebayes.isSelected())
                                        classifier[1] = 1;

                                if (radIBK_1.isSelected())
                                        classifier[2] = 1;

                                if (radIBk_3.isSelected())
                                        classifier[3] = 1;

                                if (radSMO.isSelected())
                                        classifier[4] = 1;

                                if (radRF.isSelected())
                                        classifier[5] = 1;

                                if (radCVR.isSelected())
                                        classifier[6] = 1;

                                String ESC = "\033[";
                                System.out.print(ESC + "2J");

                                txtOut.setText("");

                                if (radUser.isSelected()) {
                                        if (!Glob.userTraining.equals("") &&
!Glob.userTest.equals("")) {
                                                FinalCalc.calcPrediction(9, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                                false, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                        } else {
                                                PrintOut("Wrong ARFF File Selection");
                                                System.out.println("Wrong ARFF File Selection");
                                        }
                                } else if (radPOS.isSelected()) {
                                        if (NGram) {
                                                FinalCalc.calcPrediction(2, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
```

```
                                                    false, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                        FinalCalc.calcPrediction(2, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    true, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                    } else {
                                        FinalCalc.calcPrediction(1, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    false, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                        FinalCalc.calcPrediction(1, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    true, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                    }
                                } else if (radSAF.isSelected()) {
                                    if (NGram) {
                                        FinalCalc.calcPrediction(4, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    false, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                        FinalCalc.calcPrediction(4, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    true, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                    } else {
                                        FinalCalc.calcPrediction(3, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    false, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                        FinalCalc.calcPrediction(3, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                                    true, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                    }
                                } else if (radPOSSAF.isSelected()) {
                                    if (NGram) {
                                        if (radP.isSelected()) {
                                            FinalCalc.calcPrediction(6, classifier,
UpdateModule, details7030, detailsTraining,
                                                    detailsTest, false, moveSelf,
moveCross, moveEnhance, detailsEnhance);
                                            FinalCalc.calcPrediction(6, classifier,
UpdateModule, details7030, detailsTraining,
                                                    detailsTest, true, moveSelf,
moveCross, moveEnhance, detailsEnhance);
                                        } else if (radS.isSelected()) {
```

```
                                        FinalCalc.calcPrediction(7, classifier,
UpdateModule, details7030, detailsTraining,
                                                detailsTest, false, moveSelf,
moveCross, moveEnhance, detailsEnhance);
                                        FinalCalc.calcPrediction(7, classifier,
UpdateModule, details7030, detailsTraining,
                                                detailsTest, true, moveSelf,
moveCross, moveEnhance, detailsEnhance);
                                    } else if (radPS.isSelected()) {

                                        FinalCalc.calcPrediction(8, classifier,
UpdateModule, details7030, detailsTraining,
                                                detailsTest, false, moveSelf,
moveCross, moveEnhance, detailsEnhance);

                                        FinalCalc.calcPrediction(8, classifier,
UpdateModule, details7030, detailsTraining,
                                                detailsTest, true, moveSelf,
moveCross, moveEnhance, detailsEnhance);
                                    }
                                } else {
                                    FinalCalc.calcPrediction(5, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                            false, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                    FinalCalc.calcPrediction(5, classifier, UpdateModule,
details7030, detailsTraining, detailsTest,
                                            true, moveSelf, moveCross,
moveEnhance, detailsEnhance);
                                }
                            }

                            lblA.setText(new File(Glob.driveC +
"posData\\summary\\anti").listFiles().length + "");
                            lblP.setText(new File(Glob.driveC +
"posData\\summary\\pro").listFiles().length + "");
                            lblN.setText(new File(Glob.driveC +
"posData\\summary\\neutral").listFiles().length + "");

                        }
                    });

                    radPOSSAF.addActionListener(new ActionListener() {

                        public void actionPerformed(ActionEvent e) {
                            radP.setEnabled(true);
                            radS.setEnabled(true);
                            radPS.setEnabled(true);
```

```
                radP.setForeground(Color.black);
                radS.setForeground(Color.black);
                radPS.setForeground(Color.black);
                btnTraining.setEnabled(false);
                btnTest.setEnabled(false);
                lblTraining.setForeground(Color.LIGHT_GRAY);
                lblTest.setForeground(Color.LIGHT_GRAY);
        }
});

radPOS.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                radP.setEnabled(false);
                radS.setEnabled(false);
                radPS.setEnabled(false);
                radP.setForeground(Color.LIGHT_GRAY);
                radS.setForeground(Color.LIGHT_GRAY);
                radPS.setForeground(Color.LIGHT_GRAY);
                btnTraining.setEnabled(false);
                btnTest.setEnabled(false);
                lblTraining.setForeground(Color.LIGHT_GRAY);
                lblTest.setForeground(Color.LIGHT_GRAY);
        }
});

radSAF.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                radP.setEnabled(false);
                radS.setEnabled(false);
                radPS.setEnabled(false);
                radP.setForeground(Color.LIGHT_GRAY);
                radS.setForeground(Color.LIGHT_GRAY);
                radPS.setForeground(Color.LIGHT_GRAY);
                btnTraining.setEnabled(false);
                btnTest.setEnabled(false);
                lblTraining.setForeground(Color.LIGHT_GRAY);
                lblTest.setForeground(Color.LIGHT_GRAY);
        }
});

radUser.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                radP.setEnabled(false);
                radS.setEnabled(false);
                radPS.setEnabled(false);
                radP.setForeground(Color.LIGHT_GRAY);
                radS.setForeground(Color.LIGHT_GRAY);
                radPS.setForeground(Color.LIGHT_GRAY);
```

```
                                btnTraining.setEnabled(true);
                                btnTest.setEnabled(true);
                                lblTraining.setForeground(Color.black);
                                lblTest.setForeground(Color.black);
                        }
                });

                frmCreateApply.setBounds(100, 100, 697, 732);
                frmCreateApply.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        public static void PrintOut(String txt) {

                txtOut.append(String.valueOf(txt) + "\n");

        }
}
```

N-Gram Calculations

PositObj.java

```
package CalcNgram;

public class PositObj {

	public Double Total_words_pos2 = 0.0;
	public Double Total_unique_words_pos2 = 0.0;
	public Double Type_Token_Ratio_pos2 = 0.0;
	public Double Number_of_sentences_pos2 = 0.0;
	public Double Average_sentence_length_pos2 = 0.0;
	public Double Number_of_characters_pos2 = 0.0;
	public Double Average_word_length_pos2 = 0.0;
	public Double verb_types_pos2 = 0.0;
	public Double noun_types_pos2 = 0.0;
	public Double adverb_types_pos2 = 0.0;
	public Double adjective_types_pos2 = 0.0;
	public Double preposition_types_pos2 = 0.0;
	public Double possessive_pronoun_types_pos2 = 0.0;
	public Double personal_pronoun_types_pos2 = 0.0;
	public Double particle_types_pos2 = 0.0;
	public Double determiner_types_pos2 = 0.0;
	public Double interjection_types_pos2 = 0.0;
	public Double nouns_pos2 = 0.0;
	public Double verbs_pos2 = 0.0;
	public Double prepositions_pos2 = 0.0;
	public Double adjectives_pos2 = 0.0;
	public Double determiners_pos2 = 0.0;
	public Double particles_pos2 = 0.0;
	public Double possessive_pronouns_pos2 = 0.0;
	public Double personal_pronouns_pos2 = 0.0;
	public Double adverbs_pos2 = 0.0;
	public Double interjections_pos2 = 0.0;

	public PositObj(Double Total_words_pos2, Double Total_unique_words_pos2,
Double Type_Token_Ratio_pos2,
			Double Number_of_sentences_pos2, Double
Average_sentence_length_pos2, Double Number_of_characters_pos2,
			Double Average_word_length_pos2, Double verb_types_pos2, Double
noun_types_pos2, Double adverb_types_pos2,
			Double adjective_types_pos2, Double preposition_types_pos2,
Double possessive_pronoun_types_pos2,
			Double personal_pronoun_types_pos2, Double particle_types_pos2,
Double determiner_types_pos2,
			Double interjection_types_pos2, Double nouns_pos2, Double
verbs_pos2, Double prepositions_pos2,
			Double adjectives_pos2, Double determiners_pos2, Double
particles_pos2, Double possessive_pronouns_pos2,
			Double personal_pronouns_pos2, Double adverbs_pos2, Double
interjections_pos2) {

		this.Total_words_pos2 = Total_words_pos2;
		this.Total_unique_words_pos2 = Total_unique_words_pos2;
		this.Type_Token_Ratio_pos2 = Type_Token_Ratio_pos2;
```

```
            this.Number_of_sentences_pos2 = Number_of_sentences_pos2;
            this.Average_sentence_length_pos2 = Average_sentence_length_pos2;
            this.Number_of_characters_pos2 = Number_of_characters_pos2;
            this.Average_word_length_pos2 = Average_word_length_pos2;
            this.verb_types_pos2 = verb_types_pos2;
            this.noun_types_pos2 = noun_types_pos2;
            this.adverb_types_pos2 = adverb_types_pos2;
            this.adjective_types_pos2 = adjective_types_pos2;
            this.preposition_types_pos2 = preposition_types_pos2;
            this.possessive_pronoun_types_pos2 = possessive_pronoun_types_pos2;
            this.personal_pronoun_types_pos2 = personal_pronoun_types_pos2;
            this.particle_types_pos2 = particle_types_pos2;
            this.determiner_types_pos2 = determiner_types_pos2;
            this.interjection_types_pos2 = interjection_types_pos2;
            this.nouns_pos2 = nouns_pos2;
            this.verbs_pos2 = verbs_pos2;
            this.prepositions_pos2 = prepositions_pos2;
            this.adjectives_pos2 = adjectives_pos2;
            this.determiners_pos2 = determiners_pos2;
            this.particles_pos2 = particles_pos2;
            this.possessive_pronouns_pos2 = possessive_pronouns_pos2;
            this.personal_pronouns_pos2 = personal_pronouns_pos2;
            this.adverbs_pos2 = adverbs_pos2;
            this.interjections_pos2 = interjections_pos2;
        }
}
```

NGramObj.java

```
package CalcNgram;

public class NGramObj {
	public Double gram2low2 = 0.0;
	public Double gram2high2 = 0.0;
	public Double gram2avg2 = 0.0;
	public Double gram3low2 = 0.0;
	public Double gram3high2 = 0.0;
	public Double gram3avg2 = 0.0;
	public Double gram4low2 = 0.0;
	public Double gram4high2 = 0.0;
	public Double gram4avg2 = 0.0;

	public NGramObj(Double gram2low2, Double gram2high2, Double gram2avg2,
Double gram3low2, Double gram3high2,
			Double gram3avg2, Double gram4low2, Double gram4high2, Double
gram4avg2) {
		this.gram2low2 = gram2low2;
		this.gram2high2 = gram2high2;
		this.gram2avg2 = gram2avg2;
		this.gram3low2 = gram3low2;
		this.gram3high2 = gram3high2;
		this.gram3avg2 = gram3avg2;
		this.gram4low2 = gram4low2;
		this.gram4high2 = gram4high2;
		this.gram4avg2 = gram4avg2;
	}
}
```

Cloud-based textual analysis as a basis for document classification

George R S Weir[1], Kolade Owoeye, Alice Oberacker and Haya Alshahrani
Department of Computer & Information Sciences
University of Strathclyde
Glasgow, UK
{george.weir;kolade.owoeye}@strath.ac.uk, oberackera@googlemail.com, hayaalshahrani@yahoo.com

***Abstract*— Growing trends in data mining and developments in machine learning, have encouraged interest in analytical techniques that can contribute insights on data characteristics. The present paper describes an approach to textual analysis that generates extensive quantitative data on target documents, with output including frequency data on tokens, types, parts-of-speech and word n-grams. These analytical results enrich the available source data and have proven useful in several contexts as a basis for automating manual classification tasks. In the following, we introduce the Posit textual analysis toolset and detail its use in data enrichment as input to supervised learning tasks, including automating the identification of extremist Web content. Next, we describe the extension of this approach to Arabic language. Thereafter, we recount the move of these analytical facilities from local operation to a Cloud-based service. This transition, affords easy remote access for other researchers seeking to explore the application of such data enrichment to their own text-based data sets.**

Keywords-data mining; textual analysis; classification; feature-set; Cloud-service; Posit.

I. INTRODUCTION

As diverse sources of data are increasingly being gathered to create large pools of potential resource, techniques for analysis that facilitate new insights and added value to the raw data are sought with enthusiasm. In this setting, our previous work on text analysis (see Section III) has proven useful as a basis for enrichment of textual data.

The present paper outlines the context of text mining and classification before describing quantitative text analysis using the Posit textual analysis toolset. Thereafter, we detail the application of Posit to the classification of text data. This is followed by an account of Posit as a Cloud-based facility and the potential benefits this affords for distributed third-party application. We conclude with a description of on-going developments to extend the available features in the Cloud-Posit system.

II. TEXT MINING AND CLASSIFICATION

Commonly, two main classes of text categorisation are recognised [1, 2]: text clustering and text classification. The first deals with finding a structure of groups within a given dataset, while the latter is given a set of groups against which each text is to be assigned. Moreover, the task of text classification is subjective in a way that human and machine might disagree on the classification of the data. Text classification can be single-labelled meaning every document is as- signed a single category, or multi-labelled in which case a document can be assigned to several possible categories. This method has the advantage of giving the user the possibility of a final decision to their own subjective opinion as several texts can be closely related to multiple categories. Several applications such as spam filtering, webpage classification, author- ship attribution or genre classification can be decided with text classifications. Among the various machine learning algorithms that have been used to build classifiers, [2] claims the ones that proved most successful in recent years are support vector machines (SVM) and boosting. SVM is a type of classification model, boosting, however, combines the decisions of a group of classifiers in order to achieve a better overall classification [2]. [3] agrees on the effectiveness of SVM, but also points out that this approach might find a suboptimal decision threshold for categories with low occurrences.

However, it remains a challenge to achieve high accuracy for all possible contexts at once, as no algorithm is most effective on all applications [4, 5]. Moreover, the labelling of the documents defines a bottleneck for every supervised classification method as it has to be done manually.

To solve this problem [6] developed a system to hierarchically classify unlabelled data. As already mentioned, classifying data manually is extremely expensive and slows the classification process down. Additionally, it grows to be an inefficient approach as with larger datasets the number of categories can exceed to thousands, of which each needs to be represented by a sufficient number of labelled documents. The

[1] Corresponding author

978-1-5386-7879-4/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCS.2018.00110

system solves this issue by using ontological knowledge and by searching 'pseudo-relevant documents on the Web' [6]. With the ontology it is possible to create a hierarchical model including the context of ancestors among different classes.

[3] compared the accuracy of SVM, k-Nearest Neighbours (k-NN) and Rocchio-Style Prototype Classifier with each other on the Reuters Corpus Vol. 1. Two variants of SVM were used. The first one was trained for each category by using the default settings and the latter tried to find optimal settings to improve results for unbalanced classes for each category and was trained using a leave-one-out cross validation. Results show that the first SVM classifications achieve the best F1 values, followed closely by the second SVM approach. k-NN and Rocchio-Style did not achieve as good results, which underlines the statement made by [2].

Another study by [4] compared results of k-NN, Rocchio-Style and Linear Least Square Fit (LLSF) with each other. Throughout the experiment k-NN achieved the best classification results, with Rocchio and LLSF showing reasonable efficiency. [4] however, states that SVM methods can be used to improve upon the k-NN results. The k-Nearest Neighbour method is a lazy learning method, because few calculations are done during the training phase. During the classification the distances to all training samples have to be calculated to find the k nearest samples, which makes it a lazy learning method and therefore more sensitive to noisy data as it only considers a few samples to make a decision [7, chap. 4].

For classifying text corpora, one has to develop an internal representation for the learning algorithms. The most common approach represents each text as a vector in which every position displays the existence of a word (set of words). Similar techniques do not only acknowledge the existence but also the frequency of words (bag of words) [8, 1]. The representation usually has a large number of features due to the number of unique words in the document. Therefore, it is appropriate to remove irrelevant features to optimise the prediction [5]. However, it needs to be shown if feature selection plays an important role when using the Posit toolset, as the number of features that can be extracted from the computed quantitative data do not expand the runtime of the learning algorithms drastically. As it is suggested in multiple papers [5, 8, 9] feature selection can improve the performance of classifiers.

[8] for example, points out that words with low frequencies can be neglected as well as so called stop words, such as 'a' and 'or'. However, for every approach one needs to bear in mind the possibly varying size of documents as the occurrences need to be normalised over the size of text. [5] suggest that the most suitable classification performance metrics is the receiver operating characteristic (ROC), which plots sensitivity against 1−specificity. The area under curve (AUC) can then be used to differentiate between perfect classification (AUC=1), classification by chance (AUC=0.5) and inverse classification (AUC=0). The advantage of this metric is its insensitivity to unbalanced categories.

In the field of computational linguistics n-grams are defined as a sequence of characters or words of length n. The Posit tool makes it possible to extract word grams of length 2, 3 and 4 including their frequency. Statistical features about word n-grams have been appropriated for text classification by [10]. The n-gram language model is handled in a similar way to a Naïve Bayes model. Each category is trained with a language model and every document can be evaluated on each of those models to decide to which it agrees the most. In this experimental paper it was shown that statistical data of n-grams can be used for a chain augmented Naïve Bayes classifier. An optimal size for n-grams can be found to improve the classification of documents. This underlines the likely importance of n-grams data, as it seems to be a source for improving upon classification accuracy.

III. Quantitative Text Analysis

The most popular approach to text classification represents each text as a vector of word occurrences (set or bag of words) [4, 8, 11, 1]. One way of modelling such a vector is denoting the occurrence of a word by setting the position to 1 and otherwise to 0. There are other models which also include the frequency of which a word appears in the text which can be of great importance to the classification. However, these approaches require a lot of computational time and optimisation, for example, using feature selection.

Another approach, which enriches the representation of texts for machine learning models is described in this section. Instead of representing a text with its words, we may calculate quantitative and statistical values for a text and use these features as a basis for classification.

The Posit Text Profiling Toolset [12, 13] offers a thorough quantitative analysis of an arbitrarily large text corpus with highly customisable features. Posit applies a Part-of-Speech tagger and outputs statistical details of the text content in terms of individual words (tokens) and word types. This frequency data is also provided for specific parts of speech, including frequency ordered details of each specific word in an analysed text. Significantly, in affording a basis for comparison between samples of text data, Posit's summary details can be employed as a feature set for use in classification of textual data.

The summary data output from Posit includes values for total words (tokens), total unique words (types), type/token ratio, number of sentences, average sentence length, number of characters, average word length, noun types, verb types, adjective types, adverb types, preposition types, personal pronoun types, determiner types, possessive pronoun types, interjection types, particle types, nouns, verbs, prepositions, personal pronouns, determiners, adverbs, adjectives, possessive pronouns, interjections, and particles. This comprises 27 features in all. An example of such output is shown in Figure 1.

When analysing texts using Posit, output is generated at several levels of detail. Of these, the summary level is the most general, e.g., the total number of verbs, nouns, adjectives, etc. (Figure 1). Two more detailed levels of output are provided: an intermediate (aggregate) part-of-speech analysis, and a finely detailed word types against parts-of-speech account.

At the intermediate level, frequency data is provided for the contents of the analysed text in terms of specific parts-of-speech, for example, types of verb: the base form of verbs, the gerund form, the past tense form, the past participle form, the 3rd person present form, the present tense (non-3rd person) form and the

modal auxiliary form. An illustration of this intermediate level is shown in Figure 2.

```
NUMBER OF TOKEN TYPES
4757 :noun_types
3099 :verb_types
1382 :adjective_types
531 :adverb_types
130 :preposition_types
54 :possessive_pronoun_types
54 :personal_pronoun_types
39 :particle_types
30 :determiner_types
8 :interjection_types

NUMBER OF POS TYPES
35928 :verbs
29860 :nouns
28510 :prepositions
24508 :possessive pronouns
24508 :personal pronouns
18234 :determiners
12410 :adverbs
9788 :adjectives
1438 :particles
188 :interjections
```

Figure 1: Example Posit summary output

At the fine detail level, frequency data is provided for each word in terms of part-of-speech type, for example, the number of occurrences of every word that is a verb of gerund form. An illustration of this fine detail level is shown in Figure 3.

```
 804 verbs_base_form.txt
 512 verbs_gerund_form.txt
 743 verbs_past_form.txt
 907 verbs_past_participle_form.txt
 179 verbs_present_3rd_form.txt
 218 verbs_present_not3rd_form.txt
  13 modal_aux.txt
3376 total
```

Figure 2: Example Posit aggregate output

```
39 being/vbg
32 growing/vbg
29 living/vbg
26 going/vbg
22 coming/vbg
18 increasing/vbg
17 making/vbg
17 leading/vbg
17 fighting/vbg
17 beginning/vbg
16 writing/vbg
16 working/vbg
15 becoming/vbg
14 spreading/vbg
12 flying/vbg
11 developing/vbg
10 seeking/vbg
 9 taking/vbg
 9 speaking/vbg
 9 having/vbg
 8 raiding/vbg
 8 learning/vbg
 8 following/vbg
 8 conquering/vbg
 8 changing/vbg
```

Figure 3: Example Posit fine detail output

IV. CLASSIFICATION USING POSIT

Since the basis of any classification is the 'matching' of features present across data samples, the feature set produced when texts are analysed using Posit provides a ready characterization of texts that can be contrasted for the purpose of classification. In our classification work to date, we have used only the summary output produced by Posit as the basis for a feature set that characterises each data sample.

To this end, [14] applied the Posit tool to generate summary output for data retrieved by the Terrorism and Extremism Network Extractor (TENE) web crawler [15]. This data had been manually classified into the categories 'pro-extremist', 'neutral' and 'anti-extremist'. Posit was applied in order to provide the quantitative syntactic features that 'enrich' the information given by the text corpus.

When used for classification with the J48 algorithm, the Posit approach matched 91.4% of the manually classified webpages correctly. An improved result of 95.3% correctly classified texts was accomplished with a Random Forest algorithm. These results led us to believe that through application of Posit analysis we could provide enriched insight on the content of textual data and afford effective classification of such data. The advantage of a quantitative approach, opposed to a vector representation of the existence of words in the text (bag of words), is that the number of features is much lower. Instead of dealing with millions of features [16], the Posit tool extracts 27 distinct values. Further research is underway to explore the extension of the Posit feature set, including frequency data on word combinations (n-grams) and frequency ratios (e.g., ratio of common nouns to proper nouns).

Following this effective application of Posit to the classification of extremist Web content, a similar approach was adopted with a dataset containing drug related texts from the Dark Web. Some of this data were manually classified as drugs-related positive or negative. A total of 1,245,410 texts were included in the initial set and this was reduced to 798,684 textual data items after cleaning. In the final data set, 91,088 items were pre-classified as drugs-related or not drugs-related.

A series of experiments using Posit-based classification were performed on this Dark Web data set, aiming to match against the training set provided by the manually classified subset of data. The results (Table 1) show that the K Nearest Neighbour algorithm (where k=1) gave the best performance (with an F-measure of 0.995), closely followed by the J48 algorithm (with an F-measure of 0.99).

TABLE I. CLASSIFICATION RESULTS FOR DRUGS-RELATED DATA SET

Algorithm	Precision	Recall	F1
J48	0.99	0.99	0.99
kNN1	0.995	0.995	0995

V. POSIT IN THE CLOUD

In order to expand the scope and range of Posit application in textual classification tasks, we are developing a full-featured Cloud-based implementation. This facilitates third-party access to the Posit analysis of plain text data sets.

The Cloud-Posit system is being developed in four phases. In the initial Phase One version, third-parties may access an interactive Cloud-based Posit facility that affords the upload of multiple data files in a set. After file-upload, selecting the 'run Posit' option, results in Posit being executed sequentially on each file in the data set. The analysis output for each file is output as a separate folder and the complete set of analysis folders is compressed into a single file archive and downloaded to the remote Web client. Figure 4 illustrates the interactive Phase One Web interface to Cloud-Posit.

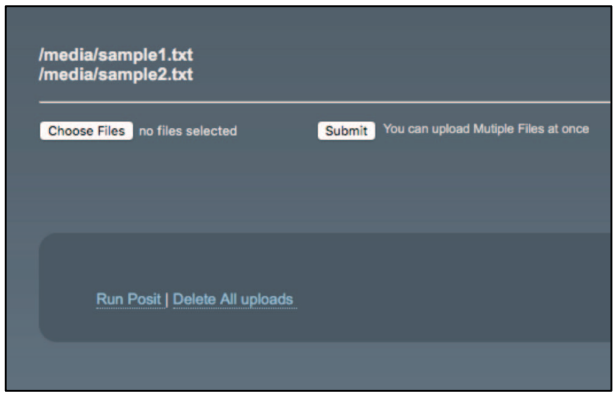


Figure 4: Cloud-Posit interactive facility

This *Phase One* facility will prove convenient for initial third-party experimentation with Posit and is well-suited to small data sets. *Phase Two* of Cloud-Posit will additionally afford API access, without need for user interaction. Through this version, remote users may directly connect, upload multiple files for analysis and retrieve the result files directly, for further local processing. The expectation is that the interactive mode will be used initially by researchers seeking to 'train' their classification model, e.g., using an appropriate classification algorithm and cross-validation techniques. Once an effective model has been constructed, the bulk of analysis data would then be processed via the non-interactive API of Cloud-Posit.

The *Phase Three* version of Cloud-Posit will supplement the default Posit set of 27 features with a multiword (ngram) frequency analysis. As indicated above, ngram data is likely to provide useful additional features for use in classification. In due course, the aim is return not only raw ngram data on submitted samples, but ngram ratios for high and low frequency ngrams (e.g., against the Google ngram corpus [17]).

The planned *Phase Four* version of Cloud-Posit, will deploy parallel developments in the use of Posit for Arabic textual analysis. This applies a customized version of the Posit system and an Arabic part-of-speech tagger with output that accounts for Arabic-specific language characteristics. In addition to the standard feature set, Arabic Posit supports ngram analysis for Arabic texts. Figure 5 shows sample Arabic bigram data from Posit. For *Phase Four,* the Posit feature analysis for Arabic and the Arabic ngram analysis for 2, 3 and 4-grams will be added to Cloud-Posit.

Figure 5: Example Arabic bigrams extract from Posit

VI. Further Applications

In addition to the use of Posit in supervised learning applications, there are further roles that it can play in corpus comparisons. For example, the detailed data analysis provided by Posit allows for contrastive review of two or more documents (or document sets). Such an approach was employed as a basis for gauging the similarity of grammatical approach across several generations of textbooks used to teach English in Japan [18]. We expect that the extended insights afforded by the aggregate and fine level details of Posit analysis will also find a role in classification. For this reason, our implementation of Cloud-Posit generates all three levels of quantitative textual analysis.

For any context in which quantitative analysis may shed light on textual data Posit can support such insights. By making this facility available through a Web service as Cloud-Posit, we aim to extend this utility to the academic and research community.

References.

[1] F. Sebastiani. Machine learning in automated text categorization. ACM Comput. Surv., 34(1):1–47, Mar. 2002.

[2] F. Sebastiani. Text categorization. In Encyclopedia of Database Technologies and Applications, pages 683–687. IGI Global, 2005.

[3] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Bench- mark Collection for Text Categorization Research. Journal of Machine Learning Research, 5:361–397, 2004. doi: 10.1145/122860.122861.

[4] B. Harish, R. M. Hegde, N. Neeti, and M. Meghana. An Empirical Study on Various Text Classifiers. Advanced Materials Research, pages 587–593, 2012. doi: 10.1109/MSR.2017.60.

[5] Y. Aphinyanaphongs, L. D. Fu, Z. Li, E. R. Peskin, E. Efstathiadis, C. F. Aliferis, and A. Statnikov. A comprehensive empirical comparison of modern supervised classification and feature selection methods for text categorization. Journal of the Association for Information Science and Technology, 65(10):1964–1987, 2014. doi: 10.1002/asi.23110.

[6] V. Ha-Thuc and J.-M. Renders. Large-scale hierarchical text classification without labelled data. Proceedings of the fourth ACM international con-

ference on Web search and data mining - WSDM '11, page 685, 2011. doi: 10.1145/1935826.1935919.

[7] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2011.

[8] G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research, 3:1289–1305, 2003. doi: 10.1162/153244303322753670.

[9] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of the 10th European Conference on Machine Learning ECML '98, pages 137–142, 1998. doi: 10.1007/BFb0026683.

[10] F. Peng and D. Schuurmans. Combining Naive Bayes and n-Gram Language Models for Text Classification. Computer, pages 335–350, 2003. doi: 10.1007/3-540-36618-0 24.

[11] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. Data Mining: A Knowledge Discovery Approach. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[12] G. R. S. Weir. The Posit Text Profiling Toolset. Proceedings of PAAL2007, 2007.

[13] G. R. S. Weir. Corpus profiling with the posit tools. In Proceedings of the 5th Corpus Linguistics Conference. University of Liverpool, 2009.

[14] G. R. S. Weir, E. Dos Santos, B. Cartwright, and R. Frank. Positing the problem: Enhancing classification of extremist web content through textual analysis. 2016 IEEE International Conference on Cybercrime and Computer Forensic, ICCCF 2016, pages 67–69, 2016. doi: 10.1109/ICCCF.2016.7740431.

[15] M. Bouchard, K. Joffres and R. Frank. Preliminary analytical considerations in designing a terrorism and extremism online network extractor. InComputational models of complex systems, pages 171-184, 2014, Springer, Cham.

[16] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. Artificial Intelligence, 97(1-2), pages 245–271, 1997. ISSN 00043702. doi: 10.1016/S0004-3702(97)00063-5.

[17] K. Wang, C. Thrasher, E. Viegas, X. Li and B. J. Hsu. An overview of Microsoft Web N-gram corpus and applications. InProceedings of the NAACL HLT 2010 Demonstration Session 2010 Jun 2, pages. 45-48, 2010, Association for Computational Linguistics.

[18] G. R. S. Weir, and T. Ozasa. Learning from Analysis of Japanese EFL Texts. Educational Perspectives, Journal of the College of Education/University of Hawaii at Manoa, 43 (1 & 2). pp. 56-66, 2010. ISSN 0013-1849

Appendix C:

detailed classifier results:

Figure 7. 11 results of J48.

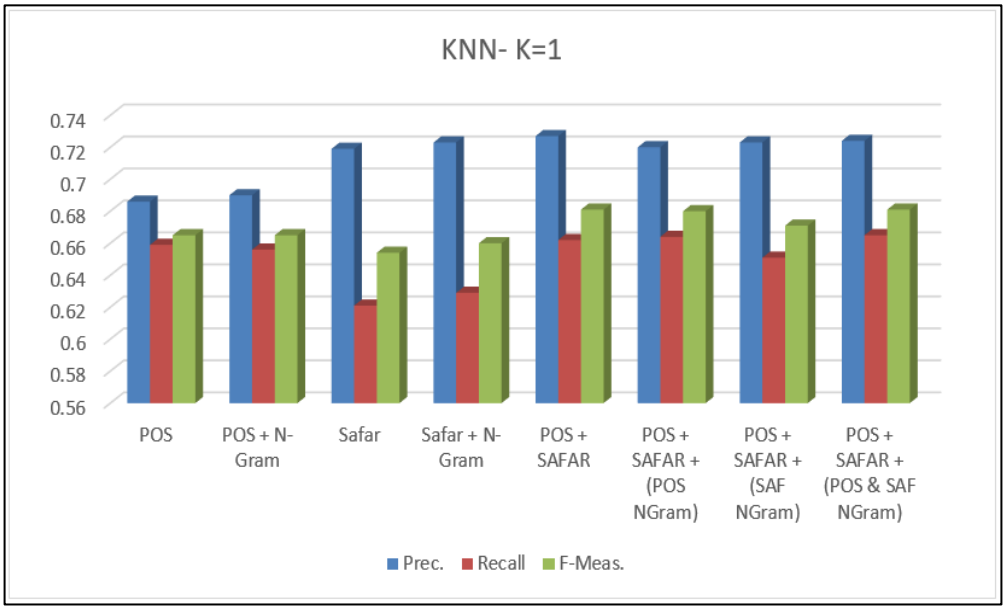


Figure 7. 13 results of KNN k=1

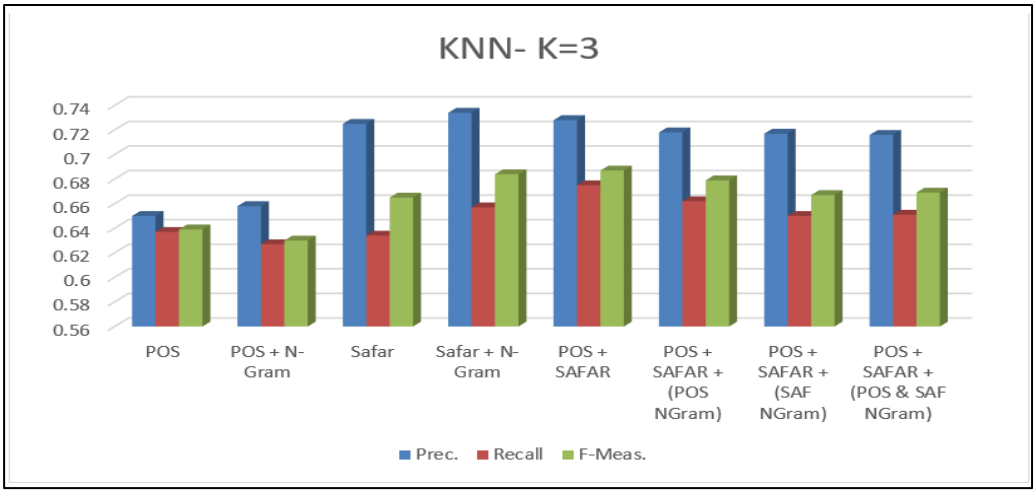


Figure 7. 14 results of KNN k=3.

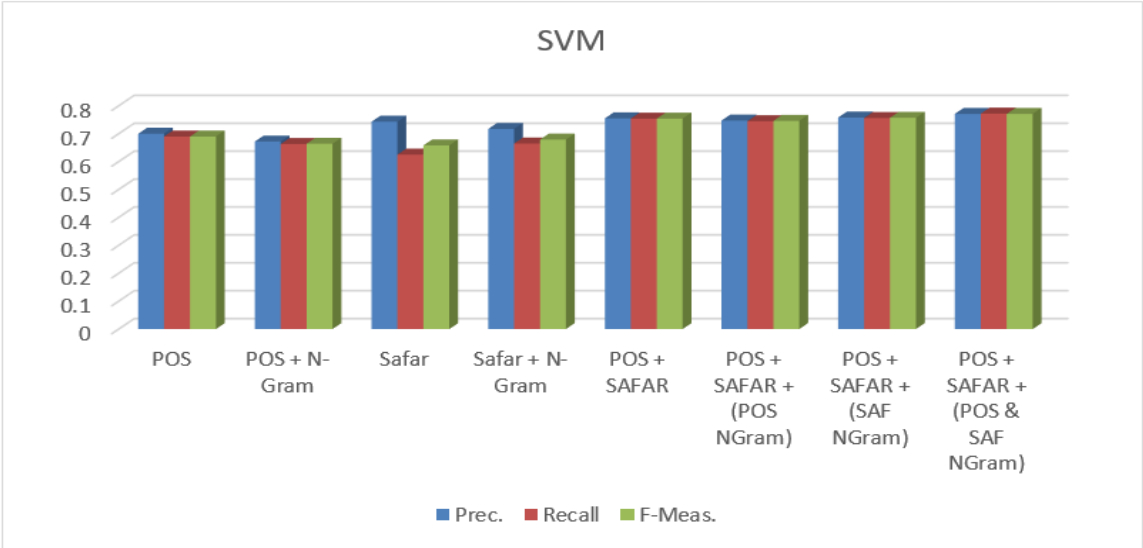


Figure 7. 15 results of SVM.

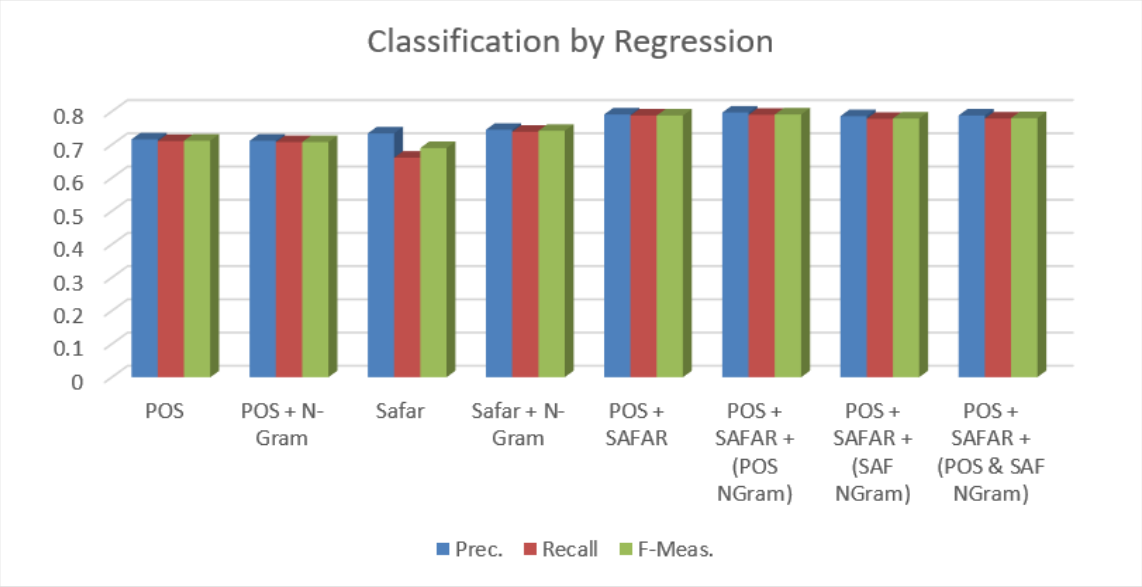


Figure 7. 16 results of classification by regression.

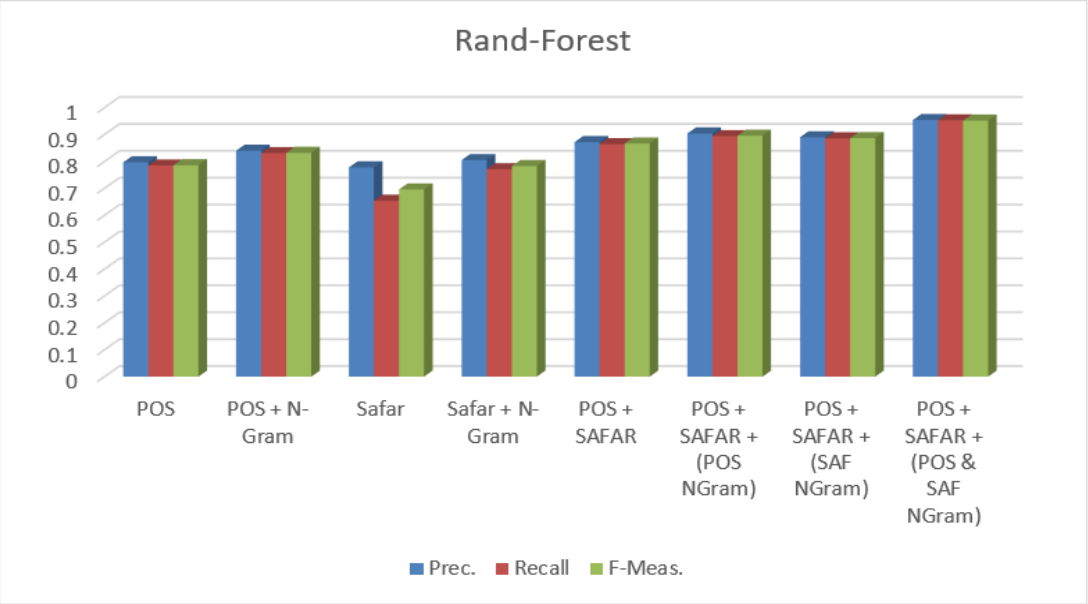


Figure7.17 results of RANDOM-FOREST.

Dataset		J48	NaïveBayes	IBk_3	SVM	Random Forest	RF Via Regression
POS	**Precision**	0.703	0.708	0.650	0.700	0.797	0.716
	Recall	0.660	0.386	0.637	0.690	0.785	0.711
	f-Measure	0.672	0.426	0.639	0.690	0.786	0.712
POS + N-Gram	**Precision**	0.693	0.676	0.658	0.672	0.840	0.712
	Recall	0.674	0.463	0.627	0.663	0.831	0.708
	f-Measure	0.680	0.514	0.630	0.664	0.832	0.708
SAFAR	**Precision**	0.671	0.615	0.725	0.743	0.778	0.734
	Recall	0.616	0.314	0.634	0.625	0.654	0.661
	f-Measure	0.634	0.326	0.665	0.658	0.696	0.690
SAFAR + N-Gram	**Precision**	0.673	0.674	0.734	0.717	0.805	0.745
	Recall	0.647	0.438	0.657	0.665	0.771	0.739
	f-Measure	0.652	0.495	0.684	0.679	0.783	0.742
POS + SAFAR	**Precision**	0.757	0.711	0.728	0.755	0.872	0.791
	Recall	0.742	0.380	0.675	0.754	0.864	0.788
	f-Measure	0.745	0.420	0.687	0.754	0.867	0.788
POS + SAFAR + POS N-Gram	**Precision**	0.764	0.691	0.718	0.747	0.904	0.797
	Recall	0.752	0.428	0.662	0.745	0.894	0.790
	f-Measure	0.754	0.488	0.679	0.746	0.896	0.791
POS + SAFAR + SAFAR N-Gram	**Precision**	0.776	0.692	0.717	0.758	0.890	0.786
	Recall	0.768	0.435	0.650	0.756	0.886	0.778
	f-Measure	0.766	0.495	0.667	0.757	0.887	0.779
POS + SAFAR + POS + SAFAR N-Gram	**Precision**	0.763	0.691	0.716	0.771	**0.954**	0.788
	Recall	0.750	0.456	0.651	0.772	**0.953**	0.779
	f-Measure	0.752	0.514	0.669	0.771	**0.952**	0.780

Table 7. 33 complete experiment results for all unseen datasets and all classifier

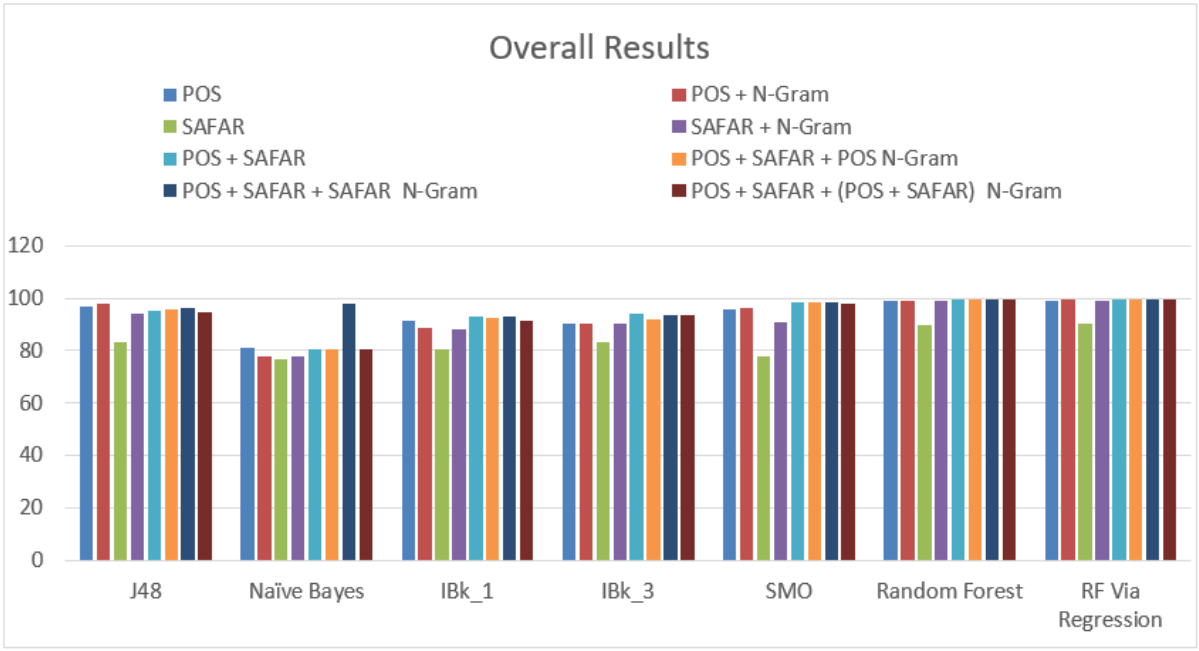


Figure 7. 18 complete experiment results for all unseen datasets and all classifier.

As figure 7.18 shows, the rows represent the deferent datasets, and the column represents the deferent classifier. Each classification process for any datasets results in three measures to determine the best results. The three measures are precision, recall, and f-measure.

The red font marked blue cells represents the best row-column best results.

It is clear that the best classifier achieved by using "classification using constructing a forest of random trees," in other words, "Classification using random forest." All datasets are giving best results in row results (relative to deferent classifier) but the best results archived by the following datasets:

- POS + SAFAR + (POS N-Gram + SAFAR N-Gram)

This result shows how POSIT and SAFAR toolkits summary results reinforce each other to produce the best prediction precision. Moreover, it shows how 2-3-4 gram information helps to enhance prediction results.

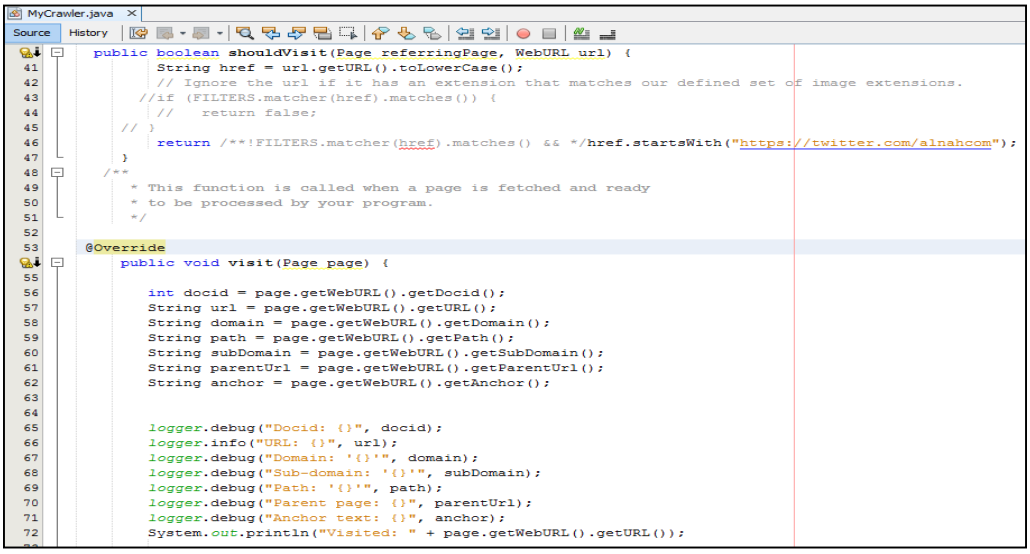


Figure 1 Collecting Arabic extremist dataset using crawler.

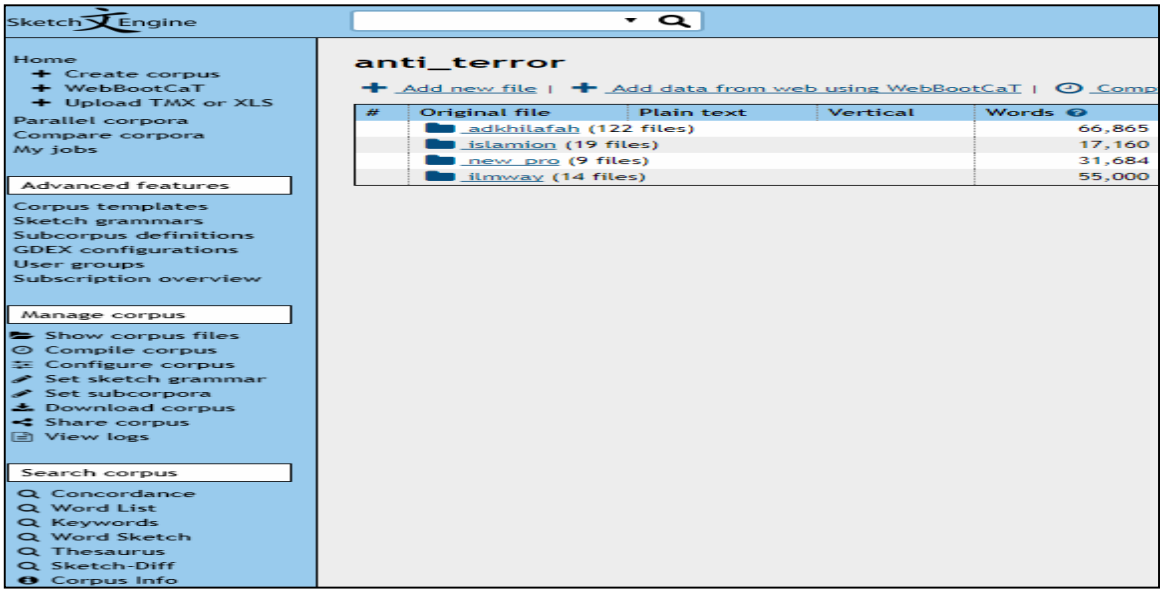


Figure ٢ Sketch engine Screen

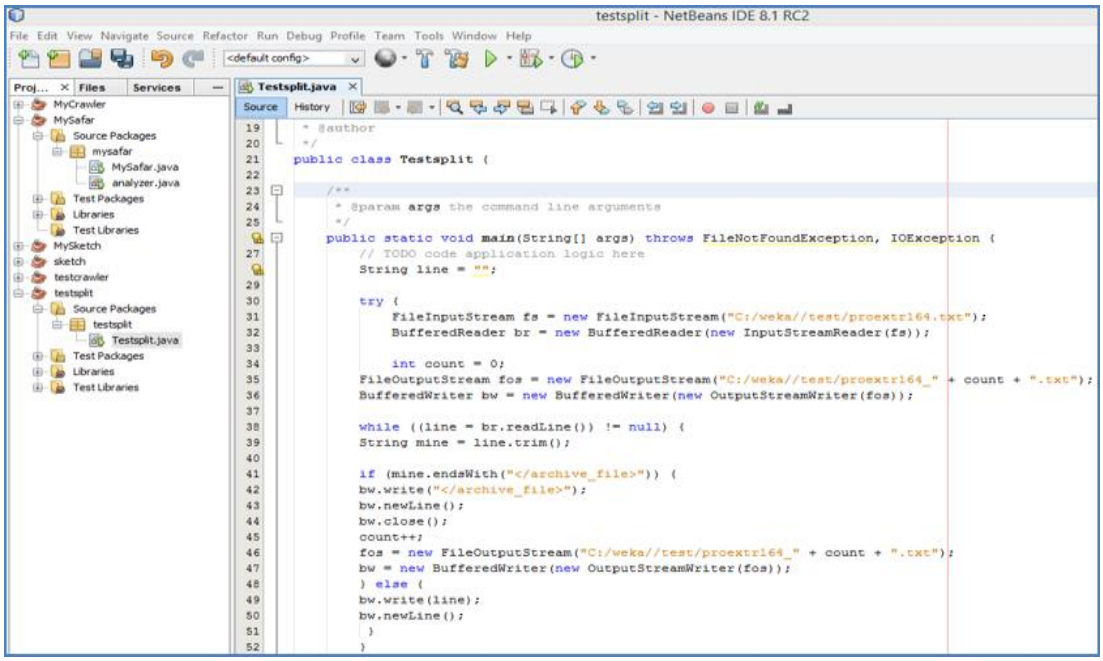


Figure ٣ Java splitter class

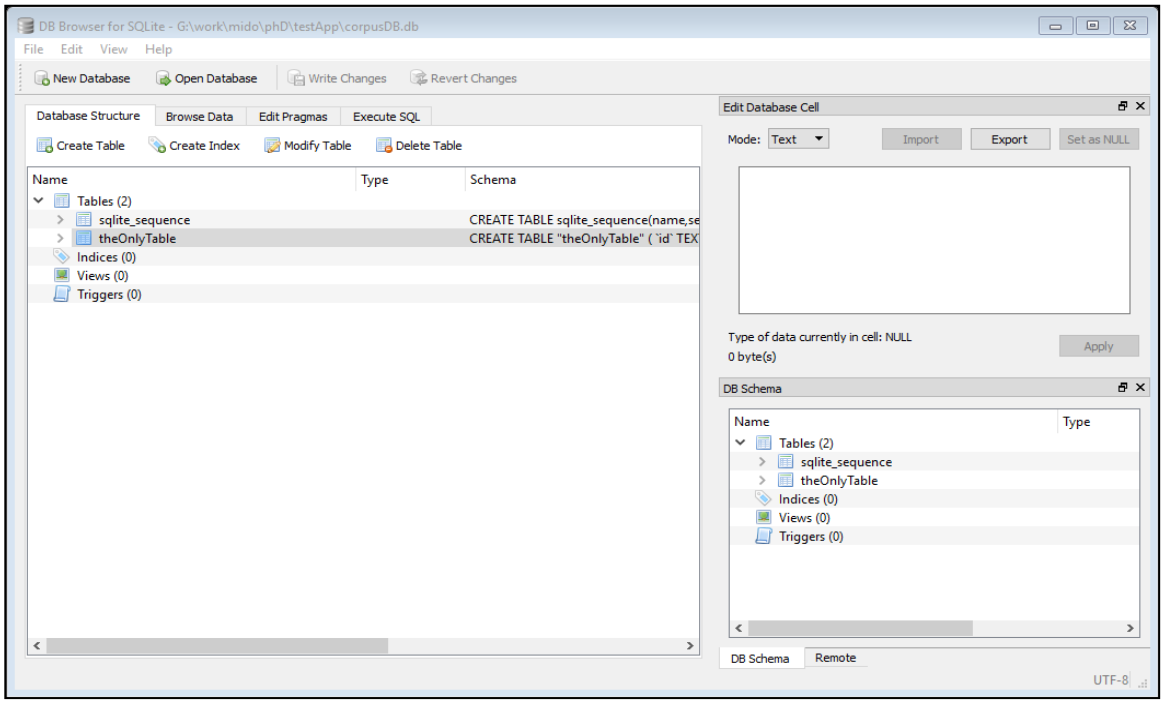


Figure 4 خطأ! لا يوجد نص من النمط المعين في المستند. **SQLite database engine**

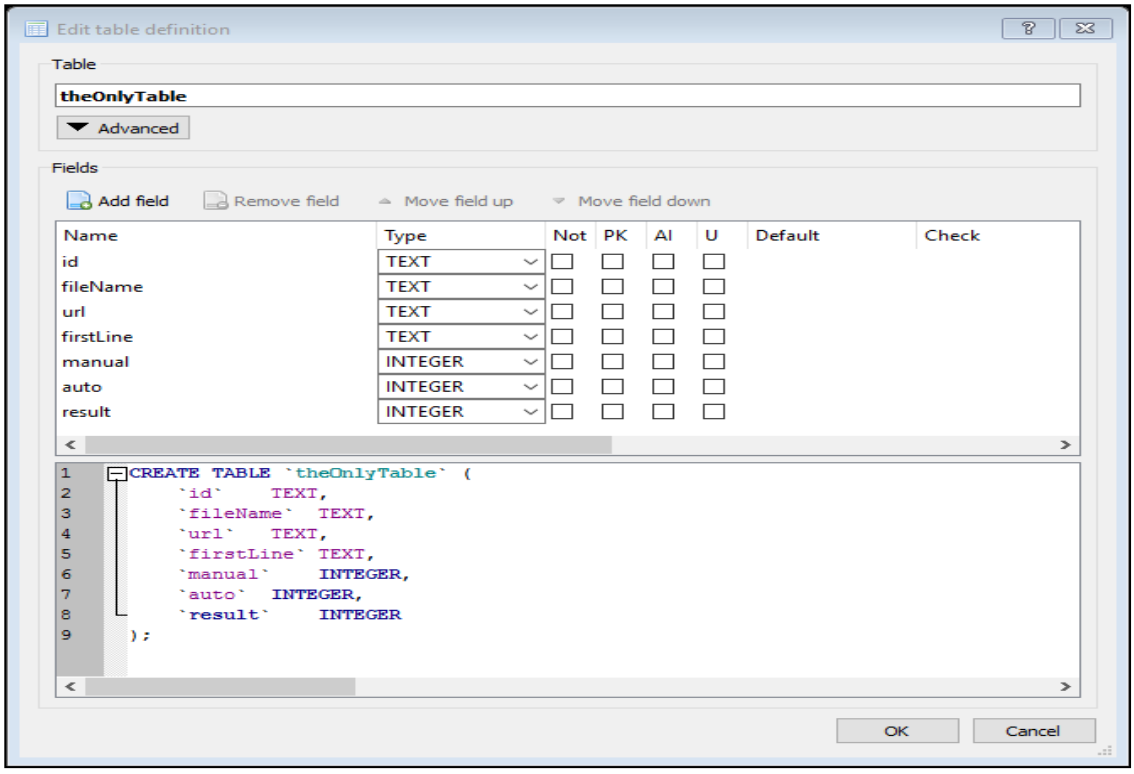


Figure 5 Table created to contain the data.

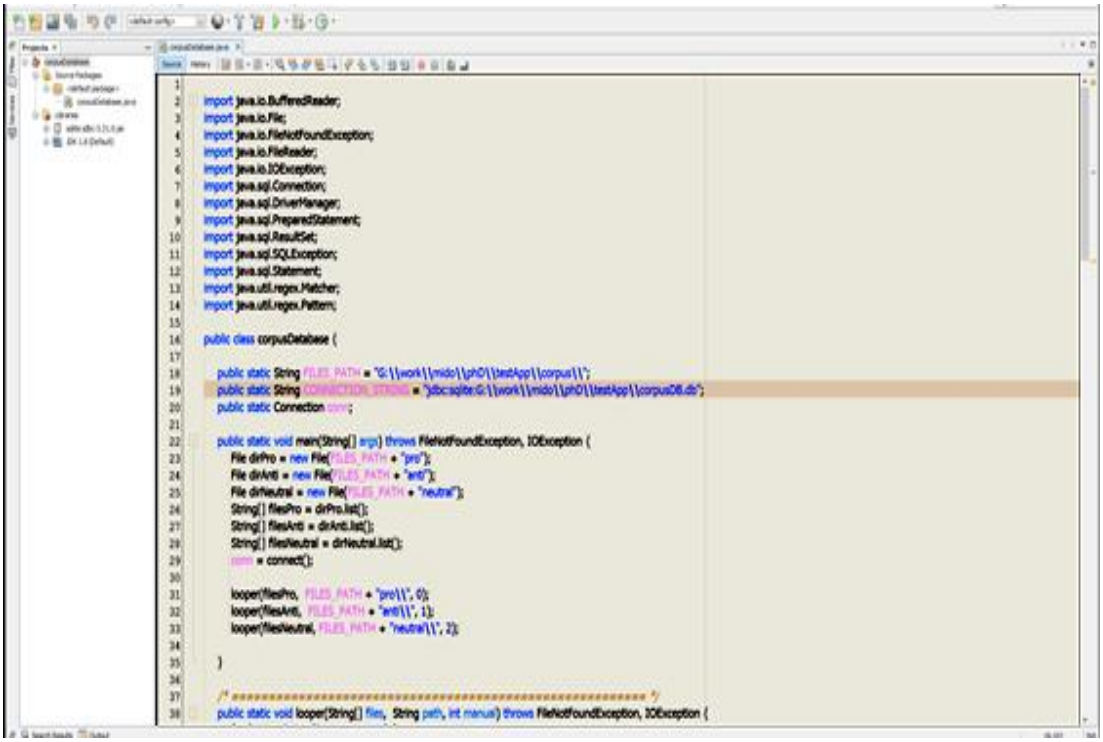


Figure 6 Java code to extract data

```
37      /* ============================================================ */
38      public static void looper(String[] files,  String path, int manual) throws FileNotFoundException, IOException {
39          for (int i = 0; i < files.length; i++) {
40              BufferedReader reader;
41              reader = new BufferedReader(new FileReader(path + files[i]));
42              String line = reader.readLine();
43              if (line != null) {
44                  String id = extractIdFromLine(line);
45                  String url = extractUrlFromLine(line);
46                  insert( id, files[i], url, line, manual, 0, 0);
47              }
48              reader.close();
49          }
50      }
51
52      /* ============================================================ */
53      public static void insert(String id, String fileName, String url,String firstLine, int manual, int auto, int result) {
54          String sql = "INSERT INTO theOnlyTable(id,fileName,url,firstLine, manual, auto, result) VALUES(?,?,?,?,?,?,?)";
55
56          try {
57              PreparedStatement pstmt = conn.prepareStatement(sql);
58              pstmt.setString(1, id);
59              pstmt.setString(2, fileName);
60              pstmt.setString(3, url);
61              pstmt.setString(4, firstLine);
62              pstmt.setInt(5, manual);
63              pstmt.setInt(6, auto);
64              pstmt.setInt(7, result);
65              pstmt.executeUpdate();
66          } catch (SQLException e) {
67              System.out.println(e.getMessage());
68          }
69      }
70
```

Figure 7 Looper function.

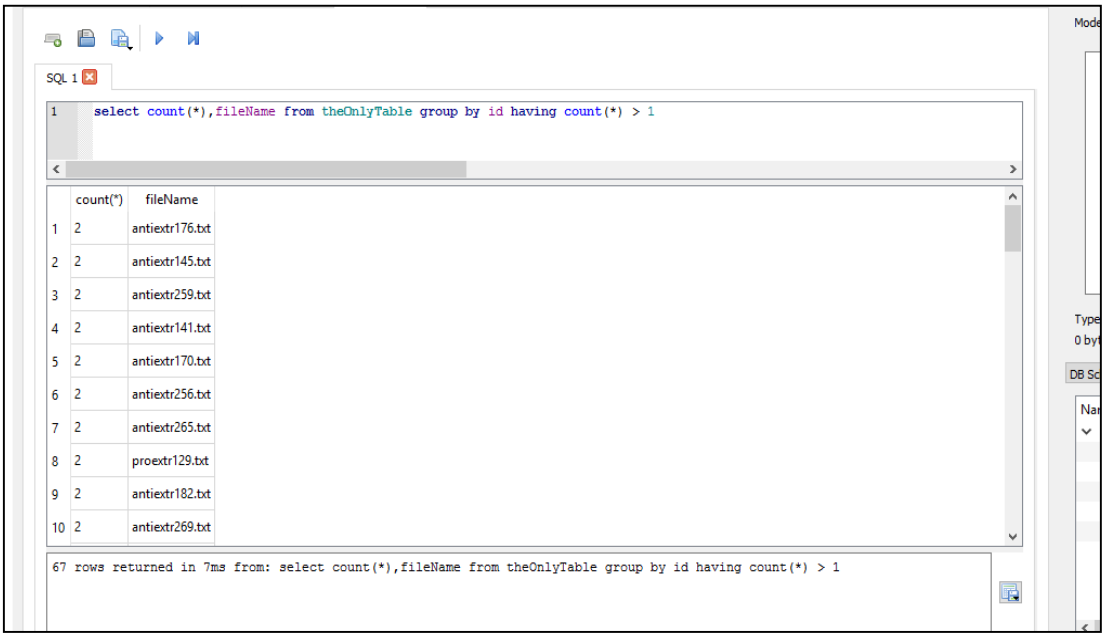


Figure 8 Results of files have same ID.

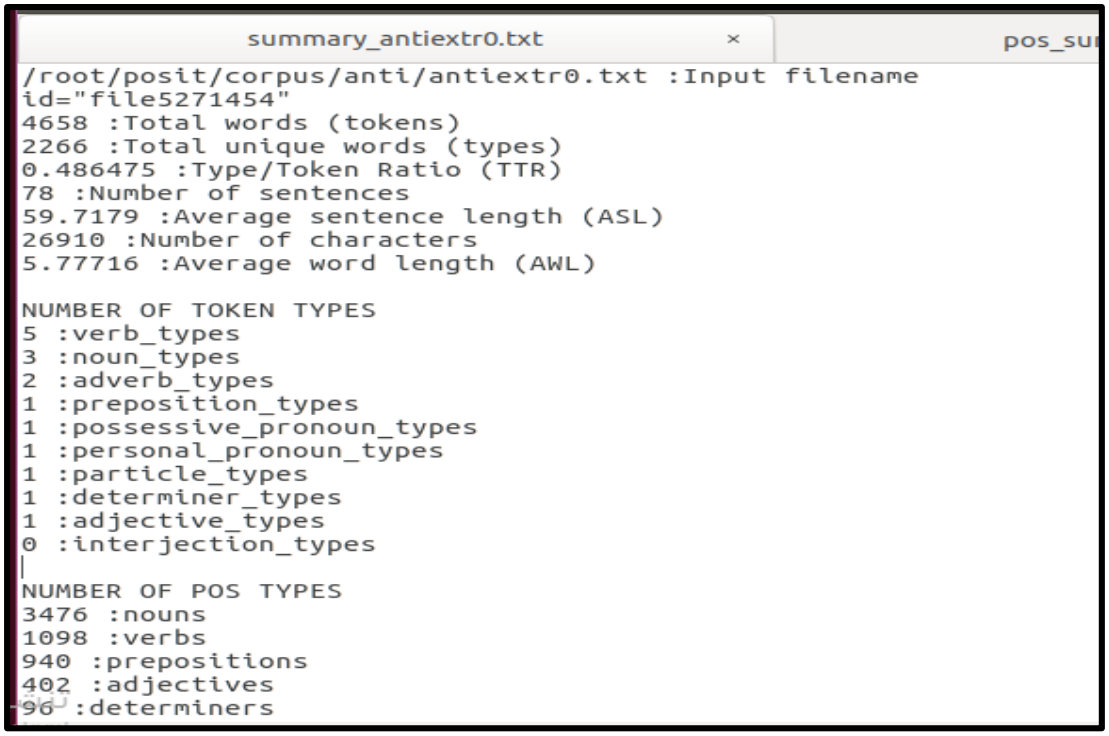


Figure 9 Example of Posit Summary output

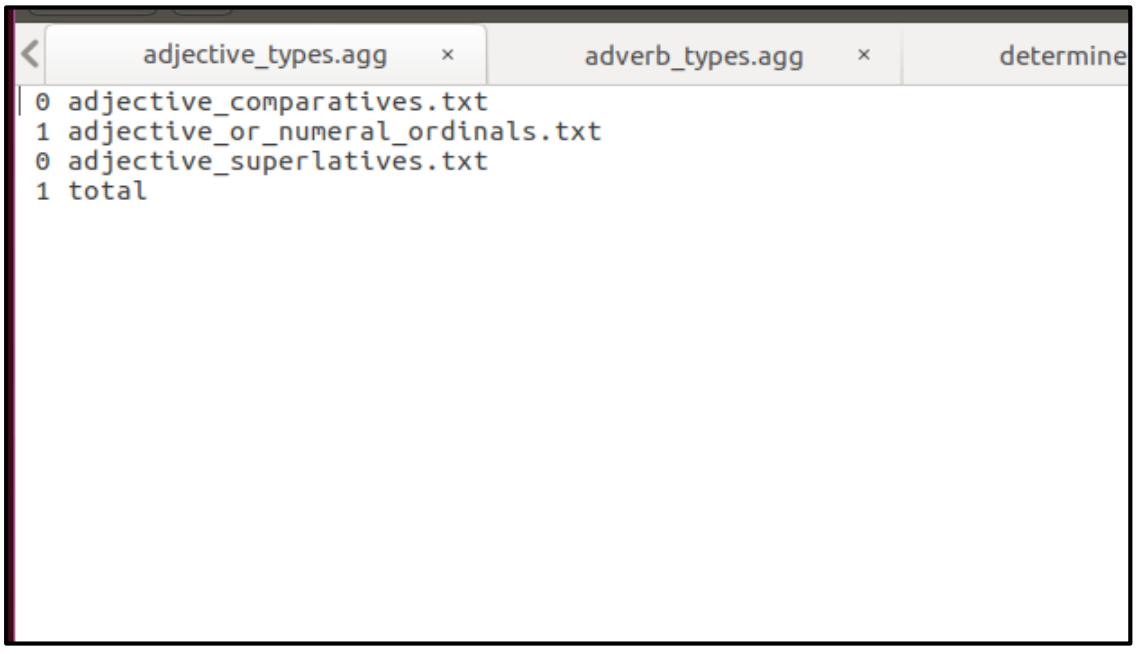


Figure 10 example of the aggregate output

```
# 1. Create word count and token frequency list
tokenize.sh $*
# 2. tag using the script tag_text.sh
echo 'Tagging input file... (may take some time)'
tag_text.sh $*
# 3. tokenize using the script pos_tokenize.sh
echo 'Tokenizing tagged file...'
pos_tokenize.sh tagged_text.txt
# 4. extract POS counts to individual files using the script pos_count.sh
echo 'Extracting POS counts to individual files...'
pos_count.sh pos_tokens.txt
# 5. Proceed to analyse results
tcount.sh
# 6. Create results summary
cd results
echo -n $* > summary.txt
echo -n ' :Input filename' >> summary.txt
echo >> summary.txt
cat type_token_count.txt  >> summary.txt
echo >> summary.txt
echo 'NUMBER OF TOKEN TYPES'>> summary.txt
cat pos_summary.dat >> summary.txt
echo >> summary.txt
echo 'NUMBER OF POS TYPES'>> summary.txt
# Replace commas with colons (for uniformity of output)
sed 's/,/ :/' pos_totals.dat >> summary.txt
echo
echo 'RESULTS SUMMARY'
cat summary.txt
```

Figure 11 The original Pos_all script file

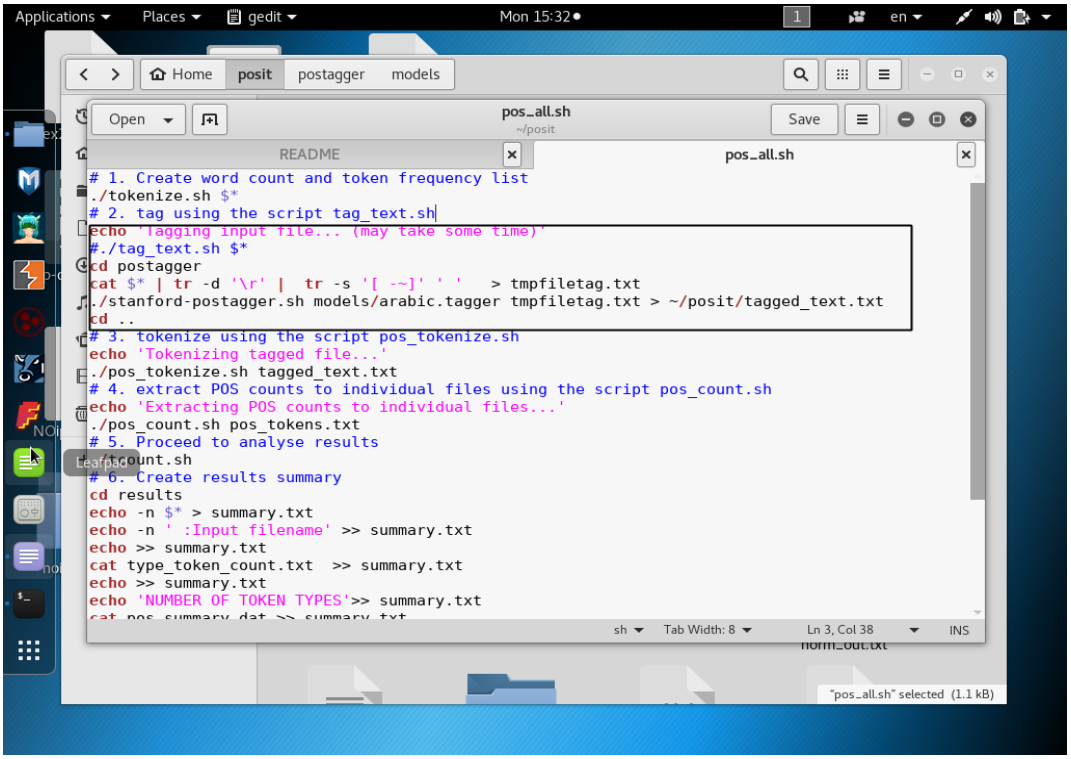


Figure 12 The modified pos_all script file

Figure ١٣ original Arabic text file

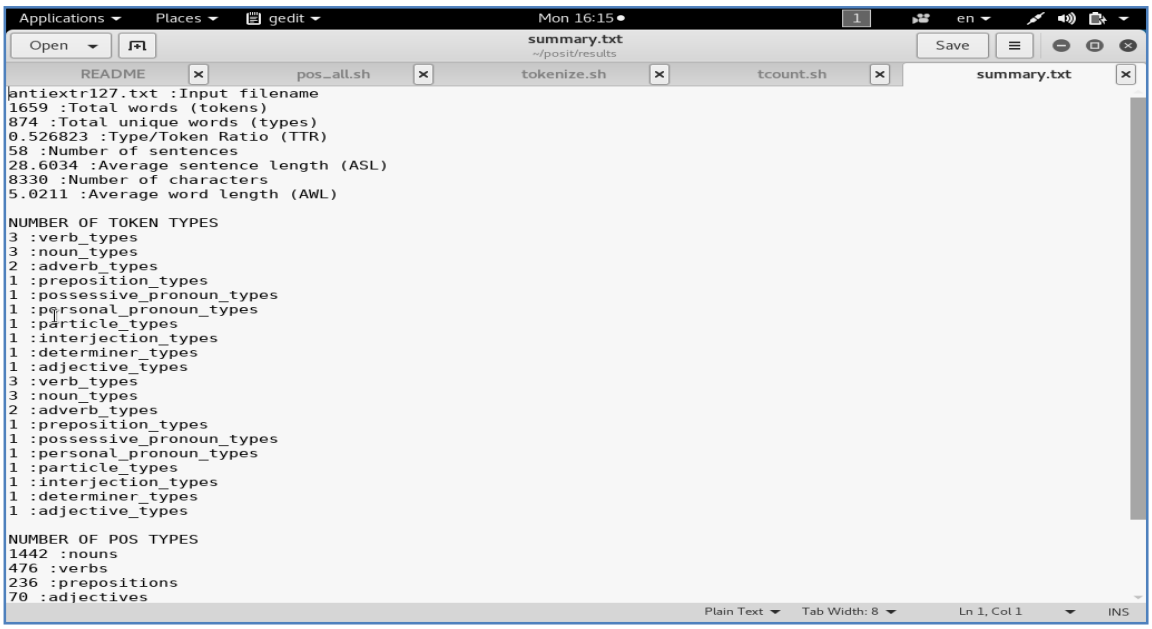


Figure ١٤ summary results from Posit

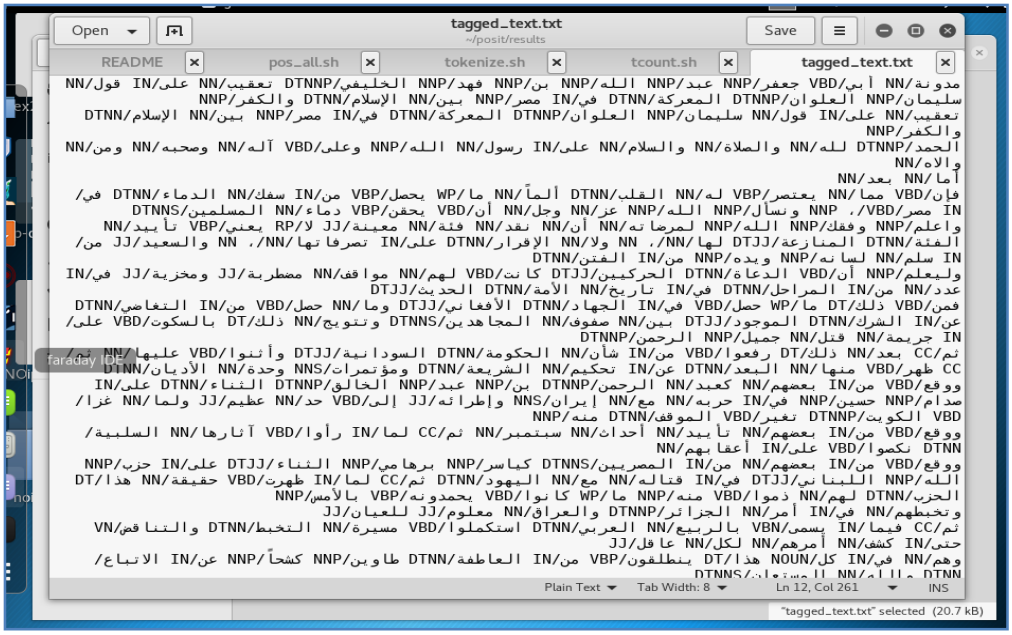


Figure 15 Tagged text file.

```
# Create a file for ngram frequencies (where n = 2 to 4)

new_dir="`basename $1 .txt`_ngram_results"
mkdir -p $new_dir
echo 'Extracting 2-gram count...'
cat $* | tr -cs '[:alnum:]' '[\012*]' | tr A-Z a-z > tmpfile1
tail --lines=+2 tmpfile1 > tmpfile2
paste tmpfile1 tmpfile2 | sort | uniq -c | sort -rn > 2-grams.txt
echo 'Extracting 3-gram count...'
paste tmpfile1 tmpfile2 > tmpfile3
tail --lines=+3 tmpfile1 > tmpfile4
paste tmpfile3 tmpfile4 | sort | uniq -c | sort -rn > 3-grams.txt
echo 'Extracting 4-gram count...'
tail --lines=+4 tmpfile1 > tmpfile5
paste tmpfile3 tmpfile4 > tmpfile6
paste tmpfile6 tmpfile5 | sort | uniq -c | sort -rn > 4-grams.txt
rm -f tmpfile1 tmpfile2 tmpfile3 tmpfile4 tmpfile5 tmpfile6
echo "Results output to directory: $new_dir"
mv 2-grams.txt 3-grams.txt 4-grams.txt $new_dir
```

Figure 16 Original n-gram.sh

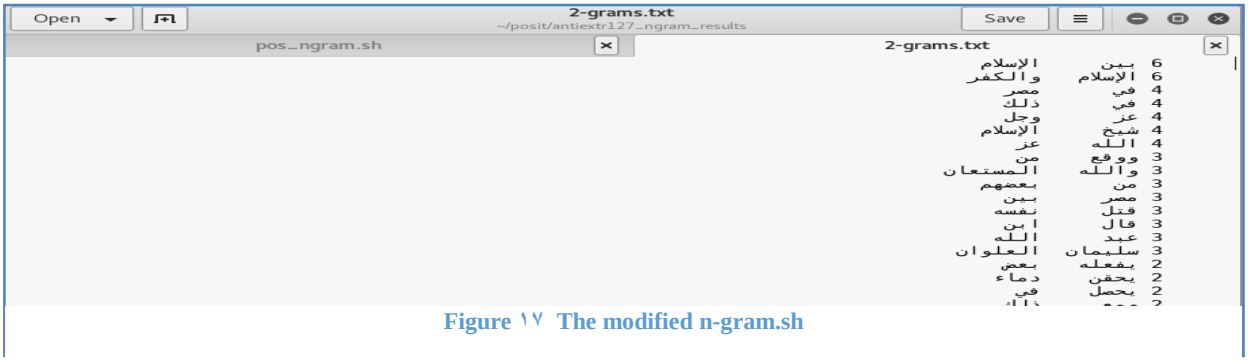


Figure ١٧ The modified n-gram.sh

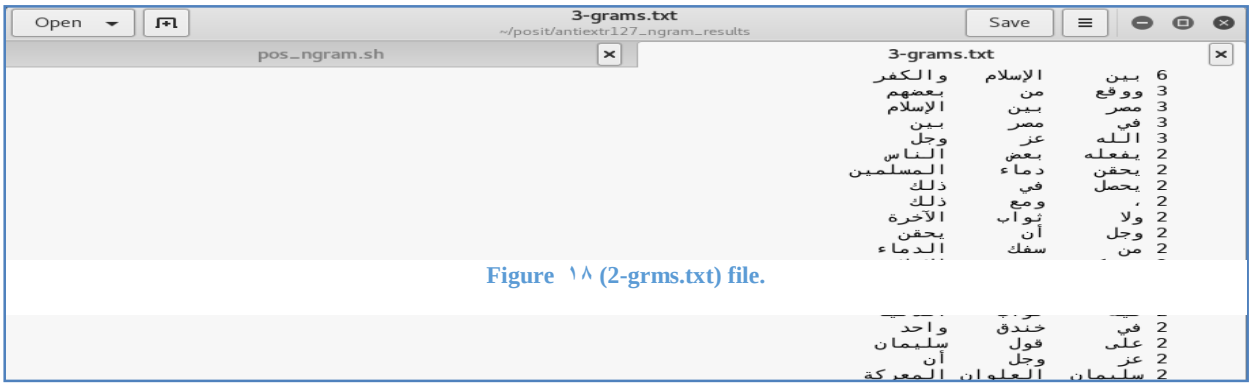


Figure ١٨ (2-grms.txt) file.

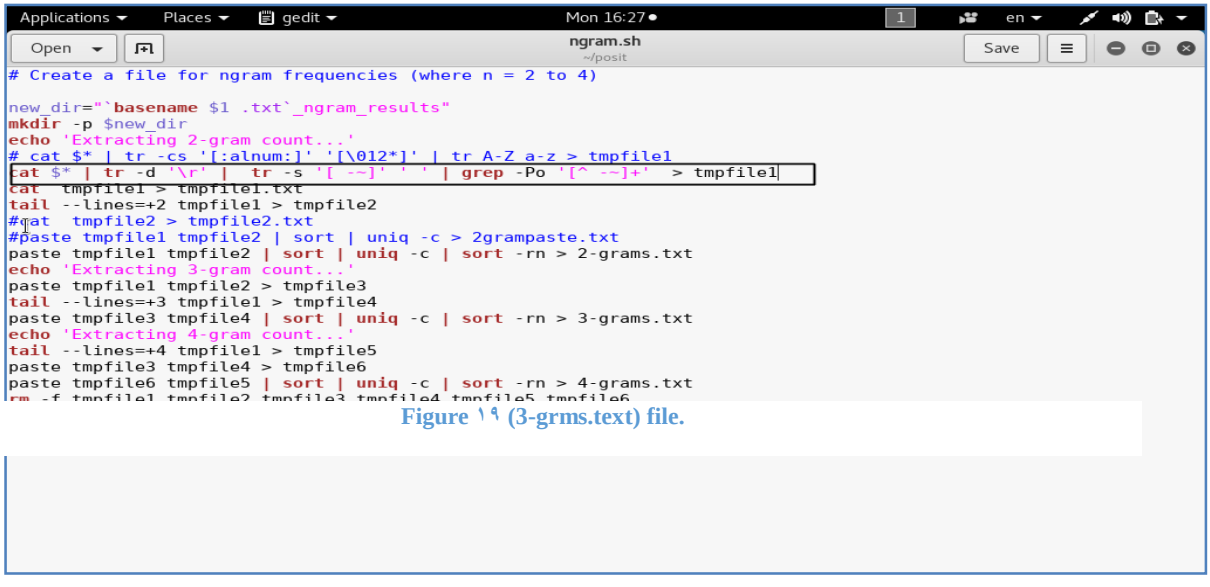


Figure ١٩ (3-grms.text) file.

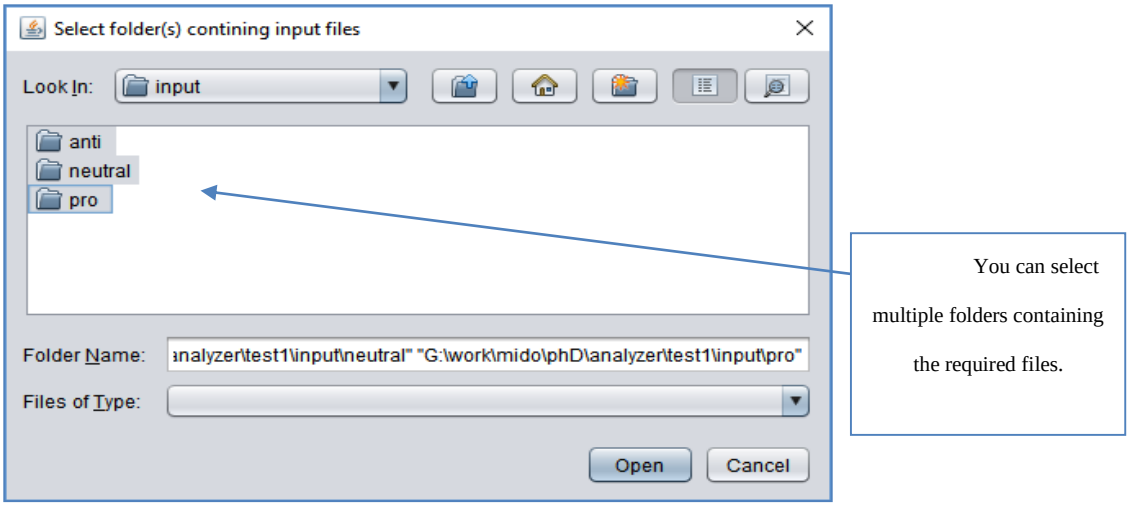


Figure 20 pop-up file dialogs from selected input files.

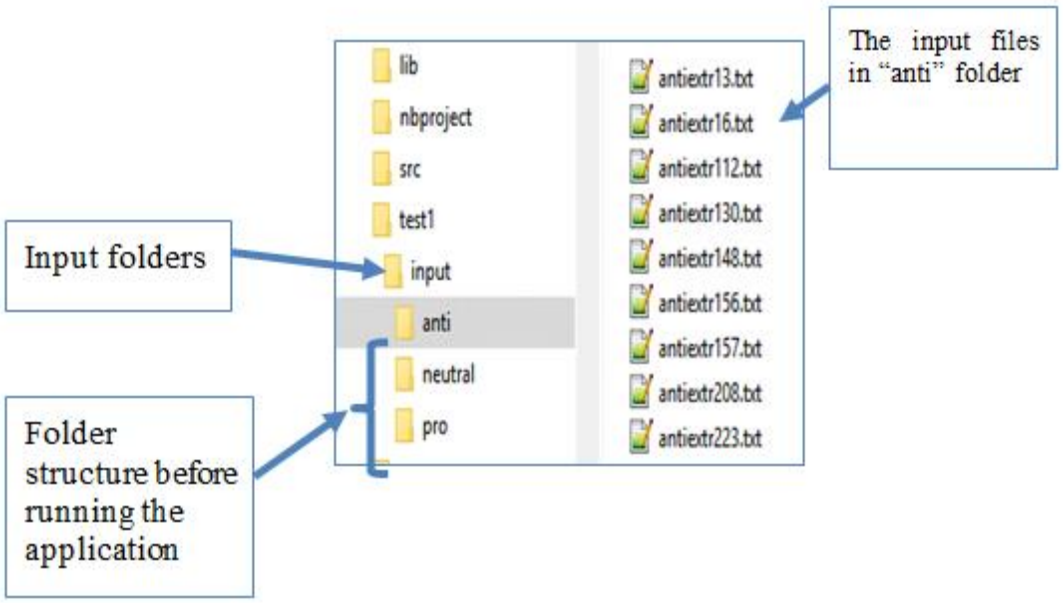


Figure 21 the input folders and the input files in "anti" folder.

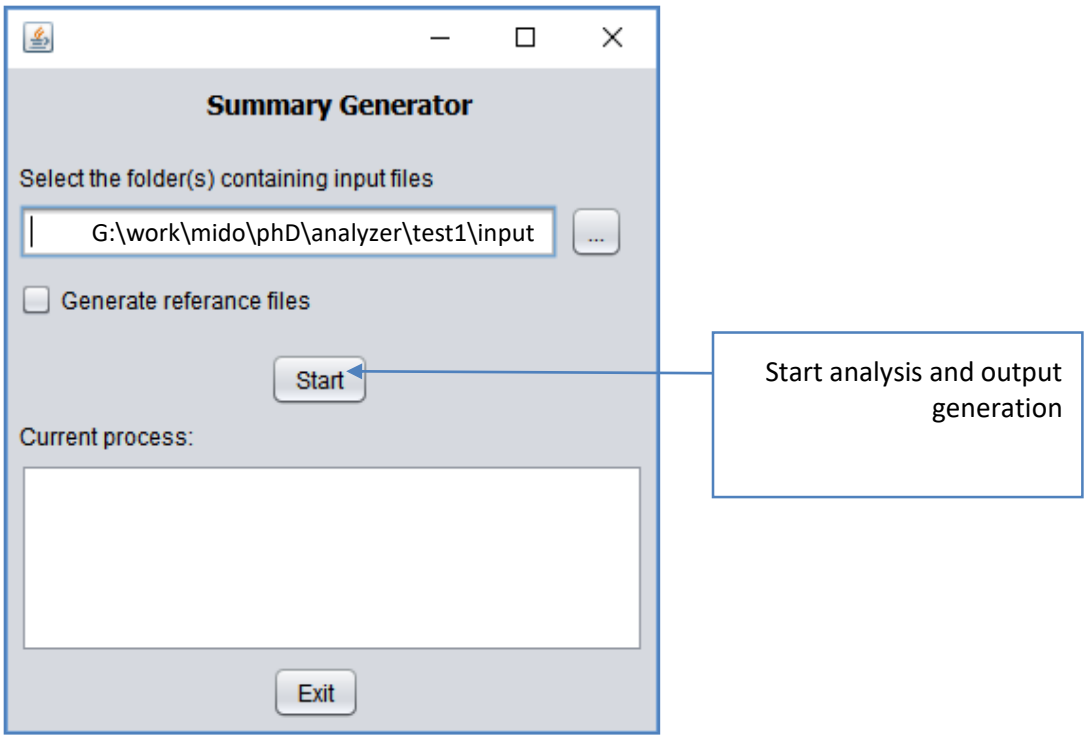
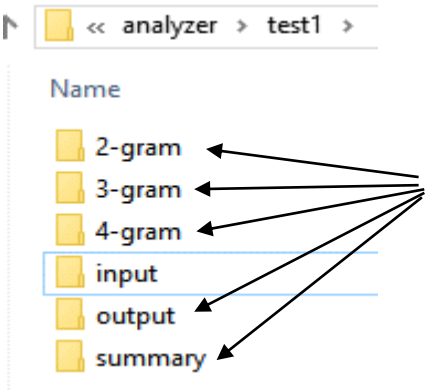


Figure 22 Start analysis and output generation.



Output Folders. Creation function name: create Required Folders ().

Figure 23 Create required folders

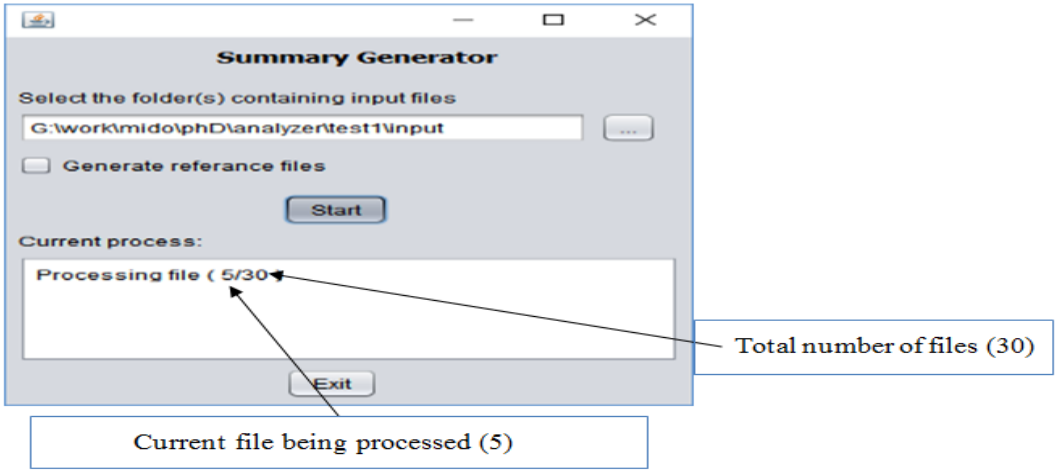


Figure 24 Number of files being processed and total number of files.

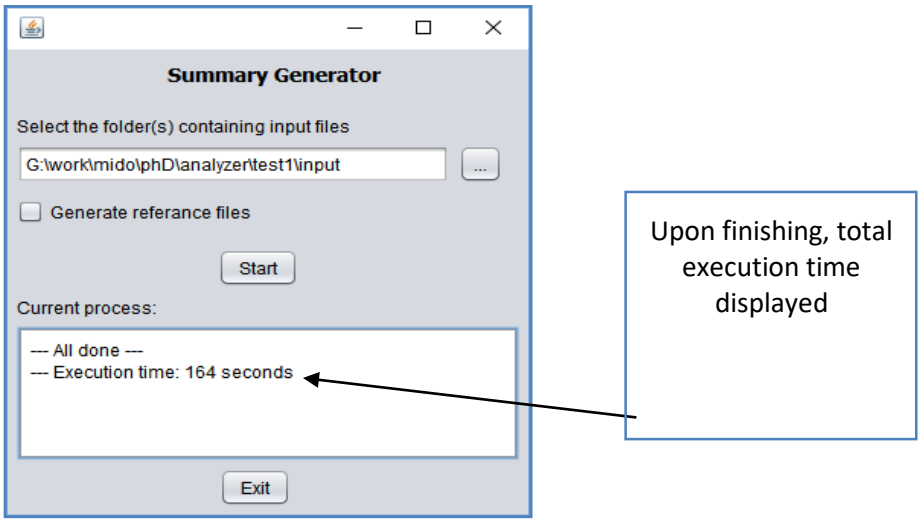


Figure 25 Total execution time.

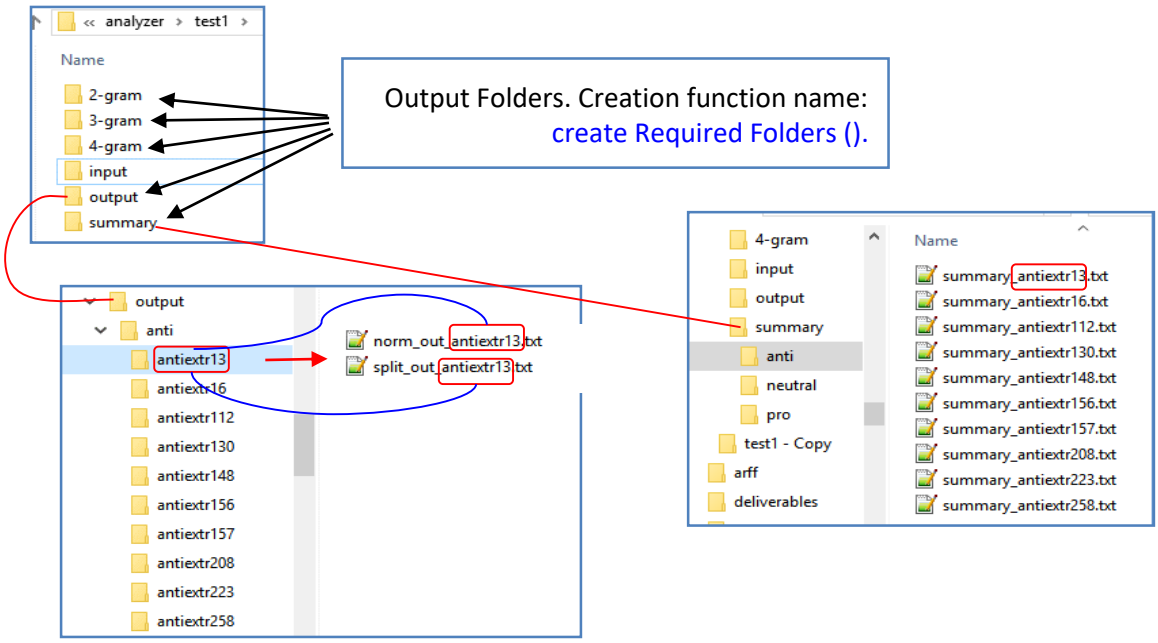


Figure 26 The structure of each output folder.

```
File fileNormalized = new File(FILES_PATH + "output" + folder + inputFileName + "\\norm_out_" + inputFileName + ".txt");
```

So, for an input file named: antiextr13.txt

The normalized file name will be :norm_out_antiextr13.txt

And the split-out file name will be: split_out_antiextr13.txt

Figure 27Generating the name of the normalized file

```
public static void generateOutputs(String inputFileName, String folder) throws IOException {
    inputFileName = inputFileName.substring(0, inputFileName.lastIndexOf('.'));
    new File(FILES_PATH + "output" + folder + inputFileName).mkdir();
    File inputFile = new File(FILES_PATH + "input" + folder + inputFileName + ".txt");
    File outputFile1 = new File(FILES_PATH + "output" + folder + inputFileName + "\\split_out_" + inputFileName + ".txt");
    File fileNormalized = new File(FILES_PATH + "output" + folder + inputFileName + "\\norm_out_" + inputFileName + ".txt");
    SAFARSentenceSplitter splitter = new SAFARSentenceSplitter();
    splitter.split(inputFile, ". . ! , ; ? ~ ˆ ؟", "... ;--; x - ؟! و أ", outputFile1);
    SAFARNormalizer normalizer = new SAFARNormalizer();
    normalizer.normalize(outputFile1, Encoding.UTF_8, fileNormalized, "abcd", "... ;--; x - ؟! . . ! , ; ? ~ ˆ ' ؟");
}
```

Figure 28 Analysis and result generation from the normalized file

Table 1: the summary output result of SAFAR

antiextr13.txt: Input filename:					
1	1823	Number of characters	14	684	Marfo3 Count
2	283	Total words (tokens)	15	601	Mansob Count
3	215	Total unique words (types)	16	732	Majror Count
4	13	Number of sentences	17	1688	Singular Count
5	0.759717	Type/Token Ratio (TTR)	18	93	Double Count
6	21.7692	Average sentence length (ASL)	19	281	Plural Count
7	6.4417	Average word length (AWL)	20	1195	Masculine Count
8	105	Verb POS Type Count	21	867	Feminine Count
9	51	Noun POS Type Count	22	360	Mojarrad Count
10	659	Nakira Count	23	349	Mazeed Count
11	483	Ma3rifa Count	24	145	Lazem Count
12	875	Modaf Count	26	136	Mota3adi Count
13	428	Mota3adi_w_Lazem Count			
NUMBER OF TOKEN TYPES:					
27	6	verb types	29	22	noun types
NUMBER OF POS TYPES:					
30	709	Verbs			
31	20	Particles			

Table 2: Posit Text Profiling Toolset:

1) Total words (tokens)	2) Particle types
3) Total unique words (types)	4) Determiner types
5) Type/Token Ratio (TTR)	6) Interjection types
7) Number of sentences	8) Nouns
9) Average sentence length (ASL)	10) Verbs
11) Number of characters	12) Prepositions
13) Average word length (AWL)	14) Adjectives
15) Verb types	16) Determiners
17) Noun types	18) Particles
19) Adverb types	20) Possessive pronouns'
21) Adjective types	22) Personal pronouns'
23) Preposition types	24) Adverbs
25) Possessive_pronoun_types	26) Interjections
27) Personal_pronoun_types	

Table 3: SAFAR: Software Architecture For Arabic language processing

1) Number of characters	2) Double Count
3) Total words (tokens)	4) Plural Count
5) Total unique words (types)	6) Masculine Count
7) Number of sentences	8) Feminine Count
9) Type/Token Ratio (TTR)	10) Mojarrad Count
11) Average sentence length (ASL)	12) Mazeed Count
13) Average word length (AWL)	14) Lazem Count
15) Verb POS Type Count	16) Mota3adi Count
17) Noun POS Type Count	18) Mota3adi_w_Lazem Count
19) Nakira Count	20) Verb types
21) Ma3rifa Count	22) Noun_types
23) Modaf Count	24) particle_types
25) Marfo3 Count	26) Verbs
27) Mansob Count	28) Nouns
29) Majror Count	30) particle
31) Singular Count	

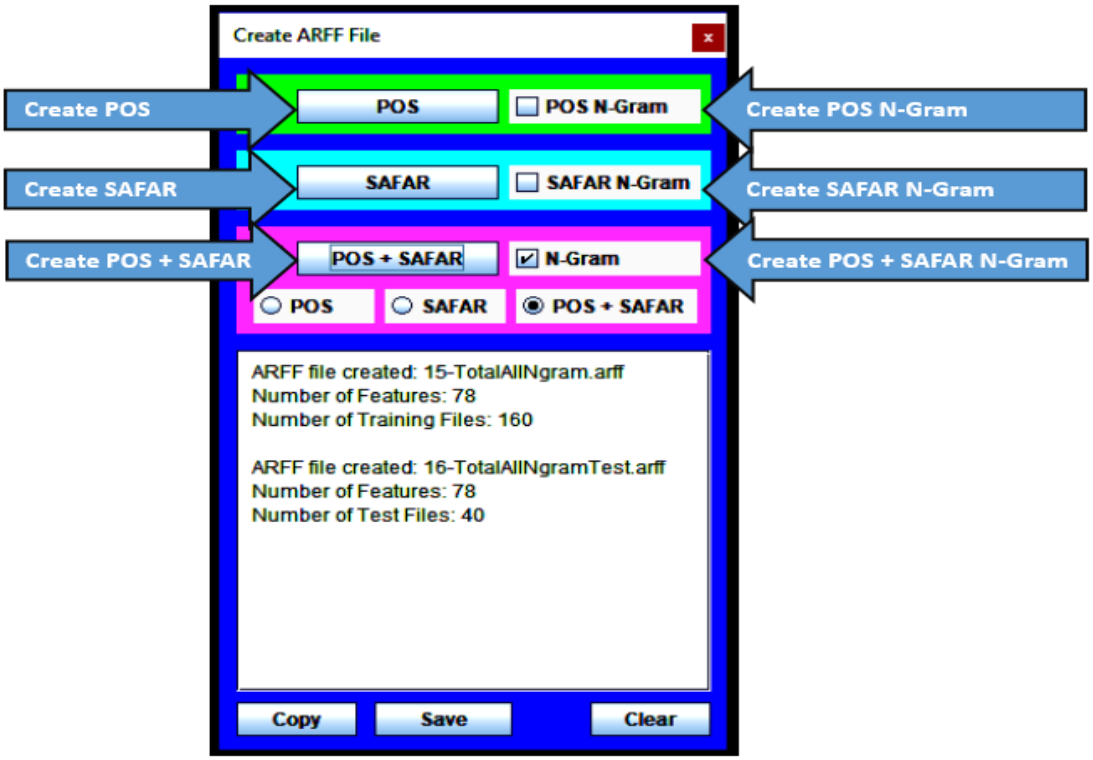


Figure 29 GUI-B: ARFF files generator.

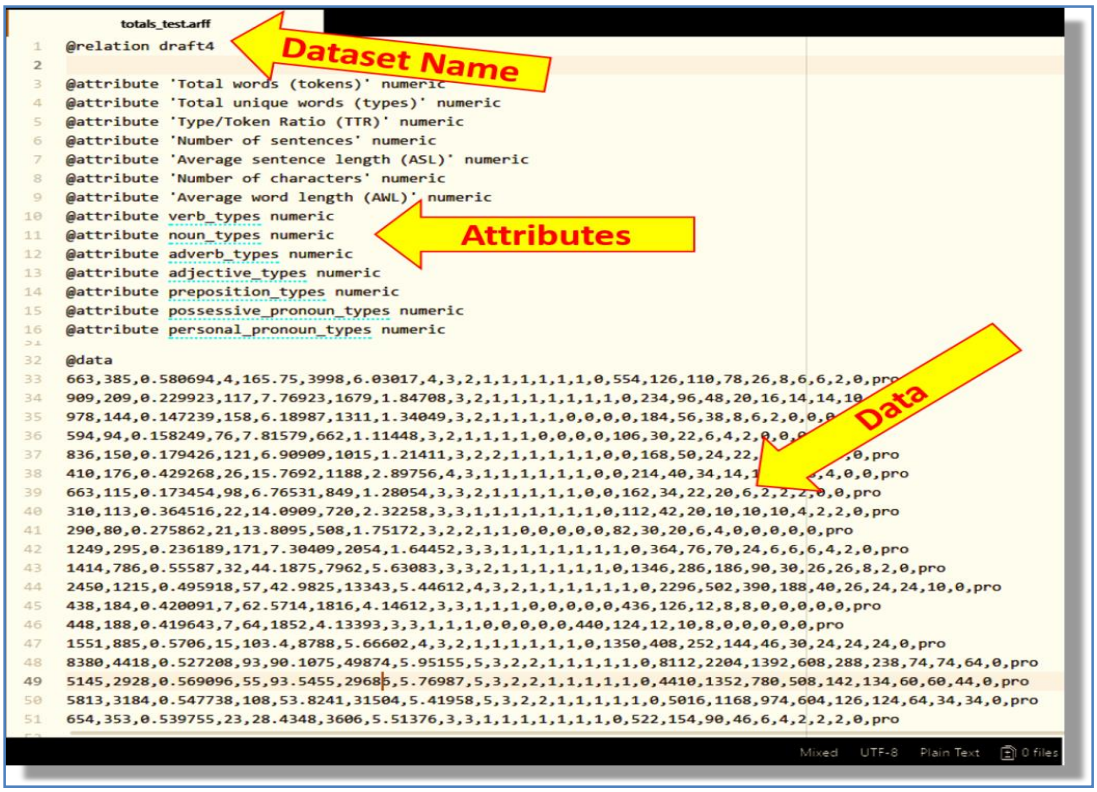


Figure 30 ARFF file format.

Table 4: different AREF files

	ARFF Files	Purpose
POS	pos_70.arff / pos_30.arff	70/30 training/test
	pos_100.arff	Create Model
	pos_test_class.arff	Supervised Test
POS + N-Gram	posGram_70.arff / posGram_30.arff	70/30 training naïve Bayes /test
	posGram_100.arff	Create Model
	posGram_test_class.arff	Supervised Test
SAFAR Toolset	safar_70.arff / safar_30.arff	70/30 training/test
	safar_100.arff	Create Model
	posGram_test_class.arff	Supervised Test
SAFAR+N-Gram	safarGram_70.arff / safarGram_30.arff	70/30 training/test
	safarGram_100.arff	Create Model
	posGram_test_class.arff	Supervised Test
POS + SAFAR	total_70 / total_30	70/30 training/test
	total_100	Create Model
	posGram_test_class.arff	Supervised Test
POS + SAFAR + POS N-Gram	total_pGram_70 / total_pGram_30	70/30 training/test
	total_pGram_100	Create Model
	posGram_test_class.arff	Supervised Test
POS + SAFAR + SAFAR N-Gram	total_sGram_70 / total_sGram_30	70/30 training/test
	total_sGram_100	Create Model
	posGram_test_class.arff	Supervised Test
POS + SAFAR + Total N-Gram	total_psGram_70 / total_psGram_30	70/30 training/test
	total_psGram_100	Create Model
	posGram_test_class.arff	Supervised Test

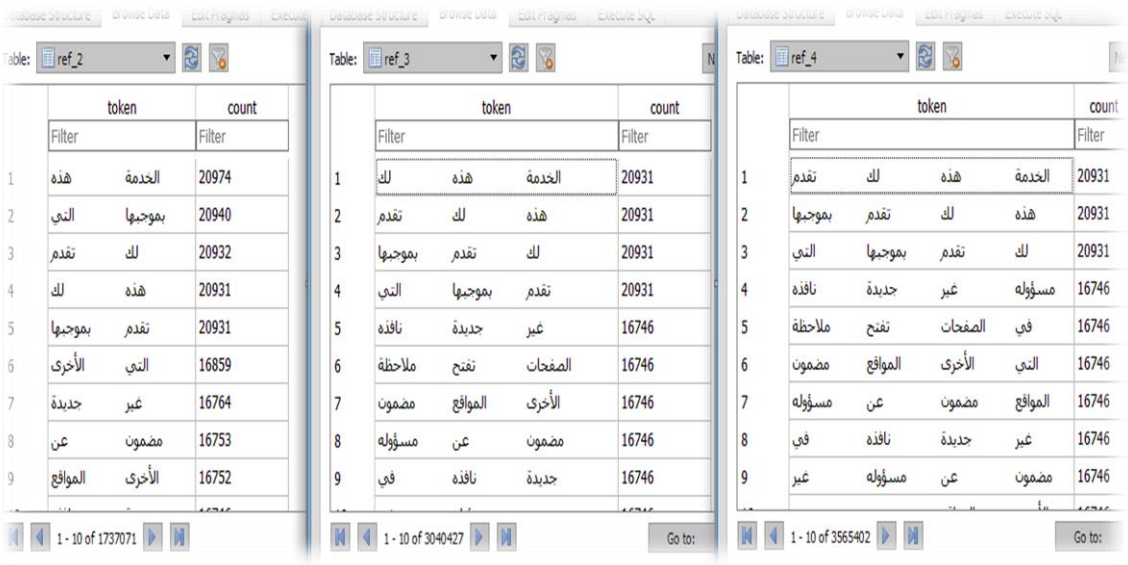


Figure 31 gram Reference SQLite tables.

Table 5: HDD and SSD comparison (https://www.cs.waikato.ac.nz/ml/index.html)

	HDD	**SSD**
Speed	HDD has higher latency, longer read/write times, and supports fewer IOPS (input output operations per second) compared to SSD.	SSD has lower latency, faster read/writes, and supports more IOPS (input output operations per second) compared to HDD.
File Copy / Write Speed	Generally, above 200 MB/s and up to 550 MB/s for cutting edge drives	The range can be anywhere From 50 – 120MB/s

	A	B	C	D	E	F	G	H
1	2-Gram			3-Gram			4-Gram	
2	0:01:23	0:01:07		0:03:37	0:02:36		0:04:05	0:03:30
3	0:00:25	0:00:25		0:01:06	0:00:52		0:01:09	0:01:11
4	0:00:32	0:00:33		0:01:28	0:01:03		0:01:50	0:01:25
5	0:00:29	0:00:25		0:01:02	0:00:44		0:01:14	0:01:00
6	0:00:22	0:00:17		0:00:50	0:00:40		0:01:09	0:00:53
7	0:00:31	0:00:22		0:00:55	0:00:45		0:01:17	0:01:01
8	0:00:28	0:00:23		0:00:56	0:00:46		0:01:18	0:00:56
9	0:00:42	0:00:44		0:01:44	0:01:28		0:02:18	0:02:02
10	0:00:42	0:00:46		0:01:25	0:01:13		0:02:02	0:01:36
11	0:00:33	0:00:31		0:01:16	0:01:13		0:01:38	0:01:25
12	0:00:28	0:00:23		0:01:07	0:00:52		0:01:18	0:01:12
13	0:00:05	0:00:05		0:00:14	0:00:09		0:00:15	0:00:13
14	0:00:24	0:00:22		0:00:59	0:00:54		0:01:29	0:01:05
15	0:00:35	0:00:36		0:01:37	0:01:21		0:02:10	0:01:46
16	0:00:10	0:00:11		0:00:25	0:00:19		0:00:35	0:00:26
17	0:01:02	0:00:50		0:02:05	0:01:53		0:03:42	0:02:45
18	0:00:25	0:00:20		0:00:56	0:00:50		0:01:37	0:00:59
19	0:00:17	0:00:16		0:00:38	0:00:35		0:01:27	0:00:49
20	0:00:31	0:00:31		0:01:27	0:01:05		0:02:10	0:01:30
21	0:10:04	0:09:07		0:23:47	0:19:18		0:32:43	0:25:44

Figure 32 Sample of time consumed by HHD vs RAMDISK based calc

	2-Gram			3-Gram			4-Gram		
	HDD	RAM Drive	Text Index ms	HDD	RAM Drive	Text Index ms	HDD	RAM Drive	Text Index ms
	0:01:23	0:01:07	**656**	0:03:37	0:02:36	**707**	0:04:05	0:03:30	**720**
	0:00:25	0:00:25	**184**	0:01:06	0:00:52	**185**	0:01:09	0:01:11	**198**
	0:00:32	0:00:33	**218**	0:01:28	0:01:03	**231**	0:01:50	0:01:25	**252**
	0:00:29	0:00:25	**155**	0:01:02	0:00:44	**160**	0:01:14	0:01:00	**183**
	0:00:22	0:00:17	**135**	0:00:50	0:00:40	**142**	0:01:09	0:00:53	**167**
	0:00:31	0:00:22	**157**	0:00:55	0:00:45	**160**	0:01:17	0:01:01	**187**
	0:00:28	0:00:23	**144**	0:00:56	0:00:46	**150**	0:01:18	0:00:56	**173**
	0:00:42	0:00:44	**271**	0:01:44	0:01:28	**301**	0:02:18	0:02:02	**348**
	0:00:42	0:00:46	**239**	0:01:25	0:01:13	**253**	0:02:02	0:01:36	**293**
	0:00:33	0:00:31	**195**	0:01:16	0:01:13	**211**	0:01:38	0:01:25	**249**
	0:00:28	0:00:23	**162**	0:01:07	0:00:52	**182**	0:01:18	0:01:12	**214**
	0:00:05	0:00:05	**34**	0:00:14	0:00:09	**36**	0:00:15	0:00:13	**43**
	0:00:24	0:00:22	**161**	0:00:59	0:00:54	**171**	0:01:29	0:01:05	**203**
	0:00:35	0:00:36	**243**	0:01:37	0:01:21	**270**	0:02:10	0:01:46	**322**
	0:00:10	0:00:11	**68**	0:00:25	0:00:19	**71**	0:00:35	0:00:26	**83**
	0:01:02	0:00:50	**345**	0:02:05	0:01:53	**379**	0:03:42	0:02:45	**443**
	0:00:25	0:00:20	**153**	0:00:56	0:00:50	**162**	0:01:37	0:00:59	**191**
	0:00:17	0:00:16	**124**	0:00:38	0:00:35	**130**	0:01:27	0:00:49	**153**
	0:00:31	0:00:31	**212**	0:01:27	0:01:05	**227**	0:02:10	0:01:30	**264**
SUM	0:10:04	0:09:07	**3856**	0:23:47	0:19:18	**4128**	0:32:43	0:25:44	**4686**
SUM ms	604,000	547,000	3,856	1,427,000	1,158,000	4,128	1,963,000	1,544,000	4,686

Figure 33 Sum of proc. duration for data sample and enhance. percentage.