The Islamic University–Gaza
Research and Postgraduate Affairs
Faculty of Information Technology
Master of Information Technology

الجامعــــــة الإســــلامية – غـزة
شئون البحث العلمي والدراسات العليا
كـليـــة تكنولوجيا المعلومات
ماجستير تكنولوجيا المعلومات

Automatic Arabic Text Summarization for Large Scale Multiple Documents Using Genetic Algorithm and MapReduce

التلخيص التلقائي للنصوص العربية المتعددة كبيرة الحجم باستخدام الخوارزمية الجينية و MapReduce

Sulaiman Nasrallah Al Breem

Supervised by

Dr. Rebhi S. Baraka

Associate professor of Computer Science

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Information Technology

October/2016

إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Automatic Arabic Text Summarization for Large Scale Multiple Documents Using Genetic Algorithm and MapReduce

التلخيص التلقائي للنصوص العربية المتعددة كبيرة الحجم باستخدام الخوارزمية الجينية و MapReduce

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

Declaration

I understand the nature of plagiarism, and I am aware of the University's policy on this.

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

Student's name:	سليمان نصر الله البريم	اسم الطالب:
Signature:		التوقيع:
Date:	25/10/2016	التاريخ:

بِسْـــــــمِ اللهِ الرَّحْمَنِ الرَّحِيمِ

الجـامعـة الإسلامية-غـزة
The Islamic University of Gaza

مكتب نائب الرئيس للبحث العلمي والدراسات العليا   هاتف داخلي: 1150

ج س غ/35/
الرقم: 2016/10/25 Ref:
التاريخ: Date:

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ سليمان نصرالله سليمان البريم لنيل درجة الماجستير في كلية تكنولوجيا المعلومات برنامج تكنولوجيا المعلومات وموضوعها:

التلخيص تلقائي للنصوص العربية المتعددة كبيرة الحجم باستخدام الخوارزمية الجينية و MapReduce

Automatic Arabic Text Summarization for Large Scale Multiple Documents Using Genetic Algorithm and Map Reduce

وبعد المناقشة التي تمت اليوم الثلاثاء 24 محرم 1438هـ، الموافق 2016/10/25م الساعة الحادية عشر صباحاً ، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة	مشرفاً و رئيساً	....
د. أشرف يونس مغاري	مناقشاً داخلياً	....
د. أحمد يحيى محمود	مناقشاً خارجياً	....

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.

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

والله والتوفيق ،،،

نائب الرئيس لشئون البحث العلمي والدراسات العليا

أ.د. عبدالرؤوف علي المناعمة

Abstract

Automatic Text summarization is one of the most important problems in the area of text mining and information retrieval. The importance of automatic text summarization comes from its ability to provide the most significant information from a large text by reducing the size of textual documents. Multi document summarization focus in extracting the most significant information from a collection of textual documents. Most summarization techniques require the data to be centralized, which may not be feasible in many cases due to computational and storage limitations. The huge increasing of data emerging by the progress of technology and the various sources of makes automatic text summarization of large scale of data a challenging task.

We propose an approach for automatic text summarization of large scale Arabic multiple documents using Genetic algorithm based on open source MapReduce model, MapReduce is a powerful parallel programming model. We make our approach insuring scalability, speed and accuracy in summary generation and try to eliminating redundancy for sentences and increasing the readability and cohesion factors between the sentences of summaries. We evaluate the proposed method using several automatic summarization quality measures in terms of Recall, Precision, F-measure. In addition to that we evaluate the parallel computation environment in terms of speed up, efficiency and scalability.

The experiments resulted in high precision and recall scores. This indicates that the system successfully identifies the most important sentences. In addition to that, the proposed approach provides up to 10x speedup score, which is faster than executing the same code on single machine. Therefore, it can deal with large-scale datasets successfully. Finally, the efficiency score of the proposed approach indicates that the largest data set utilize the available resources up 62% which is a satisfying result taking into account the available data set sizes.

Keywords: *Text Summarization, Parallel Genetic Algorithm, MapReduce, Hadoop*

الملخص

تعتبر عملية التلخيص التلقائي للنصوص من أهم المهام المتعلقة بمجال التنقيب عن البيانات واسترجاع المعلومات. تكمن أهمية التلخيص التلقائي للنصوص بأنها قادرة على استخراج أكثر المعلومات أهمية من مجموعة كبيرة من المستندات النصية. أما مجال تلخيص المستندات المتعددة فهو يركز على استخراج أهم المعلومات الموجودة في مجموعة من المستندات النصية المتعددة. أغلب تقنيات تلخيص النصوص تتطلب أن تكون البيانات المطلوب تلخيصها موجودة في مكان مركزي واحد، لكن في عدة حالات من الصعب تحقيق هذا الشرط بسبب محدودية التخزين والمعالجة. وهذا بسبب التطور الكبير في مجال تكنولوجيا المعلومات والبيانات الكبيرة التي نتجت من خلال أنشطة البشر. لذلك أصبحت عملية التلخيص التلقائي للنصوص المتعددة كبيرة الحجم عملية تحدي وذلك بسبب الزيادة المطردة والمتواصلة لأنشطة البشر والمصادر المختلفة للبيانات.

قمنا باقتراح طريقة لعمل التلخيص التلقائي للنصوص العربية المتعددة وكبيرة الحجم باستخدام الخوارزمية الجينية وMapReduce وهو عبارة عن نموذج للبرمجة المتوازية. هذه الطريقة تحقق الدقة في استخراج الجمل المهمة من النصوص والسرعة في عملية التلخيص وقابلة للتوسع في حال تم إضافة المزيد من البيانات النصية والموارد اللازمة للعمليات الحسابية. تمت عملية تقييم الطريقة المقترحة باستخدام عدة قياسات مشهورة مثل الدقة (Precision)، التذكر(Recall) ، سرعة عملية التلخيص (Speedup)، كفاءة استخدام الموارد المستخدمة في المعالجة (Efficiency)، قابلية النظام للتعامل مع الزيادة في البيانات النصية والموارد (Scalability).

لقد أظهرت النتائج بعد تطبيق الطريقة وجود مؤشر جيد بالنسبة للدقة والتذكر، وهذا يشير الى أن النظام قام باستخراج الجمل المهمة من هذا العدد الكبير من النصوص. بالإضافة لذلك أظهرت النتائج المتعلقة بسرعة عملية التلخيص أن النظام المقترح كان أسرع 10 مرات عند استخدام مجموعة من الأجهزة عنه عند استخدام جهاز واحد. أخيرا تبين بعد عملية التلخيص أن النظام المقترح قام باستغلال الموارد المتاحة بكفاءة تصل الى 62% وهي نتيجة جيدة الى حدا ما عند الأخذ بعين الاعتبار حجم البيانات المتوفرة.

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

وَقُل رَّبِّ زِدْنِي عِلْمًا

Dedication

To My Great Mother, Father, the reason of what I become today,

To my dear wife,

To my beloved Sister,

To My beloved brothers,

To My Best Friends,

This work is dedicated

Acknowledgment

Firstly, I am grateful to Allah for the good health that were necessary to complete this work.

I would like to thank my thesis supervisor, ***Dr. Rebhi Baraka*** *Dean of the Faculty of Information Technology at the Islamic University of Gaza, who allowed this thesis to be my own work and steered me in the right direction.*

I would also like to acknowledge the academic staff of Faculty of Information Technology at The Islamic University of Gaza for sharing their expertise and knowledge during my study.

Finally, I must express my very profound gratitude to my parents, my wife, my sister, my brothers and my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Sulaiman N. Al Breem

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

ANLP	Arabic Natural Language Processing
API	Application Programming Interface
CA	Classical Arabic
HDFS	Hadoop Distributed File System
HTML	Hyper Text Mark-up Language
IDF	Inverse Document Frequency
MCA	Modern Classical Arabic
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part Of Speech Tagger
ROUGE	Recall-oriented understudy for gisting evaluation
TF	Term Frequency
T_S	Execution Time of Single processor
T_P	Execution Time of Parallel processors

Chapter 1

Introduction

1.1 Background and Context

With the huge increase of digital data due to the intensive human activities, the area of text mining and information retrieval become more interesting. The importance of automatic text summarization stems from its ability to provide the significant information from a large text by reducing the size of textual documents. It can be used in many real life applications in many sectors especially in minimizing reading time of articles and providing the most important information about some events. Luhn (Luhn, 1958) was the first to talk about automatic summarization based on selection of significant sentences from a document using term frequency measures.

In general, automatic text summarization process can be divided by many factors:

- By its nature: Extraction or Abstraction.
- By document count: Single Document or Multi-document.
- By language: Single language or Multi-lingual.

Extraction based summarization approach is commonly used in most text summarization techniques (Aliguliyev, 2009) which extract some parts from a document to produce a summary without any modification. While abstraction methods use complex language techniques to analyze and reform the document content using different words to produce a summary. Single language summarization is used to generate a summary from a single document while multilingual summarization makes summary from multiple documents from multiple languages (Giannakopoulos, 2013).

In single document summarization, the information from single document is gathered using various ways to get the most important information from the sentences of the document (Luhn, 1958), (El-Shishtawy & El-Ghannam, 2012). Multi document summarization focuses on extracting the most significant information from a collection of textual documents. It deals with many issues related to multi documents; like the presence of redundant sentences and decreasing the readability and text cohesion factors on the produced summary.

It deals with analyzing and understanding a collection of documents for searching the silence and important hidden information to compact the text to generate a minimized summary to help the user to understand the important parts of information in a large document collection without the need to read whole documents. Multi document summarization can be used in many fields like news articles, blogs, web pages, tweets, books, reports and archives. One of the most early works in the area of multi document summarization is carried out by NLP group at Columbia university (McKeown & Radev, 1995). They released a system called SUMMONS.

Recently there have been many models for automatic summarization of multiple documents like sentence extraction approaches (Goldstein, Mittal, Carbonell, & Kantrowitz, 2000), feature extraction approaches (Aristoteles, Ridha, & Julio, 2012), clustering approaches (Waheeb & Husni, 2014), graph based approaches (Erkan & Radev, 2004), Genetic algorithm approaches (Nandhini & Balasundaram, 2013). One problem with these approaches is that the required time to summarize multiple documents is raised incredibly, this makes most the current systems unable to deal with the increase of data sizes.

Genetic algorithm is a popular method used in summarization (Golberg, 1989), (Holland, 1975). In addition to that, Genetic algorithm is based on the principles of evolution, heuristic search and natural selection to find optimized and novel solution to hard problems. Genetic algorithm selects the best part of the problem to provide the best solution using simple logic and basic operations such as initialization, selection, crossover, mutation that discussed on details in Section 2.3. Although Genetic algorithm is suitable for searching problems (Whitley, 1994) , it suffers from the high computation and memory resources that are needed if the size of the individuals of the problem solution is increased.

As mentioned above, the huge increase of data emerging from the progress of technology and the various sources makes automatic text summarization of very large scale of data a challenging task. In order to deal with this issue, many technologies emerge in the area of parallel processing. MapReduce (Dean & Ghemawat, 2008) is one of the state of the art models for processing large scale data. The model deals with parallel and distributed systems, it provides mechanisms for parallelization, fault-tolerance, locality optimization, and load balancing. MapReduce uses divide-and-conquer principle using two main methods, the first method is Map and is used to convert the data into key/value pairs as intermediate values, the second method is Reduce and is used to combine the generated intermediate data from Map function to produce the results.

To perform automatic text summarization of multiple Arabic documents for large-scale data, Genetic algorithm can be implemented in parallel fashion (Saha, 2014). There are many models to implement Genetic algorithm over MapReduce. One of them is Simple Genetic Algorithm (SGAs) which is similar to running on a local machine. The second is using Island approach (Whitley, Rana, & Heckendorn, 1999), where the data is distributed on the machines of the cluster and evaluation is done locally and individuals can be migrated between islands in certain time.

One major difference of multi document summarization to single document summarization is the complexity of choosing the best sentences to construct a summary in the presence of redundancy. Many researches use similarity measures to reduce the redundancy in the summary (Momtaz & Amreen, 2012).

In this research, we build an approach that combine Genetic algorithm with MapReduce model to perform automatic text summarization for Arabic multiple documents. MapReduce is used to speed up the summarization process and provide the required scalability while maintaining quality of the summarization process.

1.2 Statement of the Problem

Automatic text summarization for large-scale Arabic text collected from multiple sources and partitioned to several locations represents a serious challenge and therefore requires a new approach to perform the summarization process in a distributed fashion that preserve the quality of the summarization while maintaining computation speedup and scalability.

The problem is how to use Genetic algorithm to perform Automatic Arabic text summarization for large-scale multiple documents over MapReduce parallel model that is accurate in terms of resulting summaries, time efficient and computation scalable.

1.3 Objectives

1.3.1 Main objectives

To design an approach based on Genetic algorithm for automatic text summarization of text with multiple documents and multiple domains while insuring scalability, speedup and maintaining accuracy. Arabic large-scale data sets. Genetic algorithm evaluates and selects the best sentences that make a summary. MapReduce model automatically processes large data in parallel and distributed fashion. The big data volumes in applications make automatic text summarization a challenging task.

1.3.2 Specific Objectives

- Gathering and processing Arabic text corpus collected from many sources from the Internet like news, blogs, tweets, books and convert them into readable text format for analysis. The corpus size should be from 2 to 4 gigabytes.
- Extracting important common features from data like sentence length, sentence position and many important features. This will affect the mechanism of sentence selection for Genetic algorithm.
- Design and implement Genetic algorithm over MapReduce model to speed up the evaluation and selection of sentences to produce a readable and cohesive summary with minimum redundancy.
- Perform a set of summarization experiments on Genetic algorithm MapReduce implementation and collect results to be used in the evaluation process.
- Evaluate the approach based on the collected results for summary quality, speedup, efficiency and scalability.

1.4 Significance

With the increasing size, dimensionality and distribution of data, the approach improves automatic text summarization performance in terms of speedup, scalability and quality. The use of Genetic algorithm improves the summarization readability and cohesion. Using parallelization techniques such as MapReduce facilities and speed the process of summarization and handle large data sets efficiently. The research using

Genetic algorithm and MapReduce to enhance Arabic language text summarization and facilitates the research in this important area.

1.5 Scope and Limitations

1- The research is concerned with performing automatic text summarization for multiple Arabic text documents by implementing Genetic algorithm over MapReduce model.

2- The summarization is based on a Genetic algorithm that uses a fitness function that finds the group of sentences having the maximum evaluation in terms of readability and cohesion with minimum redundancy.

3- The source data are stored in separate files in text formats.

4- The text documents used in the experiments are assumed to be already categorized to the chosen domains.

5- Extraction summarization type be the only summarization to be used to make summarization the other kind such as abstraction is excluded.

6- The summarization is concerned with Arabic language only.

7- No post processing is needed for the produced summaries.

8- Fixed summary length is used rather than compression ratio from the original corpus.

9- Because the large size of data, three specialists will be manually evaluating a random sample from the produced summaries for measuring the summaries quality.

10- We evaluate the quality of the produced summarization through precision, recall and f-measure, which are well known and agreed upon measures.

11- We use 1, 2, 4, 8, 12 and 16 nodes to measure the effects on the speedup, efficiency and the scalability of proposed approach.

1.6 Methodology

To achieve the research objectives and carry out automatic text summarization for large scale Arabic text using Genetic algorithm and MapReduce, the research methodology consists of the following phases:

Phase 1: Data Gathering

To make automatic text summarization for large scale data, we need to collect a large data corpus as the main input of the system. There are many sources for Arabic documents like online news, online books and many other online sources. For this purpose, we will collect data from online newspaper websites, the collected data should be stored on the storage of MapReduce framework, Hadoop Distributed File System (HDFS) in text format.

Phase 2: Data Pre-processing

Data pre-processing is an important step to make automatic text summarization. It involves preparing the text for future processing. We will carry out many processing steps for the input text such as:

1- Text cleaning
2- Sentence tokenization
3- Arabic stop word removing
4- Arabic root stemming
5- Arabic word normalization
6- Remove diacritics
7- Part of speed tagging (POS)

Phase 3: Feature Extraction

The multi document summarizer is based on Genetic algorithm and requires ranking every possible solution based on the fitness function. The features are collected from the text directly and are considered as the basic measurements of the fitness function of every summary, the used features are likely to consist of the following features:

1- Sentence position
2- Sentence length
3- Noun occurrences
4- Verb occurrences
5- Readability measures
6- Cohesion measures
7- Term frequency

Phase 4: Designing the Genetic Algorithm as MapReduce

The proposed approach will be composed of several steps starting with dividing the data into separate individuals and evaluating them to select the best individuals for evolving using the main Genetic algorithm operators' crossover. This will insure the iterative nature of Genetic algorithm over MapReduce as described in the following steps:

1- Create initial population: The Genetic algorithm has an iterative nature and we will make every iteration as MapReduce job, at the first iteration we should create the initial population which contains the whole data divided into chromosomes or individuals. Every individual has a fixes length of random sentences and considered as a possible summary. The fitness of the created individuals is calculated using the aggregation of the features selected from all the sentences of each individual.

2- Iterative Genetic algorithm as MapReduce: Genetic algorithm will be implemented over Hadoop MapReduce framework using simple Genetic algorithm which simulates the sequential Genetic algorithm. Every Genetic algorithm iteration is encapsulated in a single MapReduce job. We should run from 3 to 5 iterations and if there is improvement on the fitness of individuals we can start new iterations.

3- Evolving Genetic algorithm: we can evolve the fitness of Genetic algorithm individuals through the selection and crossover operations. At the end of each iteration we will find the best individual and store them for the next iteration and so on, and store back the fail individuals to have another chance.

4- Sorting Individual Sentences: after termination the Genetic algorithm, we get the individual with maximum fitness and sort its sentences using appropriate sorting mechanism. We can use the chronological ordering, which sort the sentences based on the sentence date.

Phase 5: Performing Experiments and Evaluating the Results

At the final phase, we will evaluate the approach for quality of generated summaries measures such as precession, recall and F-measure. In addition to that, the performance of the parallel computing environment will be evaluated in terms of speed up, efficiency and scalability.

1.7 Thesis Organization

The thesis is organized as follows: Chapter 2 discusses theoretical and technical foundations with topics related to the research area. Chapter 3 states the works related to the research. Chapter 4 presents the design of the parallel summarization approach using Genetic algorithm. Chapter 5 presents the implementation of the approach and the experiments. Chapter 6 presents the evaluation of the proposed approach. Finally, Chapter 7 presents the conclusion and future work.

Chapter 2

Theoretical and Technical Foundation

To perform the automatic text summarization of large-scale text multiple Arabic documents using Genetic algorithm and MapReduce, we should provide some basic concepts and many techniques.

2.1 Text Summarization

In general, text summarization is the process of providing short information and reduced version of online text resources such as newspapers, books and human activities, which contain important overview of the current events. Summarizing the text can help in understanding a large text containing many important events in a short time without the need to read the entire text.

There are two forms of text summarization based on the summarization nature, abstraction and extraction. Abstraction summarization use computer program for in-depth understand and analyze the text and provide new description which may contains new words which do not exist in the original text. While extraction type of text summarization, analyze the text and extract the main important and readable parts from the original text to construct a summary with the main concepts.

2.1.1 Summarization Methods

There are many methods for automatic text summarization. The first method is proposed by (Luhn, 1958), which depends on the selection of significant sentences from a document using term frequency measures. Another important method is using text-clustering method to group related terms into groups, and rank them based on various features. Many researches use graph methods to make text. The summary is a subgraph of the main graph containing less redundancy and high weighted sentences. Often, text summarization methods depend on analyzing text and extracting some important features from it, which present the main important information. These features can include similarity with title, term scoring, sentence position and sentence length. A popular method uses Genetic algorithm as global search mechanism to improve the selection criteria of the sentences of summary. Genetic algorithm is presented in more details in Section 2.3 and using Genetic algorithm in text summarization is presented in Section 3.1.

2.1.2 Summarization Challenges

The huge increase of online human activates have produced large amount of data. Traditional single and multiple text summarization methods are designed to process small amounts of data, therefore using single machine can be appropriate. Nevertheless, when talking about large-scale data it cannot deal with it efficiently because of the long processing time and large storage needed to make summarization. Arabic language has a complicated structure and hard morphological analysis, therefore making automatic text summarization for Arabic text is a hard task. In Chapter 3 we present various approaches to text summarization which varies depending on extraction methods, the kind of summarization, the amount of data and the applicability of parallelization.

2.2 Arabic Natural Language Processing

Natural Language Processing (NLP) is a part of artificial intelligence that involved in developing techniques, theories and software for analysing, understanding and interpreting the natural languages of human (Chowdhury, 2003).

Arabic language is one of the most spoken languages in the world especially in Arab world and some countries in Africa speak it natively. Therefore, as natural language, Arabic has many important features which make it highly structured and derivational language. Arabic content and online users increase rapidly due to the high availability of online resources that provide Arabic content like Wikipedia, news agencies and knowledge provide. Many studies emphasize that Arabic becomes a popular and online Arabic reached more than 375 million speakers (Stat, 2015).

Although, the Arabic content does not exceed 5.2% of the international content (Stat, 2015), it is increased rapidly and this percentage is due to the improve in the near future. This presence of the Arabic content need suitable approaches for analysis, understanding and interpreting Arabic content using modern NLP techniques.

Intensive research in the area of NLP for English language has been done and there is a need for improving NLP techniques for Arabic Language. The concept

Arabic Natural Language Processing ANLP is concerned with developing theories, tools and techniques for analysing and understanding Arabic language. Many of ANLP component may defer from its English equivalent because Arabic has a complex linguistic structure (Farghaly & Shaalan, 2009a).

In general there are two types of Arabic language (Najeeb, Abdelkader, & Al-Zghoul, 2014) , the first is Classical Arabic (CA) or Qur'anic Arabic. The ancient Arab used classical Arabic widely and it is highly complex, accurate, imaginative and sophisticated. On the other hand, Modern Classical Arabic (MSA) is the global spoken language in all areas of Arabic usage, like online web sites, television programs and newspaper. Arabic language is consisting of 28 letter and written from right to left and unlike English it uses the Verb-Subject-Object style in writing sentences. Arabic letters do not have capitalization and some letters have several forming writing based on its position in the word.

ANLP have various components that can be applied to Arabic text, like, Stemmer, Stop Word Removal, Named Entity Recognition, Speech Recognition, Part of Speech Tagging, Machine Translation, Word Sense Disambiguation, Morphological Analysis, Question Answering, Text Generation and many other components. In our research, we will list some of the techniques we use in Text Summarization.

2.2.1 Stop Word Removing

We can define stop word removal task as removing some words from text before applying NLP. This can be carried out by using manual constructed list. It is a language dependent task because every language has its own word list. By default stop words is commonly used in any language and it is very important, however, it presents a problem in text processing field (Silva & Ribeiro, 2003). We must remove these words because they have no impact to make text important and make text non-informative.

For Arabic many researches propose a list of words to be removed from text, some of them have built a list of 1000 Arabic words manually (Al-Shalabi, Kanaan, Jaam, Hasnah, & Hilat, 2004).

2.2.2 Stemming

Stemming methods are used to remove some letters from a word and return the word into its roots. As described in Table (2.1), it deals with three types of affixes; the first is prefix and it is placed before the stem of the word like un-use. The second is infix, which exists between the stem letters. The third is suffix and it added at the end of the stem like playing.

Table (2.1): Arabic Word Parts Example.

Word	Prefix	Infix	Suffix	stem
يتعامل	ي	ت، ا	و، ن	عمل

There are two main methods for stemming. The first is root stemming which removes the three types of affix: Prefix, Infix and Suffix to return the word into its root form as shown in Table (2.2). The root stemmer may change the structure of a word and can make some issues for text processing because many words with different meanings may belong to the same word stem.

Table (2.2): Root Stemmer Example.

عمل	عملوا يعملون عاملون

The second is light stemming, which is an improved approach of root stemming, therefore, it only removes the prefix and the suffix from the word and the infix remains. Many researches prefer light stemming (Al-Maimani, Naamany, & Bakar, 2011). On the other hand, light stemming may affect the semantic similarity between sentences because similarity is based on typical word matching.

2.2.3 Part of Speech Tagging (POS):

Natural language usually consists of group of words and every word builds the structure of a sentence or paragraph and belongs grammatically to a category of speech. POS is the process of marking every word of a text to a group of language such as noun, verb, and adjective as shown in Table (2.3).

In general, there are two main categories of POS; the first is rule-based and stochastic taggers. One of the early POS was developed by (Brill, 1992) from English and (Zanoli & Pianta, 2009) for Italian language using rule-based method that automatically learns rules. For Arabic, khoja (Khoja, 2001), (Mohamed & Kübler, 2011) proposes a POS for Arabic text using predefined tag sets consisting of three main sets and this enables to derive more than 35 sub tags. POS is a complex task for Arabic language because Arabic have a complex morphological structure. Therefore, before tagging a word, many processing of the text must be done such as normalization and stemming.

Table (2.3): Part Of Speech Example.

POS	DT	NN	VB	NNS	JJ
Description	Determiner	Singular noun	Verb, base form	Plural noun	Adjective
Word	The	worker	fix	vehicles	lonely

2.2.4 Text Readability

There are many important features to make text summarization popular, summary text must be readable easily. Readability means how the text can be easily understandable by more people. There are many formulas for computing the difficulty of English text like Gunning Fog Index FOG (Gunning, 1969), SMOG (Mc Laughlin, 1969), Flesch–Kincaid (Kincaid, Fishburne Jr, Rogers, & Chissom, 1975). Theses readability measures use some properties gathered from text like, average sentence length, average word length and average number of syllables in text.

For Arabic language, few work is done for estimating readability of text. One of the early works for Arabic language readability measures is by (Al-Dawsari, 2004). They use formula with five readability features, which include average word length, average sentence length, word frequency, percentage of nominal clauses and percentage of definite nouns. Another approach is proposed by (Mat Daud, Hassan, & Abdul Aziz, 2013) and uses a corpus of Arabic words, every word is assigned with a rank. For computing the readability using the formula: Readability= (summation of sentence words rank)/number of sentence words.

A state of the art approach, proposed by (El-Haj & Rayson, 2016) calculates the Arabic text readability with and without diacritics. It is considered as a modified version of the famous measure like Flesch and Fog metrics. The method is used to count the short, long and stress syllables in Arabic, which is required for measuring how easy is the Arabic text. By analyzing a large Arabic text corpus, they found the average word length to be five words and the average syllables count is four. This result is used in the equation of readability score of Arabic text. The method assigns higher score to the sentences containing word with small syllables and with simple words to this make it easy and readable sentences. We use readability measures in assign weight to the sentences for improve sentence selection in Section 4.2.

2.2.5 Text Cohesion

Another important measure of text quality is text cohesion. It helps readers to make sense of what they read and what writers want to convey. It is grammatically or lexically linking text sentences. It makes the information in the text organized and connected well and share related information. In general, lexical text cohesion is achieved by many metrics (Todirascu et al., 2013) generated from the text like average similarity, which use similarity measures like cosine similarity to find the similarity between sentences. Another important metric is word overlap, which measures the number of common words in two consecutive sentences. The cosine similarity of two sentences in our proposed approach is described in details in Section 4.2.6.

2.3 Genetic Algorithm

Genetic algorithm is heuristic search method which has an iterative nature and is created based on the principles derived from the Darwinian theory that simulate the evolutionary rules and natural selection (Golberg, 1989). In general, Genetic algorithm uses the fitness function that makes the solution individuals evolve independently based on the survival of the fittest concept. Genetic algorithm uses the principle of divide and conquer for eliminating the search space and deals with every individual separately. This require dividing a big problem into smaller and potential solutions especially for large computational issues. Genetic algorithm can evolve the proposed solutions using some genetic operations and randomly changes some features from the population based on the strength of each solution.

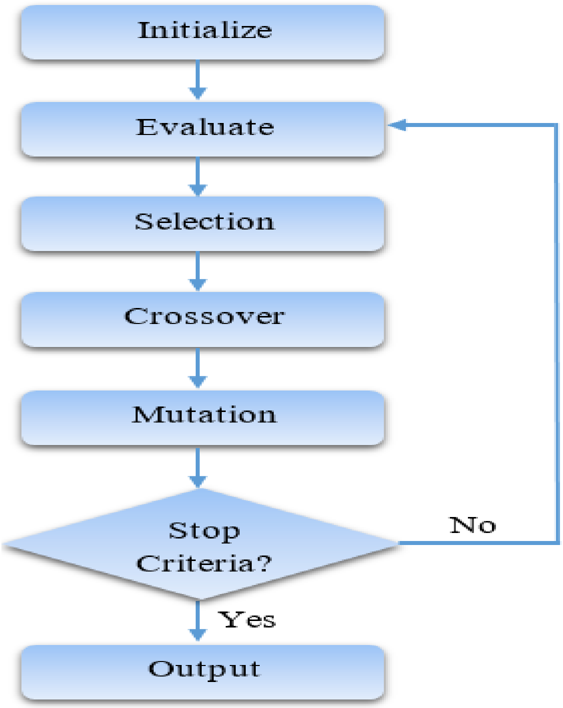

Figure (2.1): Genetic Algorithm Architecture.

Usually, the search space of a large problem is divided into smaller units that can easily be processed individually to decrease the needed computation and the required memory and storage. The abstract model for Genetic algorithm is shown in Figure (2.1).

2.3.1 Genetic Algorithm Operation

Genetic algorithm operation used to guide an algorithm through many procedures to find the best solution from a collection of potential solutions. The operations of Genetic algorithm as shown in Figure (2.1) are described in details as the following:

2.3.1.1 Population Initialization

This step is used to define the search space of the problem by dividing the solution space into single elements called individual or genome or chromosome. Every individual can be considered a possible solution. There are many representations of the individual like binary representation where the search individuals are represented by an array of zero or one. This method is not preferred for large amount of data because it needs large commutation and memory resources. Another approach is using value representation, where every position in the individual is using a single value.

2.3.1.2 Evaluation

After create the search space, we need to evaluate every individual to know its fitness or strength. This step is done by using the fitness function *f(x)* by assigning a value that represents the importance or fitness of every individual to select the best individuals to evolving and mating.

2.3.1.3 Selection

The process of selecting the best chromosomes to evolve by the next operations crossover and mutating for generating new offspring. The selection is a crucial step according to Darwinian Theory (it says that good parents produce a good generation). There are many approaches for selection. The first method is Roulette Wheel Selection or fitness proportionate selection (Bäck, 1996). It uses a probability factor to determine the probability of an individual for selecting next level by using the equation to compute the $\mathcal{P}_i$ (probability of individual i) $Pi = \frac{fi}{\sum_{j=1}^{N} fj}$ where f_i is the fitness of individual i and $\mathcal{N}$ is number of individuals in the population.

This method increases the probability of strong chromosomes be evolved and can exclude the weak chromosomes from the evolving process. Many researches claim it is not fair method for selection. The second method is Tournament selection (Miller & Goldberg, 1995); it is used to run many tournaments between few selected individuals from the population. The winner individuals among all tournaments is selected to the next level. We can control the compression by adjusting the size of every tournament.

The third method is Rank Selection (Sivanandam & Deepa, 2007), where a rank is assigned to every individual after sorting from the worst to the best. The worst is ranked as 1, the second is ranked as 2, the best is ranked as $\mathcal{N}$ rank (where N the number of individuals. This method enables the weak individual to enhance itself and produce an improved offspring.

The last method is Elitism (Shukla, Pandey & Mehrotra, 2015) method, it is based on the fact when evolving the population by crossover or mutating; we may have a chance to lose the good chromosomes and may produce a weak offspring. This method copies the whole content of the best individuals to the next generation after sorting them in descending order. Then we can use a classical method to complete the selection process.

2.3.1.4 Crossover

The Darwinian Theory is based on genes as basic units of evolution. The crossover operation is used to let any two chromosomes to be parents and produce new children. A child holds some of the characteristics of the original parents (Vekaria & Clack, 1998). The crossover operation takes two chromosomes as parents and reproduce new chromosomes as children of the reproduction process. There are many forms of crossover, one of the famous forms is single point crossover where a random point is selected for two parents and swap the rest of parents' data after the point as shown in Table (2.4).

Table (2.4): One point crossover example.

Parents, point = 3		New Offspring
A B C	D E F G H	K L M D E F G H
K L M	N O P Q R	A B C N O P Q R

The second type is two-point crossover, where two points in the parents are placed on every parent at the same point and the data between the two points are swapped to produce a new pair of chromosomes as shown in Table (2.5).

Table (2.5): Two point crossover

Parents, points 3,6			New Offspring
A B C	D E F	G H	A B C N O P G H
K L M	N O P	Q R	K L M D E F Q R

The third type is uniform crossover where the swap is placed on the gene level rather than partition level like single and two-point crossover. The uniform crossover is done by selecting random genes from one parent and swap them with their corresponding genes on the other parent and so on.

Uniform crossover use the mixing ratio, which is the ratio of the gene that must be transfer from one parent to the other. As example, if we use 0.4 as ratio that mean

40% of one parent are swapped with 40% from the second parent as shown in Table (2.6).

Table (2.6): Uniform crossover.

Ratio 40%	New Offspring
A B C D E F G H K L	A B O P E F S H K U
M N O P Q R S T V U	M N C D Q R G T V L

2.3.1.5 Mutation

Rather than swapping genes between two parents like crossover, mutation changes one or more gene values in a chromosome to arbitrary value(s). The mutation is one of the evolution methods of Genetic algorithm that uses probability of mutation factor to control the chromosome form and can change it entirely to a new form. If we use very small probability of mutation, the new offspring will be very similar to their parents and this cause to slow or stop the evolution. If we use large probability of mutation, the generated chromosomes will have new characteristics, which differ from their parents and can affect the fitness of the chromosome. An example of mutation and mutation probability factor shown in Table (2.7)

Table (2.7): Probability of mutation example.

P_m 30%	New Offspring
A B C D E F G H K L	A B O D E F S H K U

2.3.1.6 Stopping criteria

Due to the iterative nature of Genetic algorithm, it needs a mechanism for stopping the evolution process. In general, there are no standard techniques for stopping the iterative process. There are two appropriate stopping criteria; the first is time based stopping criteria (Poli, Langdon, McPhee, & Koza, 2008), where the Genetic algorithm stops after a fixed number of iteration or specified period regardless the result of evolution produced acceptable fitness. The second is based on the overall fitness of individuals (Bhandari, Murthy, & Pal, 2012). It is used when the average fitness does not have significant improvement after a number of generations.

Although Genetic algorithm makes revolution in optimization algorithms, it suffers from some potential problems. Genetic algorithm uses evolution of individuals but there is no guarantee to find the optimum solution. Another serious issue is the heavy computation needed for optimization problems making Genetic algorithm unsuitable for large-scale data sets. However, the nature makes Genetic algorithm it easily implementable in a parallel fashion.

2.3.2 Parallel Genetic Algorithm Approaches

The popularity of Genetic algorithm led researchers to develop an adaptive version for use in parallel environments. The main obstacle in Genetic algorithm is its heavy computation for computing and evaluating the fitness function of individuals. Therefore, we can use parallel systems to spread the time consuming process to several machines. Parallel environments use divide and conquer to divide the tasks over machines that are connected to a network using simple logic.

In general, there are two famous approaches for parallel Genetic algorithm. The first is fine-grain parallel Genetic algorithm. It is like sequential Genetic algorithm where the whole individuals are divided to all machines to make evaluation of every chromosome and the results are returned to the master machine. The second type is coarse-grain Genetic algorithm; every machine has its own population and implements Genetic algorithm operation lonely. This method is also called distributed or island approach, where every machine act as island and individuals can be migrated between these islands (Munawar, Wahib, Munetomo, & Akama, 2008).

Proposed researches focused on using MapReduce (Dean & Ghemawat, 2008) for parallel Genetic algorithm to solve such types of problems for large-scale data sets. Fine-grain Genetic algorithm uses a separate MapReduce job for every iteration. Every node is responsible for making basic Genetic algorithm operations and evaluation to store the best individuals to global file on distributed file system such as HDFS as explained in Section 2.5. This will divide the overhead of evaluation computation a cross the cluster nodes.

2.4 MapReduce Model

MapReduce (Dean & Ghemawat, 2008) is a programming model created by Google foundation in 2004. It is used for processing large-scale data using a cluster of commodity hardware. The paradigm uses the principle of divide and conquer to process the data in parallel fashion using two main methods, Map and Reduce. Map method process the primary data record and produce key/values. Reduce method is used, then, to make user defined operations to the values that share the same key.

Although there are many approaches for processing large-scale data like Apache Spark (Meng, et el., 2013), the popularity of MapReduce is gained because of its simplicity and abstraction. It can deal with structured or unstructured data and allow programmers to write their own code using many programming APIs. In addition to that, MapReduce provides high scalability, high fault tolerance, high availability and efficient load balancing mechanism.

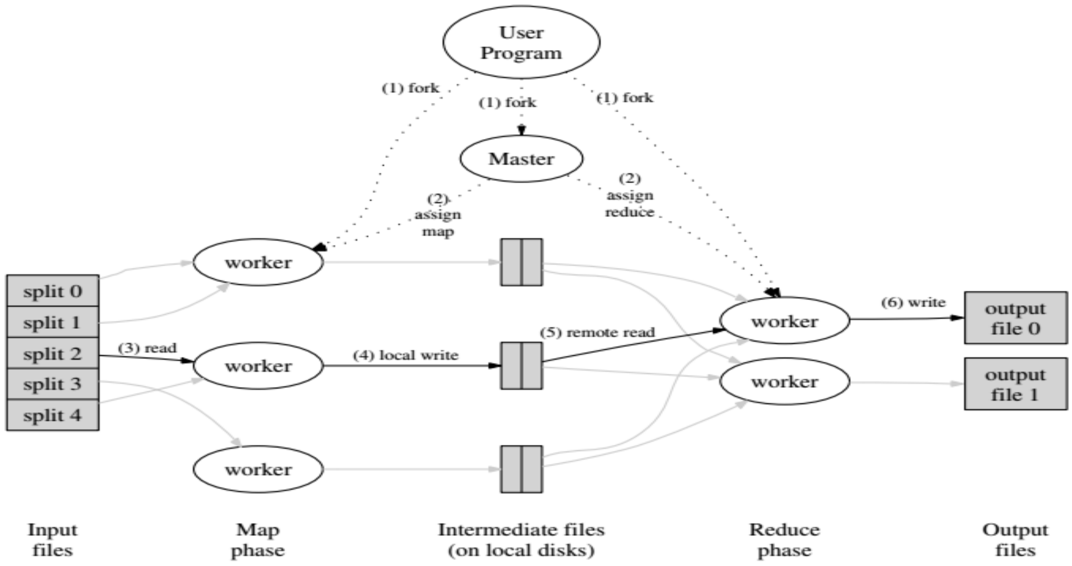

Figure (2.2): MapReduce execution overview (Dean & Ghemawat, 2008).

MapReduce run programs automatically a parallelized fashion and run on a large commodity cluster of machines. The system handles all the details of partitioning the data and scheduling the execution of the use program between the machines. Therefore, enabling programmers to focus on the program logic and let the details of parallelization of the system and decrease the lines of code. MapReduce model has many features such as:

1- Scalability
2- Availability
3- More Flexibility
4- Fault Tolerance
5- Rack awareness
6- Simple programming model

2.4.1 MapReduce job workflow

As shown in Figure (2.2), MapReduce executes a job as follow:

- Divide the data file into small chuck and send them to Map task on every machine on the cluster.
- The Map process the data, generate intermediate key to every record, and send them to the main thread.
- Intermediate key/value records sorted by key. Moreover, send the values that have the same key to the same reducer.
- The reducer applied to the key/value records and do the required operations. Finally, after reduce complete on all machines the control returns back the main program to end execution and save results on HDFS.

2.5 Hadoop as MapReduce Realization

Hadoop (White, 2012) is an open source realization of MapReduce. Written in Java and incorporate open source distributed storage, such as Hadoop Distributed File System (HDFS).

2.5.1 Hadoop Architecture

Based on the model of MapReduce, Hadoop has the following parts:

- Masternode: is the manager of all Hadoop services and nodes. It is control the execution of user jobs and responsible for make the communication between the workers in the cluster.
- Slavenode: is responsible for serving the commands from the master node by making read and write commands from the file system's clients along with perform block creation, deletion, and replication upon instruction from the Master.
- Namenode: is the part of Hadoop that manage and coordinate the files metadata like files directories and folders structure. Moreover, Namenode server make a log for every operations and transactions of stored file. In addition, it is make commands for creation of file chuck replications if a server goes failure. Namenode server is a crucial part of Hadoop; therefore, there is only one Namenode server.
- Datanode: is the physical store of Hadoop files block. The Datanode stores the entire file and if the file size larger than the default 128 Mb, its partition the file

into smaller parts and replicate it on the other machines in the cluster. Add that it is notify the Namenode to any change on files or folders to store its metadata.

- Job Tracker: is the interface of job execution between users and the MapReduce model. When a user starts a MapReduce job, Jobtracker place it in a queue of pending jobs and executes them on a first-in/first-out basis and then manages the assignment of map and reduce tasks to the task trackers.
- Task Tracker: is a slave to the Job Tracker, it manages to run the tasks received from the Job Tracker and make reports on job execution to Job Tracker.
- YARN (Yet another Resource Negotiator): is the next generation of Hadoop's compute platform (Vavilapalli et al., 2013). It is managing all the resources of Hadoop. It communicates with the client, tracks resources on the cluster, and orchestrates work by assigning tasks to Node Managers, which allows a new concept for big data analytic applications such as interactive SQL, real-time streaming.

2.5.2 HDFS

Hadoop Distributed File System (HDFS) is a distributed file system designed to store and access large scale and store structures or unstructured data (Borthakur, 2008) and use commodity hardware run its services. HDFS use the hosted operating system file system to storing large data by partitioning it to small chucks as block. By default, the block size is 128Mb and this make it deal better with large file than small file (Gunarathne, 2015). As shown in Figure (2.3), every block replicated on all other cluster machines to ensure reliability and fault tolerance, therefore the high availability if any machine goes down.

Like the Master/Slave architecture, HDFS consists of two main components as shown in Section 2.5.1 Hadoop Architecture). The first is Namenode server and Datanode server. The Namenode responsible for storing the Meta data of the structure of distributed file system. Wherefore, only one Namenode server for single cluster and no physical data blocks stored on Namenode server. On the other side, Datanode is responsible for creating, deleting, and replication of large files using the operating system file system. Datanode also receive instruction from Namenode server to serve uses requests.

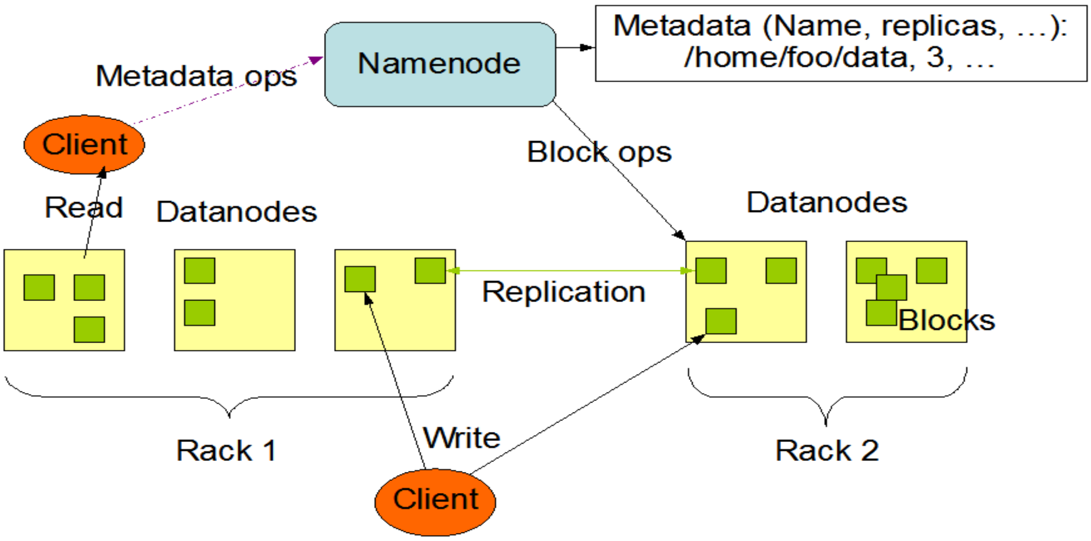

Figure (2.3): HDFS Architecture (Gunarathne, 2015).

In general, HDFS use the normal hierarchical structure for creating files and directories like traditional file systems using traditional Linux file commands for managing files, which make it easy to use.

2.6 Data Mining for Large Data Sets

As mentioned, MapReduce gets its popularity in solving problems for large-scale data using simple logic and commodity hardware. Data mining is an important field in information science, however the growth of information due to internet era make mining large data set a challenging task. This require using new approaches for processing such data using different logic. Many researches turn to use MapReduce model due to its simplicity and robustness.

Data mining usually deals with heterogeneous or unstructured data to extract useful knowledge from massive datasets. This fact makes MapReduce a suitable solution for large-scale data mining applications. MapReduce facilitates the way for reading, processing and storing data which make MapReduce have the required flexibility for many data mining tasks. Many researches use MapReduce for solving data mining problems. (Zhao, Ma, & He, 2009) use MapReduce to implement a fast clustering algorithm K-mean and resulting in high speed up and to scalable result. Another approach proposed by (Kang & Faloutsos, 2013) for mining social media networks using graph algorithm which is difficult to be implemented by a single machine.

2.7 Summary

In this Chapter, we provide an overview of the technical and theoretical foundations related to the research problem. We introduce text summarization and its related issues for Arabic language. In addition to that, we introduce important notions in the area of Arabic natural language processing and some useful tasks for making summarisation of Arabic text. Genetic algorithm and its operations are also described and the need of Genetic algorithm for intensive computation for large-scale data. Finally, we introduce the MapReduce model and its realization, Hadoop, with important features for simplifying complex computations.

Chapter 3

Related Works

Many research for automatic text summarization focus on transforming large text documents into small summary. For English language, there are numerous amount of text summarization research for multi document as well as single documents. However, little effort is exerted for Arabic language text summarization in general. The main reason is that the Arabic language has a rich morphology and is highly derivational and the lack of tool for making NLP for Arabic language.

We review a number of works in text summarization for Arabic and English language. First we present works that employ Genetic algorithm in text summarization using different approaches. In addition to that, many methods use clustering and graph based approaches for making summarization of text. Finally, we present many works use MapReduce model for solving common data mining tasks.

3.1 Text Summarization Using Genetic Algorithm

One of the earlier models that integrate summarization and Genetic algorithm is proposed by (Y.-X. He, Liu, Ji, Yang, & Teng, 2006). They propose a model for multiple documents summarization using Genetic algorithm for Chinese concept lexicon and corpus, the objective of their proposed work is to maximize the coverage of topics and minimize the redundancy of contents. They use semantic analysis and statistical techniques for improving the Cross-document Structure Theory (CST) by representing the documents and elements and their relationships as a network and improving the quality of the result by applying Genetic algorithm.

In (Qazvinian, Hassanabadi, & Halavati, 2008), a model is proposed for making automatic text summarization that is based on a fitness function that evaluate and produces summary based on three factors: Readability factor, Cohesion factor, and Topic-Relation factor. The important sentence is extracted using weighted features and using some coefficients to let the user control factor to select his favourite feature but their method need to improve the cohesion of their extracted summaries using various techniques.

Another utilization of automatic text summarization is to help people having problems like reading difficulties (Nandhini & Balasundaram, 2013). The objective is to generate single document summary that makes a balance between F-measure, readability, and cohesion using Genetic algorithm. To find the best combination of sentences, they use important weighted features to make summary with high informative score and good cohesion. Their method provides good results but suffers from high computation when the summarizing large documents.

Many approaches for text summarization is language independent, which are based on statistical methods that are based on features extracted from a text. In (Litvak, Last, & Friedman, 2010) an approach is used for making single document text summarization based on a linear combination for a list of features. They use a trainable Genetic algorithm to select the best sentences to make summary of a document. They use the summarization evaluation ROUGE (Lin, 2005) recall as the fitness function to weight the sentences of the summary. Their method does not use similarity methods to handle the redundancy and is inefficient for large text documents.

Genetic algorithm is also used for multi document summarization. (Bossard & Rodrigues, 2010) propose a method for multi document summarization using Genetic algorithm that is based on clustering methods. The centre of the cluster is created based on the highest ranked terms based on term frequency method TF-IDF. Sentences are evaluated based on their similarity to the cluster centre. Sentence redundancy is detected at early stage at sentence selection step. Their system does not use linguistics methods for sentence selection that can improve the quality of summaries. In addition, the approach cannot deal with large documents due to Genetic algorithm high computation.

3.2 Text Summarization Using Feature Extraction

In the field of automatic text summarization, extracting sentences from text using features is a common method. It is based on the importance of text parts like sentences, word and phrases using some statistical methods. These methods are dealing with assigning a score for every extracted unit and ranking the overall system by aggregating all of these features. There are many types of feature extraction methods. One of the first methods is significant sentences matures (Luhn, 1958) which score sentences based on important words. Another important method is sentence position (Baxendale, 1958), they depend on a study that found 80% of the important part of a paragraph is the first sentence. Another approach proposed by (Harold P Edmundson, 1969) shows that the automatic summarization of text can be done by extracting text components with high frequency content like important words (cue words), title and heading words and positions of sentences. In our approach, we use some of these features in the Genetic algorithm evaluation mechanism like sentence position and sentence length.

Due to the complexity and lack of sophisticated tools, Arabic language, little automatic text summarization researches are carried out. Using feature-extracting approach. (Hewahi & Kwaik, 2012) propose a model for making text summarization for single Arabic text documents using features list extracted from the Arabic text. Their system uses linear combination of features for ranking sentences and selecting the top rank as selected summary. Semantic similarity is used to measure the similarity between sentences and sentences with title. Moreover, they use some important features like named entity and place to improve the summary.

These features may show that a sentence has important information and will give better results. They compare their system with Sakhr Arabic online summarization system and show that their system overcome it. This system is not sufficient for processing large amount of data and it does not use modern NLP tools to identify the important sentences.

3.3 Text Summarization Using Clustering

Using fast statistical computation, a model proposed by (Goldstein et al., 2000) is based on Maximal Marginal Relevance (MMR). They use this method to summarize multi document based on input keyword (user query) by computing the similarity using cosine similarity between the input query and the input text and partially generated summary and the current chosen sentence. The MMR gives a weight to the sentence based on relevance and redundancy with the selected sentences in the summary. Their model gives good results but it does not involve any linguistic techniques and cannot deal with large amount of data.

MEAD model (Radev, Jing, Styś, & Tam, 2004) is multi document text summarization that is based on clustering the sentences of group of documents based on a shared event by choosing a centroid of each cluster and relate a sentence to the closest centroid by ranking. The centroid is consisting of group of words that exist in all documents in a cluster. They use weighted features like centroid value, positional value, and first-sentence overlap to select which sentence is included in the summary. To detect redundancy of sentences, a negative value is added to the score of a sentence that overlap with a sentence that has a higher score. Although MEAD system is a sophisticated summarization approach, it ignores many important features such as readability and does not designed to use with large data.

A state of the art model proposed by (Nagwani, 2015) makes summarization of large scale single text documents using MapReduce model. It is based on semantic clustering topic modelling using Latent Dirichlet Allocation (LDA). They use clustering to group the similar documents using k-mean clustering algorithm then apply LDA to extract the topic related to each document in the cluster. They use MapReduce to achieve scalability and speed up the processing time. To evaluate their model, they use ROUGE-1 (Lin, 2004) and pyramid scores. They achieve good scalability due to their results and they successfully make their system scale to make summarization to handle these types of text documents. Nevertheless, summarization of single document is not complicated like multiple documents. Consequently, the approach we proposed will use many methods to make multiple document summarizations for large-scale text documents.

Another approach of multiple Arabic documents summarization is proposed by (El-Ghannam & El-Shishtawy, 2014). It ranks document sentences based on key phrases. They build a cluster of key phrases from the documents, rank these key phrases, and give a score to every key phrase. They evaluate every sentence in the cluster of document sentences based on the score of key phrases that exist in that sentence to choose one sentence that represents an important topic. Their work emphasize that key phrases can improve the summarization.

There are several work to make summarization for Arabic documents. (Waheeb & Husni, 2014) propose an approach to makes automatic summarization of Arabic multiple documents using clustering approaches. They use k-mean clustering algorithm to cluster the sentences of pre-classified Arabic documents. They focus on increasing the quality of produced summaries by eliminating the redundant information and noisy data by using similarity measures and language processing tools. They use recall and precision metrics to evaluate the clustering performance and this leads to high weight sentences, which improve the recall and precision. The result of their approach support our ideas to make text summarization for large-scale documents based on weight of sentences.

A model proposed by (Froud, Lachkar, & Ouatik, 2013) depends on Latent Semantic Analysis to make text summarization for single Arabic document. Their method solves the problem of noisy data to make term clustering more accurate. Rather than clustering the sentences of the document, they cluster the important terms. They extract the important sentences based on the relative weighted important terms. They do not use semantic similarity measures to improve the clustering of documents therefore affecting the selection mechanism of related sentences. However, they do not deal with multi document that make summarization a hard task due to diversity of topic.

Several text summarization approaches focus on using scoring sentences that are based on common features. Shortcoming of this is that many features do not have the large impact on the overall score. A modern method for Arabic language proposed by (Oufaida, Nouali, & Blache, 2014) make single and multiple summaries for Arabic text using Minimum Redundancy Maximum Relevance (mRMR). It is used to get the features that select a group of features that represent the whole list of features. Hierarchical clustering method used to group the sentences into related clusters. This step is important before scoring sentences using mRMR. To detect redundancy, they add a score to every sentence that detect if a sentence share information with other sentences to decrease the ranking. Based on their results, their system use minimum language analysis methods lead to enough speed but the overall result gives low ROUGE (Lin, 2005) scores especially in multi document summarization.

3.4 Text Summarization Using Graph Approach

Graph approach is gaining more popularity for making automatic text summarization due to its simplicity and speed. It is constructed from a directed graph containing vertices and edges G = (V , E). In the area of text summarization, we can represent a document as a graph by transforming sentences into vertices or nodes and the relationship between them as edges.

A model proposed by (John & Wilscy, 2015) for making automatic text summarization for single and multiple documents using Vertex Cover algorithm. They use sentences as the vertices and the similarity between the sentences as edge weights between their corresponding vertices. They use cosine similarity and Normalized Google Distance to measure the similarity between sentences. In addition, they use Term Frequency (TF) and Inverse Sentence Frequency (ISF) to weight and evaluate the important words. For correctly finding ideal summary, they select subset of the graph containing the high weighted sentences that cover the important concepts and taking into consideration the redundancy problem in the generated summary. Result of graph systems overcome many systems that use clustering, but they do not work well for large documents because the graph size increases and consequently computation time increased.

(Zhang, Sun, & Zhou, 2005) propose a graph model for performing summarization for multiple documents, that combine the surface features with the content features using Hub/Authority framework. They use first sentence, cue phrase and sentences length features to identify important sentences. They use clustering approaches to identify the sub topics for a collection of documents. To build a graph, feature words and cue phrase are used as the vertex of Hub and sentences regarded as the vertex of authority. An edge is placed between an authority sentence and hub word if that sentence contains a word in the hub. This is step important to make the relationships between edges and its related authorities. Final summary generation is done by ranking sentences of every sub topic and order sentences based on its rank.

TextRank (Thakkar, Dharaskar, & Chandak, 2010) is an important method to make automatic text summarization. Its uses a graph of sentences to find the best path starting with the first sentence and ending with the last sentence. This method builds a graph consisting of sentences as nodes and there is an edge for connecting any two sentences. They use cosine similarity to find the edge strength and give a weight of every sentence in the graph. The summary constructed by selecting a group sentences occurs on the shortest path between the first and the last sentence of a document. This method selects the sentences that are related to other sentences, therefore decrease the sudden shift of information and maintain the flow of information of the summary.

3.5 MapReduce in Data Mining

Data mining is an important field in computer science, the increase of data requires using new models and algorithms to process large data amount especially for cloud environments. MapReduce is an attractable option for using data mining for large data sets. Many recent researches are carried out by applying MapReduce model for many data mining tasks like classification (Wu et al., 2009) (Q. He, Zhuang, Li, & Shi, 2010) (Rao & Yarowsky, 2009) (Paniagua, Flores, & Srirama, 2012). Another important task is clustering. (Hans, Mahajan, & Omkar, 2015) (Ene, Im, & Moseley, 2011), (Ferreira Cordeiro et al., 2011) and (Zhao et al., 2009) proposed clustering models based on MapReduce and show that MapReduce can be effectively used for these type of problems and give good results in terms of speed up and scalability. This encourage improving the quality of the data mining application for the big data environment.

3.6 Genetic Algorithm over MapReduce

MapReduce is popular for processing large-scale datasets while Genetic algorithm is sufficient for problems dealing with searching and optimization techniques. One of the main problems of Genetic algorithm is the evaluation of the individuals of the search space. The time increase incredibly when the number of variables increases. There are many works that use Genetic algorithm in large-scale data. (Saha, 2014), (Geronimo, Ferrucci, Murolo, & Sarro, 2012), (Verma, Llora, Goldberg, & Campbell, 2009) and (Jin, Vecchiola, & Buyya, 2008) implement Genetic algorithm in many fashions. Simple Genetic algorithm is the basic and it is very similar to the classical Genetic algorithm. One approach is the Island approach, where the data is divided into separate islands and a node processes every island with the possibility of migrating individuals from one island to another. The results they achieve show that Genetic algorithm can be scaled by using MapReduce to give acceptable speed-up and scalability.

In the above works of automatic text summarization, we notice that the approaches they propose have some advantages and limitations. The main limitation is that they do not have the ability of processing large text documents. Therefore, a new model for performing automatic text summarization on large-scale documents is required. MapReduce will be used to speed-up the Genetic algorithm processing time.

3.7 Summary

In this Chapter, we provided various works related to our research problem. These identify many techniques for text summarization like clustering, feature extraction, Genetic algorithm, and graph approaches. We show drawbacks of these approaches of text summarization especially for large size datasets. We show that MapReduce has advantages for data mining applications like clustering, classification and summarization.

Chapter 4

Multi Document Summarization System Design

To carry out automatic text summarization for large-scale Arabic text using Genetic algorithm, the proposed approach is divided into several phases. Starting with the pre-processing phase and concluded with evaluation phase. The system architecture is illustrated in Figure (4.1). The proposed approach consists of four main components.

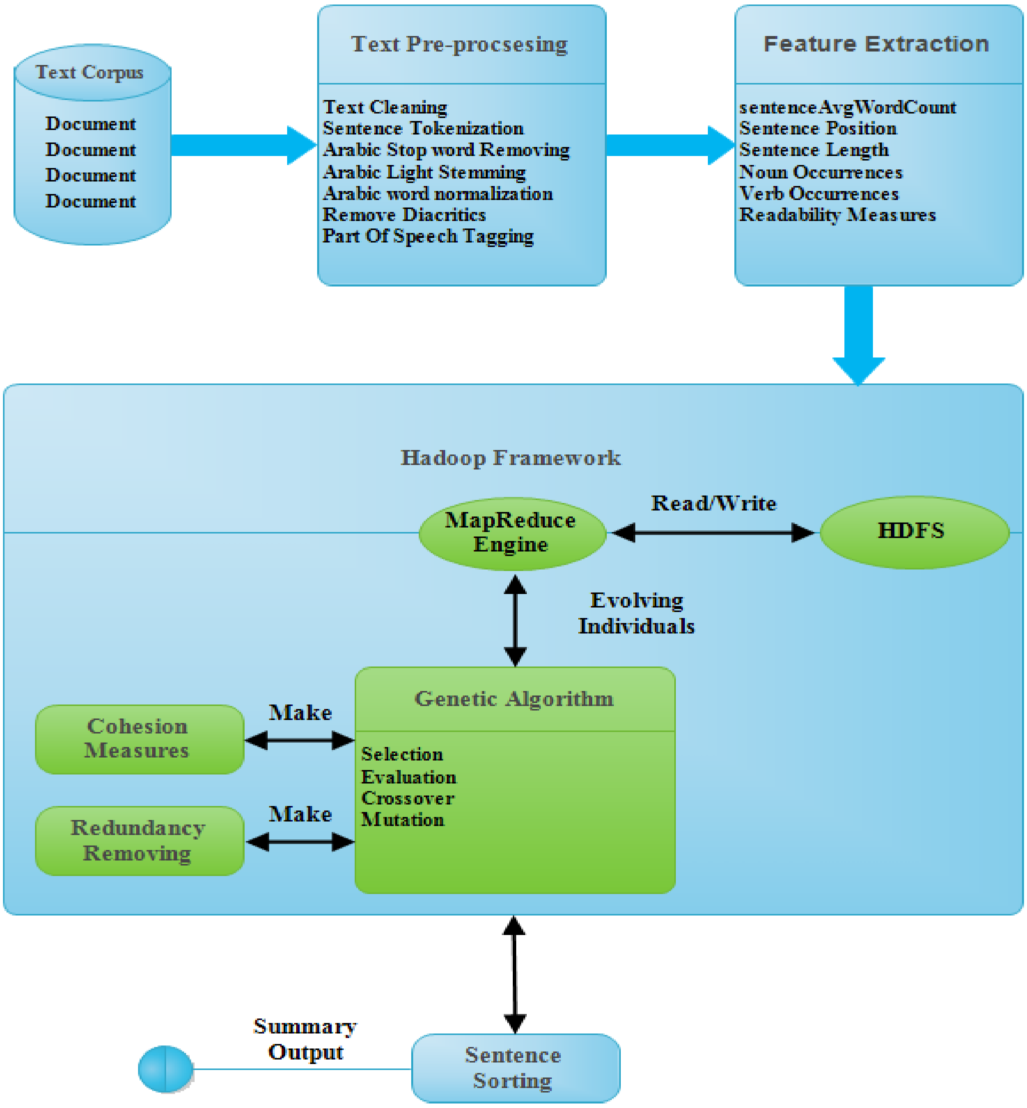

Figure (4.1): Multi Document Summarization architecture.

The first component is preprocessing component make some operations for the text like cleaning and stemming and POS to prepare the text for feature extraction phase. The second component is feature extraction component used to extract many features from the text which essential for computing the fitness of the every individual in Genetic algorithm phase. The third component is Genetic algorithm, which have an iterative nature, represents the core operations and use MapReduce model to speed up the computation and ensure high scalability. In addition to that, this component used to compute the cohesion of every individual and remove redundant sentences from every individual. Finally, the sorting component, which sort the sentences of the winner individual and present the final summary.

4.1 Text Pre-processing

Text preprocessing is an important process that is taken into account before any text processing. It affects the result of the text processing applications because it removes many noisy data and important to prepare the text for processing. We preformed the following preprocessing steps in the proposed approach as illustrated in Figure (4.2).

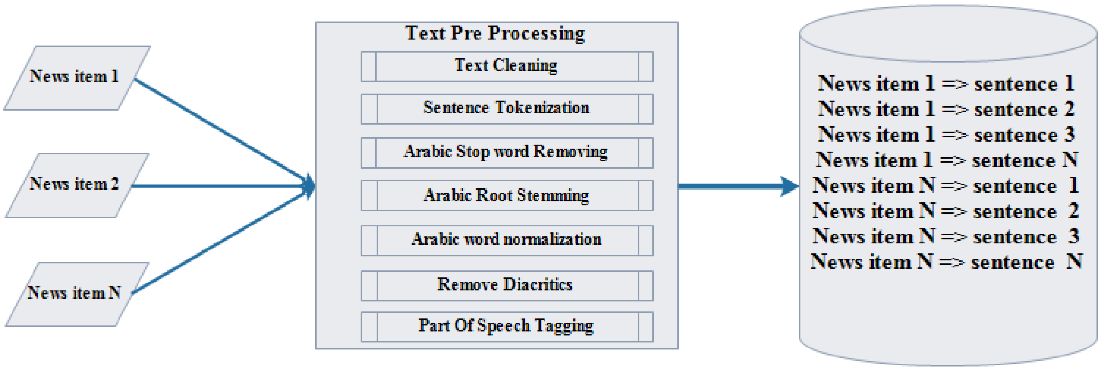
News item 1
News item 2
News item N
Text Pre Processing
Text Cleaning
Sentence Tokenization
Arabic Stop word Removing
Arabic Root Stemming
Arabic word normalization
Remove Diacritics
Part Of Speech Tagging
News item 1 => sentence 1
News item 1 => sentence 2
News item 1 => sentence 3
News item 1 => sentence N
News item N => sentence 1
News item N => sentence 2
News item N => sentence 3
News item N => sentence N

Figure (4.2): Text Pre-Processing

4.1.1 Text Cleaning

Text cleaning is an important process, which precedes any data mining application. It is used to prepare the data for further processing. The collected text is from several Arabic Palestinian newspapers. It contains more than 830 thousand news items and about 70 different categories. This requires normalizing the number of categories into few categories. Another issue is removing formatting where news editors use some online rich text editors to format the news and add some features like quotes and hashtags, which are stored as special codes in the database.

Therefore, the news is stored as a mix of normal Arabic text, HTML tags and some special code for formatting the news text for the web. The first step is removing HTML tags, the special formatting codes and the hash tags. Since the news are collected from many websites, text cleaning becomes a difficult process due to the diversity of special formatting codes. Also we remove non Arabic words and the punctuations from the text.

4.1.2 Sentence Tokenization

Usually in morphological analysis, many NLP techniques starts with the tokenization process to divide a streaming of text into words, phrases or sentences. For Arabic multi document summarization, we need tokenization to break every news item into a collection of sentences using some common delimiters like dot, question mark, exclamation mark and new line. These marks determine the border of every sentence and consider the words near to it as new sentence. However, advanced morphology focus on special boundary that is not considered here such as abbreviations like Dr. Mr. and real number like 15.015, which make the tokenization process more complex.

This requires some attention when processing such cases because they are not marked as sentence boundaries and require some attention. The final output of this process is stored as list of sentences for every news item. We should do this step because we need to extract some text features from all sentences as explained in Section 4.2 like sentence position and sentence length, which depends on all sentences of single news item.

4.1.3 Arabic Stop Word Removing

Stop words is a commonly used word in any language used to connect part of speech and we need to ignore them when weighting sentences to save text-processing time and focus on the important terms. Therefore, we need to apply similarity measures like cosine similarity. The stop word has no impact on the importance of a text because it is repeated many times. Stop word is language dependent and is constructed manually using stop word list.

4.1.4 Arabic Root Stemming

As described in Section 2.1.2, stemming is the process of removing some different affixes in any place of a word to return it to its root. We can use root stemmer to get the root of any word or light stemming to remove the prefixed and suffixes only. In multi document summarization, we use root stemmer because we need to know the basic information of a news and return many word styles that belong to a single root. This is required in the term weighting process in Section 4.2.9, for similarity measures, which used in detecting similar sentences in redundancy removing process and for text cohesion measures based on common words roots.

4.1.5 Arabic Word Normalization

Arabic have a complex morphological structure; the letters can be written in many forms. Before performing any processing to the text, we need to uniform the similar letters into one form. For example, the letter Alef can be written in more than four forms and the Yaa Letter in three forms as explained in Table (4.1) and all these forms belong to one form.

Table (4.1): Arabic Words Normalization

Letter	Forms
ا	أ إ آ ا
ى	ي ئ ى

4.1.6 Remove Diacritics

Diacritics are used to make the Arabic text reading easier, therefore, usually beginners use it to spell and understand the text correctly as explained in Table (4.2). In text processing, we need to remove these diacritics because they do not provide any importance to the text.

Table (4.2): Arabic diacritics.

Arabic common diacritics
عٌ عُ عً عَ عٍ عِ عْ ع

4.1.7 Part of Speech Tagging (POS)

In the POS task, we identifying the types of Arabic words as explained in Section 2.1. POS helps us to know the important part, which provide important information. We mark every word to the appropriate category in terms of noun, verb, adjective and many other types. This step is required for identifying the number of each category in every sentence to use it in sentence weighting and in the feature extracting process.

4.2 Feature Extraction

Feature extraction methods are widely used in text summarization applications. The first summarization proposed by (Luhn, 1958) uses the occurrences of important word feature in a text to build a summary. We use the extracted features from a collection of documents to assign a value for every sentence in the big corpus. Genetic algorithm uses the fitness function to rank and determines the strength of every individual in the population. Our approach is dealing with extracting some importance features from the Arabic text to use it for computing the fitness function of chromosomes.

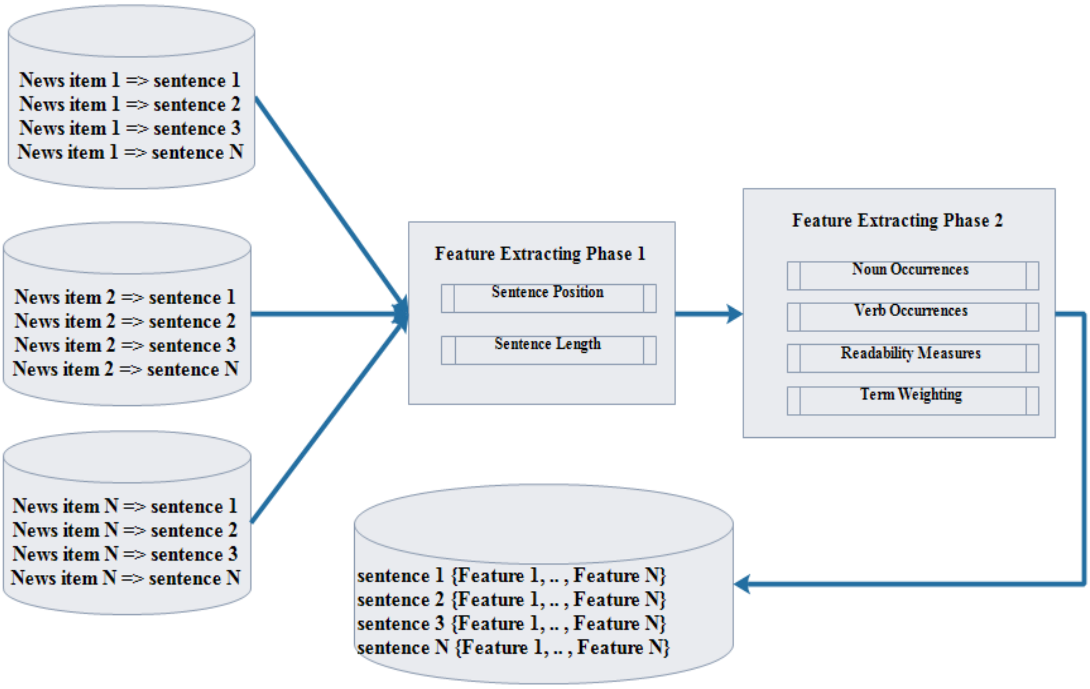
News item 1 => sentence 1
News item 1 => sentence 2
News item 1 => sentence 3
News item 1 => sentence N
News item 2 => sentence 1
News item 2 => sentence 2
News item 2 => sentence 3
News item 2 => sentence N
News item N => sentence 1
News item N => sentence 2
News item N => sentence 3
News item N => sentence N
Feature Extracting Phase 1
Sentence Position
Sentence Length
Feature Extracting Phase 2
Noun Occurrences
Verb Occurrences
Readability Measures
Term Weighting
sentence 1 {Feature 1, .. , Feature N}
sentence 2 {Feature 1, .. , Feature N}
sentence 3 {Feature 1, .. , Feature N}
sentence N {Feature 1, .. , Feature N}

Figure (4.3): Feature Extraction

Most of these features are computed only one time after text pre-processing phase. This is because it is time-consuming process; therefore, it is computed only once and the same features is used at every Genetic algorithm iteration. The text cohesion feature is computed every iteration because the summary content is changed during the crossover and mutation operation. The feature extraction phase of the proposed approach is illustrated in Figure (4.3).

4.2.1 Sentence Position

Usually, the first sentences and the last sentences contain information more than the other sentences (H. P. Edmundson, 1969). We can compute the sentence position weight in the document using the following formula:

$$Sp = \frac{N-i+1}{N} \qquad \text{Equation 5.1}$$

where $\mathcal{N}$ is the number of sentences in a document and i is the position of sentence.

4.2.2 Sentence Length

The purpose of summarization is to provide the most important information from a collection of documents. Therefore, we should take into account the length of a sentence the very short sentence does not contain enough information. so we do not consider it as important (Al-Hashemi, 2010). On the other hand, a very long sentence makes the reader unable to understand and integrate the context from the content. We can find the sentence length feature from the equation (Suanmali, Salim, & Binwahlan, 2010):

$$\frac{\#(words\ in\ Sentence\ i)}{\#(words\ in\ longest\ sentence)} \qquad \text{Equation 5.2}$$

Sentence position and sentence length features should be computed together after cleaning the news item because they are depending on the whole news item.

4.2.3 Noun Occurrences

Using part of speech tagging POS, the number of noun words can be calculated. Many studies shows that the presence of nouns make the text more readable and this is because sentences contain more noun words which may have important information (Feng, Elhadad, & Huenerfauth, 2009). Another study proposed by (Bouras, Tsogkas, 2008) shows that using noun in text summarization can improve the precision scores.

4.2.4 Verb Occurrences

In most languages, the verb is an important part of a sentence, which express about using some action and make readers know about occurring events in the text. Sentences contain verbs may include and cover some events and shows that these sentences describe an important event. This step is required in the feature extraction phase and is done by counting the number of words, which tagged as verb words in every sentence by using part of speech tagging POS.

4.2.5 Readability Measures

As shown in Section 2.2.5, the readability test is used to score text in terms of how the text is easy to read. An important feature of the summary should contain easy sentences that can be read from different readers from different ages. Using Osman readability measure (El-Haj & Rayson, 2016), the readability of each sentence in the corpus is measured such that it can be used in the sentence scoring for the summary generation. This will ensure that the individual, which consisted of a group of sentences and have a high readability score is more easy to read from those have low score.

4.2.6 Cohesion Measures

Multi document summarization creates a summary from multiple sources, therefore the summary may contain sentences from deferent topics. As shown in Section 2.2.6, text cohesion measures show how these sentences are connected and related to each other. In our proposed approach, the Genetic algorithm fitness function measures the text cohesion feature of all individuals at every iteration. This improve the quality of the summary and gives high score to the summary, which contains similar sentences. In general, there are many approaches to measure the cohesion of a text. An approach proposed by (Nandhini & Balasundaram, 2013) uses cosine similarity to measure the cohesion of a summary.

In order to measure the cohesion of our multi document summarization approach, we depend on the root stemming of each sentence to build a similarity matrix and compute the average similarity of each sentence as the text cohesion of the potential summary. Every entry of the similarity matrix defines the cosine similarity of the stemmed sentence words for all sentences. We consider the similarity of the same sentence in the column and row to be equal to zero rather than one because sentences with cosine similarity are identical. The cohesion of the individuals of Genetic algorithm population is computed using the following equation:

$$sim(sen1, sen2) = \frac{\sum_{i=1}^{n} Ai\ Bi}{\sqrt{\sum_{i=1}^{n} Ai} . \sqrt{\sum_{i=1}^{n} Bi}} \qquad \text{Equation 5.3}$$

Where A, B is the vector of word of the stemmed of sentence A and the words of stemmed of sentence B (Sidorov, Gelbukh, Gomez-Adorno, and Pinto, 2014). We should notice that the text cohesion feature is computed at every iteration of Genetic algorithm. This is because the content of individuals may change during the crossover operation as explained in Section 4.3.

4.2.7 Term Weighting

The purpose of multi document summarization is to select the most important information from a collection of documents. Therefore, we must select the sentences that contain important information using the most frequent term. There are many methods to know the important words in a text document. Term Frequency & Inverse Document Frequency TF-IDF is one of the early methods to rank terms of documents. It is a statistic method used to know how a term exists in a document and across multiple documents. The first part is Term Frequency (TF), which reflect the number of times a term T occurs in a document D. The Inverse Document Frequency (IDF) represents how many times a term T occurs in all documents of a text corpus (Salton & Buckley, 1988).

Our approach, partitions all sentences into groups of possible summaries and every summary consisted of fixed number of sentences. Every summary represents a single document and therefore, the IDF measure is not efficient because it measures the frequency of a term in groups of documents while our approach based on a single summary, which represents a single document. We only choose the TF measure by performing single MapReduce job to count the terms of the whole population. We give a weight for each sentence in the population as the aggregate occurrence of each term in the sentence. Next in Section 4.3, we present how Genetic algorithm can be designed on MapReduce model.

4.3 Designing Genetic Algorithm as MapReduce

In order to make multi document summarization using Genetic algorithm over MapReduce model to speed up the summarization. The proposed approach performs the following. Firstly, the sentences are stored after tokenization with their features, which are extracted in Section 2.2 in the following format: sentenceID, sentenceText, sentenceStems, List<sentenceFeatures>. This process eliminates the computation time needed for calculating the sentence features in every Genetic algorithm iteration. The iterative Genetic algorithm summarization starts with dividing the data into separate individuals and evaluate them to select the best individuals for evolving using crossover operator. This insures the iterative nature of Genetic algorithm over MapReduce as described in details in next sections.

4.3.1 Creating Individuals

Genetic algorithm is implemented over Hadoop, a MapReduce programming model. Firstly, we need to define the data structure of Genetic algorithm. Gene is the smallest part of Genetic algorithm. It stores single part of information of the chromosome (individual). The chromosome is a collection of genes, which holds larger amount of characteristics. In the case of automatic document summarization of multiple Arabic documents, the sentence is the smallest part of information; therefore, gene is a single sentence. To make a summary we need a collection of sentences and thus the chromosome consists of a collection (array) of sentences selected randomly from different source documents.

To summarize the first step, we need to shuffle and randomly arrange the sentences stored on the HDFS into a new file containing all shuffled individuals. For traditional summarization applications, the compression ratio concept controls the size of the resulting summary. In our case, the output summary is large. Our proposed approach uses a fixed length individual by splitting the whole dataset into typical length. Every chromosome has a form consisted of sentence id, sentence text, sentence stems and sentence features. The new data are stored into a new file on HDFS where every line is a single chromosome.

4.3.2 Ranking Individuals

This step involves in computing the fitness function of the individuals generated from the previous step. The fitness function value is computed using linear combination of the pre-computed feature except cohesion feature. Firstly, cohesion feature value is used to determine how much the sentences of the individuals belong to each other using similarity measure, like cosine similarity measure. After this step, every individual is stored with its fitness function value. In addition to that, we need to remove the weak sentences by conducting a tournament to make the evolution of only strong individuals.

The previous steps are performed in parallel fashion to speed evaluating the individuals, which is time consuming operation in Genetic algorithm. Therefore, this step prepares the iterative nature of Genetic algorithm and the next steps are involving improving the fitness and evolving the individuals via crossover.

At the end of each iteration we find the best individuals and store them for the next iteration and so on, we will stop the execution when no major change occurs in the fitness of the individual. The final schema of individual consists of individual sentences with their features and the fitness function value of the chromosome in the current iteration. This step is important to minimize the computation overhead for computing the fitness before evolving the individuals in the next step.

4.3.3 Iterative Simple Genetic Algorithm

As shown in Section 2.3, Genetic algorithm has many forms for parallel environments, one of the simplest form select-recombination (Goldberg, 1989), which can be transformed into parallel execution with minimal set of operations (Verma et al., 2009). The basic idea behind this approach is to select good individuals from the population and evaluate them to produce new generation and repeat the operation until meeting some convergence criteria.

Initially, individuals are stored into HDFS with their features and fitness. The evolving process of Genetic algorithm is performed in iterative nature using single MapReduce job for every iteration. This step is dealing with evolution using the crossover operation and comparing the final fitness of every iteration with the fitness of the previous iteration. This is required to decide whether to start a new iteration or terminate the iterative Genetic algorithm.

4.3.4 Evolving Individuals

Genetic algorithm is used to improve the fitness of the new generations by choosing a strong parent to mate them. The process of creating new individuals uses two main Genetic operation crossover and mutation. However, in data intensive systems the mutation operation becomes more difficult. The main reason is the nature of the summarization process required to replace a gene that contains a single sentence with a gene having a new sentence randomly. This requires searching in HDFS large file and selecting a random sentence, which does not exist in the individual and do the same for all individuals. This step increases the resource consumption incredibly therefore; we may neglect the mutation operation without the need of it is improving the individual fitness.

Due to its simplicity, the crossover operation can easily be implemented in parallel in data-intensive systems. In its simplest form crossover is used to swap a fixed number of genes from any two individuals. This process is performed at every MapReduce iteration to generate an improved new individual. Section 5.2.3 shows the details of implementing the iterative nature of Genetic algorithm using MapReduce, conducting tournament and how to deal with redundant sentences in the individuals during the uniformed crossover operation.

4.3.5 Stop Iterative MapReduce

After every iteration, we should monitor the maximum. The concept of select-recombine Genetic algorithm as shown in Section 4.3.3 uses the iteration to be from 3 to 5 times. Initially, we make 3 to 5 Genetic algorithm iterations and after that we monitor the maximum fitness, if it is greater than the fitness of the previous iteration, we start a new iteration. If it is less than the fitness of the previous iteration, we terminate the MapReduce and use the previous individual as the ideal summary with the individual that have the maximum fitness. Section 5.2.4 present the implementation of stopping the iterative Genetic algorithm.

4.4 Sorting Summary Sentences

After terminating MapReduce iterations, we have the individual with maximum fitness, which contain a collection of sentences. The last step is used to sort sentences in chronological form. Multi document summarization select sentences from multiple documents, therefore, we should order the generated summary sentences such a way the user can be easily understand the events. The sort process itself does not guarantee to improve the generated summary quality but improves the overall user comprehension and readability of a summary (Barzilay & Elhadad, 2002).

There are many strategies to sort sentences for multi document summarization the first is majority ordering, which order the sentences according to is order at the original text. The second is chronological ordering, which order the sentences of the summary according to it is publish date. Another strategy used in feature extracting system used to order sentences based on its rank (Sripada, Kasturi, & Parai, 2005). They put the high ranked sentence in the first and so on. In the proposed approach, we sort the sentences according to published data in ascending order to produce the final summary output.

4.5 Summary

In this chapter, we presented an approach for making text summarization for large-scale multi Arabic documents using Genetic algorithm and MapReduce. We use MapReduce model for parallelizing the execution of Genetic algorithm. We performed many text pre-processing tasks, which is required for computing important features for evaluating the Genetic algorithm individuals in parallel. In the next chapter, we show the detail of implementing the proposed approach on Hadoop, MapReduce system, more information about the set of experiments and the setup of system environment.

Chapter 5
Implementation and Experiments

At this point, we have designed the system for performing automatic text summarization for multiple Arabic documents using parallel execution of Genetic algorithm. In this chapter, we show the details of implementing the system using specific tools and software. The dataset is collected from many different online databases that run on MySQL and this requires transforming the news items into simple text format. This step prepares the text for processing using MapReduce that may affect the quality of the text and the final summary output. The datasets are stored in MySQL table contain a collection of news of deferent categories. The first step we need to extract the data to the HDSF and process it. In addition, we show how Genetic algorithm can be implemented as iterative MapReduce jobs. Finally, we present the environment setting of the proposed approach, the corpus details and sets of experiments.

5.1 Data Pre-Processing

Before starting the pre-processing phase, we need to store the news items in a single file on HDFS. This is an essential process because HDFS use the concept of blocks for storing the data. Each HDFS block uses the default 128 MB size. If the file is size larger than the default size, the file system breaks the file into chunks and replicates them on different nodes. In our case, we deal with text files in few kilobytes, this enables the system to reserve a space of 128 MB for every news file, which is not efficient for disk space management, and requires a Map phase for each file.

We suggest an efficient mechanism to overcome the above limitation. We store all news belonging to a category in a single file on HDFS. This enables the file system to deal with a maximum of 10 categories, which can be represented as single HDFS file and make the file system divide it into the available nodes. This enables the Map phase deal locally with the input file parts available on the machine that runs it. This decreases the execution time and the network overhead to read data from other nodes.

A MapReduce job is used to read the content of every category file in parallel. When starting a new job for reading the content, there are two basic methods. The first is Map, which invoked by the master program driver for reading a single line of the input file every time it is called. Hadoop has many features for enhancing and ensuring the accurate and effective reading of data from distributed parts across the cluster nodes. This decreases the network overhead by selecting the nearest node and using local data chunks for a Map rather than reading it from another node.

5.1.1 Text Cleaning

Most data mining tasks starts by preparing data for processing. In the web area, the data always contain many irregular or many variants due to reasons like human mistakes or use of special text formatting. This may include HTML tags, special formatting tags and hash tags, which makes text readable, and make web site navigation easy. The process of text cleaning presented in Section 4.1.1 and consisted of remove special tags and characters, remove non-Arabic words, remove diacritics and normalize Arabic characters.

5.1.2 Tokenization

Multi document summarization for large-scale data starts with breaking a single news item into a set of sentences for building the summary. The tokenization process is explained in Section 4.1.2 and during this process, we should extract some two features from the text like sentence position and sentence length. These features must be computed in context of the news items because its computation depends on the other sentences of this particular news item. A type of tokenization is word tokenization, where the sentence is divided into a set of words for the stemming process. The stemming is important for weighting sentences to know how a sentence contains frequently used terms. Therefore, it is performed during the tokenization for one time only rather that doing it every time in the sentences ranking process.

This should increase the pre-processing task execution time but on the other hand decrease evaluation and the evolving operation in Genetic algorithm, which is the most time consuming.

5.1.3 Term Frequency

The proposed approach uses term frequency for ranking the sentences to build a summary. The term frequency counts the number of times a term *t* occurs in a document *d*. Another important term is the IDF, which find how many times the term t, occurs in a collection of documents. Nevertheless, in our large-scale environment, the IDF factor is not sufficient since the text stored in a single file. Therefore, we only use the term frequency feature to weight the word of every sentence as shown in Algorithm (5.1).

Algorithm (5.1): Term Frequency count.

```
Map (key, value), Reduce (word, sum)

class Mapper
        method Map(doc a in pathOfFile)
                for all terms in sentence s ∈ doc a do
                    if ( term t is not stop word)
                            Emit(term t, count 1);
                    end if
                End for
End class Mapper

class Reducer
        method Reduce(term t, counts [c1, c2, . . .])
        sum ← 0
        for all count c ∈ counts [c1, c2, . . .] do
                sum ← sum + c
                Emit(term t, count sum)
```

To do that, we need a single MapReduce job to count the occurrence of every word in the corpus. This step is required for computing the weight of every sentence in the feature extraction step as shown in the next section. We should deal with stop words at this moment because they may occur more than any other words. Stop words used in any language to connect the different part of text; therefore, we should remove it from the term weighting process. The output of this step is every stemmed word with number of its occurs.

5.1.4 Features Extraction

In the previous step, some features are extracted that require the whole news text in order to find text position and text length. As explain in Section 4.2 the remaining features can computed based on the sentence text without taking into account the other sentences except text cohesion feature. Like text pre-processing, feature extraction is a time consuming process, therefore we compute it one time only and reuse the features in every iteration of the parallel Genetic algorithm. We should notice that the process of feature extraction takes the longest time of phases this due to using POS and using the sentence readability measure for a numerous number of sentences.

In our Arabic multi document summarization for large-scale data, the number of sentences exceeded 6.5 million sentences, in order to be processed and summarized, require using a cluster of machines. In this phase, we extract the noun count per sentence feature, the verb per sentence feature and the sentence readability feature. Another important feature is weighting every sentence as the aggregate of every stems occurs from the file in the previous step. This is very important because it gives the sentences that contain very frequent terms a higher score than the sentences that contain low frequent terms.

Algorithm (5.2): Text Pre-processing and Feature Extraction.

```
Map (key, value)

Input: path of HDFS contains .txt files
Output : <sentenceID, sentenceText, SentenceStems,
sentenceFeatures(sentenceLenght, sentencePosition)>.
while pathArray.hasNext do
    file = pathArray.fetchCurrentFile
    sentences = file.splitToSentences.toArray
    For i=0 to sentences.length do
        sentence = sentenceClean(sentences[i]);
        sentenceStems = rootStem(sentence);
        sentenceFeatures = extractFeatures(sentence);
        sentenceFeatures = sentenceFeatures+ senPosition(sentences);
        sentenceFeatures = sentenceFeatures+ senLenght(sentences);
        sentenceFeatures = sentenceFeatures+ weightSentence(sentence);
        sentenceID = generateUniqueSequenseID;
    Write<sentenceID, sentence, SentenceStems, sentenceFeatures>;
    End For;
End while;
End;
```

The text pre-processing implementation step is performed using a Map phase only because there is no aggregation of the values based on a common key. However, this phase represents the core of a sentence ranking mechanism to determine the strong individuals for implementing Genetic algorithm based on the fitness function which based on the features of each sentence.

Algorithm (5.2) shows the text cleaning, tokenization and feature extraction implementation. It explained how sentence position and text length computed depending on the news item sentences. The algorithm shows extracting the main features and weight every sentence according to the number of occurs of every stem at the sentence. The output of Algorithm (5.2) is considered the final output for the text pre-processing and feature extraction phases, which prepare the text for the parallel Genetic algorithm Based on MapReduce as shown in Section 5.2.

5.1.5 Text Shuffling

Now, we get a group of files, every files contains the sentences belonging to a single category stored together with its stems and features. There is an important step before passing text to Genetic algorithm first step: population initialization. We need to reorder the sentences of every category randomly because the pre-processing step stores the sentences of every news item in a sequential order. This ensures that every individual contains sentences from many different news to achieve the diversity of multi document summarization, which means that the summary covers most of the topics of the original text.

In MapReduce, the key is used to group the values that have the same key to send them to the same Reducer. We can exploit that to redirect every sentence to a random Reducer to store them randomly. We generate a random ID for every sentence to ensure that we distribute them to a random Reducer. The Reduce phase invoked after all the Map tasks finished to store the sentences randomly.

Algorithm (5.3): Text Shuffling.

```
Map (key, value), Reduce (word, sum)

class Mapper
    method Map(doc a in pathOfFile)
        for lines ∈ doc a do
            UUID = generateRandID(line.sentence);
                Emit(UUID , line);
        End for
End class Mapper
class Reducer
    method Reduce(line)
        write(line);
```

Algorithm (5.3) explain the process of text shuffling which crucial for creating individuals as explained in Section 5.2.1. It read a line through the Map method and generate random ID number to send this line to random Reducer. This step makes the Reduce receive unordered sentences to make every individual consisted of sentence from different news items to achieve the diversity option in the summary.

At this stage, the text is prepared by cleaning it and segmenting it into a group of sentences and get the stem of every sentence. In addition, we calculate all the required features, which is the basic operation for evaluating our individual's fitness function. The next section considered as the core of parallel Genetic algorithm to make text summarization for large-scale text. It shows how individuals processed, evaluated and evolved used iterative MapReduce jobs.

5.2 Parallel Genetic Algorithm

We can Genetic algorithm over MapReduce model to exploit the powerful of parallel computing. The high computation needed by Genetic algorithm especially for finding the fitness of large-scale individuals, makes using MapReduce more demanding. This should divide the high processing time across a cluster of machines that processes parts of text files separately.

5.2.1 Creating Individuals

Often, Genetic algorithms start with creating the initial search space of the potential solution or in other words the search space. The traditional proposed multi document summarization uses the compression ration concept, which controls the percentage of the produced summary to the original data. However, in large-scale data this may not be time and space efficient because the size of the resulting text summary is considering large. Another type of compression is uses a fixed length of the summary regardless of the original text size. Therefore, this is suitable for large-scale text summarization.

In the proposed approach, we prefer to use a fixed summary length in order to simplify the adaptation of Genetic algorithm in MapReduce model. As mentioned in Section 5.1.5 randomly rearrange sentences enables the summary to cover most of the topics of the original text as possible much as. The next step is to sequentially create the individuals or chromosomes, which is a basic data structure of Genetic algorithm as explained in Algorithm (5.4) using single Map job only.

Algorithm (5.4): Create Individuals.

```
Map (key, value)

class Mapper
summaryLength;
individual;
processed ← 0;
    method Map{
        while (processed < summaryLength) {
            individual ← individual+ currentLine;
            processed++;
        }
        write individual;
        reset(individual);
    }
End class Mapper
```

As show in Algorithm (5.4), it reads a single sentence which represented by a single line at one time. The desired summary (individual) length of individual used as parameter, and loop until processing number of lines equal to the required individual length. Finally save the current individual to start creating new individual. Every single individual is considered as a possible solution in the search space for Genetic algorithm and initially resulting individuals should have a fixed length to perform the evolving process of individuals efficiently. When all sentences are distributed to individuals, all the individuals are stored in a single file on HDFS to make it easy for the next processing phases.

5.2.2 Scoring Individuals

Genetic algorithm depends on the principle of survival of the fitness, which means that the strong individual can still survived and produce strong children. To use the principles of Genetic algorithm in automatic text summarization, we rank the individuals based on the output of Section 5.1 to know the best individual or other words the best summary. As mentioned before finding the fitness of every individual consumes most time in Genetic algorithm, therefore, we implement this task in

MapReduce in order to divide the computation across a cluster of machines and therefore, reduce the overall needed time.

Individuals consist of a group of sentences. Every sentence is stored with its extracted features (Section 4.2 and Section 5.1) and these are the basic measurements of the fitness of every individual. We can make an optimization for automatic text summarization, which decreases the computing time of the fitness of every individual in every iteration by storing the aggregate of all features with every sentence.

We mention in Section 5.1 that the content of individual may change during the evolving task in crossover operation. Therefore, we find the text cohesion feature of individuals in every iteration. We perform scoring individuals in a Map task only by reading an individual that represents a single line in an HDFS file based on the format shown in Table (5.1).

Algorithm (5.5): Scoring Individuals.

```
Map (key, value)

class Mapper
Fitness ← 0;
individual;
method Map {
    individual ← parseIndividual(currentLine);
    textCohesion = calculateCohesion(individual);
    individual.updateFitness(Fitness);
    Emit(key, individual);
}
End Class Map
```

Table (5.1): Individual Format.

Individual$_1$	Sentence$_1$<feature$_1$, feature$_2$, …>,Sentence$_2$ <feature$_1$, feature$_2$, …>, …
Individual$_2$	Sentence$_2$<feature$_1$, feature$_2$, …>,Sentence$_2$ <feature$_1$, feature$_2$, …>, …

Every line represents a potential summary, therefore we should rank every individual based on its features and this is performed in Algorithm (5.5). The algorithm is parsing on line every time and compute the text cohesion feature. It updates the new fitness of the individuals to use it in the iterative MapReduce job, which presented in

Section 5.2.3. The process of scoring individuals is performed once using single MapReduce job and output one file for every category.

Table (5.2): Individual Final Format.

Individual$_1$	Sentence$_1$<featuresSum>, Sentence$_2$< featuresSum >, …
Individual$_2$	Sentence$_1$<featuresSum>, Sentence$_2$< featuresSum >, …

The final format of every individual stores the aggregate sum of all features as the format in shown in Table (5.2). We separate the cohesion features from other features because they are computed at every iteration due to the change in individual sentences during the crossover operation. This above is performed done using a single Map operation only once to find the fitness of every individual in parallel. The evolving of population requires using iterative nature of Genetic algorithm using several MapReduce jobs.

5.2.3 Evolving Population

The evolving of Genetic algorithm population encapsulated using MapReduce is done in iterative nature. At every iteration, many operations are used to generate new set of individuals from the current individuals. Survival of the fittest is the main principle in Genetic algorithm to pass characteristics of the strong individuals to the new generation using genetic operations. Every iteration uses Map phase and Reduce phase to read individuals from the HDFS and evaluate them in parallel to decide starting new iteration or stop Genetic algorithm.

Algorithm (5.6): Map Phase of Each Iteration of GA.

```
Map (key, value)

Class Mapper
Method Map{
    individual ← parseIndividual(line);
    emit(key, individual);
}
End Class Mapper
```

At every iteration, a tournament is conducted between fixed numbers of individual to select the fittest individuals to perform uniform crossover operation. At the first iteration, the Map method simply reads the current line from the HDFS and sends it to the Reducer as shown in Algorithm (5.6). The main Genetic algorithm operations are performed using selection and crossover at the Reduce phase of every iteration.

As explained in Algorithm (5.7), the Reducer performs the core operations of Genetic algorithm. The tournament selection is performed using a predefined number of selection pool to specify how many individuals participated in a single tournament. Only one individual survives and is passed to the crossover, therefore the failed individuals are stored back to the HDFS to have another chance at another tournament. In addition to that, the new individual's fitness is calculated at the end of Reduce phase before storing the individuals to use them at the next iteration if it is conducted.

There are many important issues solved in method CrossoverAndSelection at every Reduce phase for all iterations. The first is handling the redundancy. In data mining tasks, we can represent any document as a vector of terms to know the degree of similarity of each term to the other terms in the document. Cosine similarity is a popular method the measure the similarity of any two vectors. We can use a similarity matrix constructed from all sentences of every individual to find the similarity of each sentence to other sentences and remove the sentence with lowest fitness.

Every matrix entry stores the cosine similarity value for the similarity of a sentence from a column and a sentence from a row. In the method CrossoverAndSelection that implemented in Algorithm (5.7), we can consider a sentence as redundant if its similarity with other sentences is greater than a threshold using the cosine similarity equation which presented in Section 4.2.6.

In multi document summarization the redundancy is a common issue, therefore, we can replace the duplicated sentence with another sentence from the corpus. Nevertheless, dealing with redundancy for large-scale data makes it a critical issue. We cannot select a sentence every time we find a redundant sentence in an individual. This requires searching in a file containing more than a million of lines and this increase the time of execution. We propose removing the duplicated sentence that have the least fitness from the individual this minimize the execution time, network overhead and do not affect the overall fitness of the individual. At every iteration, we should find the text cohesion and the new fitness of all individuals after the crossover operation due to the change in the content of the individual during the crossover.

Algorithm (5.7): Reduce Phase of Each Iteration of GA.

```
Reduce (key, values)

Class Reducer
textCohesion ← 0;
maxIndividual;
tournamentSize;
processedInd ← 0;
tournamentArray;
Method Reduce{
    While (values.hasNext){
        individual ← parseIndividual(values.next);
          If processedInd < tournamentSize
                textCohesion ← calcTextCohesion(individual);
                individual.updateFitness();
               if individual.fitness() > MaxIndividual.fitness()
                    maxIndividual ← individual;
              end if;
                tournamentArray.add(individual);
         else
               CrossoverAndSelection(tournamentArray);
        End if ;
        processedInd ← processedInd + 1;
       if(tournamentArray.size() == tournamentSize )
            tournamentArray.clear();
       end if;
   }
   writeMaxIndividual(individual);
    writeNotWinIndividuals();
}
```

At every iteration, we should find the individuals with maximum fitness to monitor the evolution of the population. Every Reducer should save the individuals with maximum fitness to a separate file from other individuals from a file on HDFS where all Reducers of this iteration do the same. This makes a decision when to start new iteration or stop the iterations as shown in the next section. Algorithm (5.7) shows how the Reducer reads all lines from the Map and how conduct a tournament between fixed numbers of individuals. In addition to that the process of saving the max fitness individual to separate file and write the failed individuals at the end of the algorithm back to give them another chance.

5.2.4 Stopping Genetic Algorithm

As shown in Section 2.3.1, there are many criteria for stopping Genetic algorithm iterations. Our proposed approach is concerned with improving the text cohesion score and the readability score of the generated summaries to maximize the quality.

The used Genetic algorithm makes 3 to 5 iteration initially and let stop the iteration to the user based on the result. The proposed approach makes 3 to 5 iterations and monitor the maximum fitness at the end of every iteration. If it is more than 10% of the previous iteration, we start a new iteration. This means that the fitness of the individuals improves incrementally and there is a chance to get improved fitness at the next iteration.

It is worth mentioning as explained in Algorithm (5.8), that this step controls the flow of Genetic algorithm iteration and decides to start a new iteration or not. Therefore, we should accurately monitor how the individuals' fitness changes during MapReduce iterations. In order to make infinite iterations if the individual improvements not reach the desired ratio (10%), we add a constraint that makes only specific number of MapReduce jobs iterations. This performs a fixed number of MapReduce iterations if the job reaches them; the iterative MapReduce is terminated even if the desired ratio is not reached.

Algorithm (5.8): Genetic Algorithm Stop Criteria.

Algorithm 5.8: Stopping Genetic Algorithm Iterations

```
Class StopGenetic
    previousFitness ← read max Individual fitness of previous iteration;
    currentFitness ← read max Individual fitness of current iteration;
    if (currentFitness > previousFitness + previousFitness * 10% or MaxIterations)
        startNewIteration;
    else
        stopIteration;
  end if; end class StopGenetic;
```

5.3 Experiments

To implement the approach for multi document summarization for large-scale Arabic documents, we use Java programming language with various libraries to making text pre-processing and features extraction more easy. Another important reason for choosing Java is that Hadoop (which open source MapReduce realization) is implemented and provides Application Programming Interface (API) using Java. In this section, we talk about the tools and libraries used in the proposed approach. The environment setup and the set of experiments conducted to realize and test the proposed approach.

5.3.1 Tools and Environment Setup

There are many frameworks and tools used in processing and executing the proposed approach especially in text pre-processing and features extracting phases. This section describes briefly the configuration and the software used for the experiment.

5.3.1.1 AraNLP Library

AraNLP (Althobaiti, Kruschwitz, & Poesio, 2014) is a free Java library containing tools for processing Arabic text. AraNLP groups many useful tasks in one toolkit, which makes it easy to integrate with existing programs or text processing packages. In general, AraNLP toolkit contains sentence tokenization, root and light Arabic text stemming, Arabic part of speech tagger (POS), word segmenter, normalizer, and a punctuation and diacritic remover.

5.3.1.2 Osman Readability Measure

As described in Section 2.2.5, our proposed approach uses a readability for scoring Arabic sentences. We use a state-of-the-art Java based library Osman (El-Haj & Rayson, 2016) for giving a numeric values to each sentence in the corpus that measures how this sentence is easy to read. This library can be used for scoring any form of Arabic text with diacritic or plain text.

5.3.1.3 Hadoop Cluster

In order to process large amount of data, there are many approaches and frameworks. Hadoop is a MapReduce model realization for processing large amount of data using a cluster of commodity hardware. We build a cluster of 17 machines that run based on Ubuntu 15.4 and a 2.6.0 Hadoop version that is running with 2 GB of Ram and 2 Core Intel Xeon(R) CPU at 3.10GHz. There is a master machine which acts as name node as described in Section 2.5. The other 16 machines act as data nodes and the cluster is connected using high-speed local switch for guarantee the high speed of communication between machines.

By default, HDFS replicates every file up to three replicas on different locations and every block size is 128 MB, which means that every chuck will reserve a constant size. This help Mapper to read the part of data locally rather than read from another machine.

5.3.2 Corpus

The proposed text summarization approach works with for large scale multiple Arabic documents, therefore, we have gathered data from many online newspapers. We have selected news from five famous Palestinian websites. These sites include Palestine Today, Paltimes, Sama News, Safa News and Alresala News. The information of each website in terms of the number of news items and the size of these items in MB are listed in Table (5.3).

The text is stored for web pages; therefore, they contain HTML tags to format the data, which require more processing time and effort due to the diversity of tags and the use of some customized tags in each webpage.

Table (5.3): Corpus Details.

Website	Number of News Items	Size in MB
Paltoday	219948	644
Samanews	207249	711
Safanew	165116	410
Alresala	131583	329
Paltimes	109990	262
Total	833886	2356

The previous table describes the original dataset before any modification or pre-processing being applied. Nevertheless, our approach uses the sentence as the basic unit in the summary. Therefore, we need to split every news item into a set of sentences and clean it from HTML tags by applying text pre-processing techniques and extract the features of every sentence for further use. Table (5.4) shows every category after pre-processing and extracting the features of every sentence of the corpus and categorizing every news item to a single category. We conduct many MapReduce jobs to apply the proposed approach on the data sets in Table (5.3) and measuring the change of individual fitness during every iteration of Genetic algorithm Section 4.3.

Table (5.4) shows that the number of news items is decreased after the pre-processing step because many news sentences are removed due to poor readability score, very long sentence, very short sentence or duplicate sentences. The removed sentences may affect the quality of the produced summary.

Table (5.4): Category Details.

Category	No. of News Items	No. of Sentences	Size in MB
Reports	241572	1968791	1648.64 MB
Politics	149044	1181291	744 MB
Local	129738	903702	683 MB
International	94972	804224	546 MB
Sport	33626	205894	140 MB
Culture	12587	99637	72.7 MB
Economy	7698	52645	37.5 MB
Technology	6415	44374	30.9 MB
Health	5534	45359	30.9 MB
Total	681186	5305917	3933.64 MB

5.3.3 Partitioning Individuals

The summarization starts with creating the potential individuals. As mentioned in Section 4.3, every individual consists of a group of sentences chosen randomly from each category. The compression ratio is a common concept in automatic text summarization models, it is a value that represent the generated summary of the original document/s size. Nevertheless, in our approach, the data volumes are large and consequently the summary size is large if we follow the mean of the traditional compression ratio.

Table (5.5): Category Individuals Size.

Category	Individual Size (in sentence)	No. of Individuals
Reports	600	3281
Politics	600	1968
Local	500	1807
International	500	1608
Sport	500	411
Culture	400	249
Economy	400	131
Technology	300	147
Health	300	151

We decide to overcome this option and make every individual have a specific size and all individuals initially should have the same size, which required for crossover operation. Creating individuals is performed after the process of shuffling the category sentences as explained in Section 5.1.5. The details of creating individuals shown in Table (5.5).

Table (5.6): Creating Individual Execution Time.

Dataset	Size	Execution Time in second
Reports	1648.64 MB	841
Politics	744 MB	484
Local	683 MB	447
International	546 MB	282
Sport	140 MB	47
Culture	72.7 MB	34
Economy	37.5 MB	19
Technology	30.9 MB	16
Health	30.9 MB	16

The process of creating individuals starts with ordering the sentences in random fashion, sequentially select group sentences with the individual size, and give unique identity number to this group as individual. Table (5.6) shows the execution time of breaking the news items of each category into a collection of individuals. Every category individuals are stored into a separate file on HDFS to be passed later to MapReduce based Genetic algorithm iterations applied to the created individuals.

5.3.4 Parallel Genetic Algorithm

The output of partitioning individuals creates a HDFS file for every dataset where every line in the file contains a single individual. The process of parallel Genetic algorithm contains the core operations of Genetic algorithm as MapReduce model realized by Hadoop. MapReduce starts a job by reading the input file and distributes its content across cluster nodes. HDFS divides the input file and sends all file chunks to the nodes. This makes the Mapper of every node read from the local copy rather than read from other nodes to utilize the network bandwidth and decrease the required network overhead.

In general, MapReduce prefer using small number of large files rather than using large number of small files. We decide to eliminate small datasets from execution by using MapReduce job and make the summarization for files sizes of 300 MB and beyond.

Table (5.7): Summarization Execution Time.

Dataset	Execution Time In Seconds					
	1 node	2 nodes	4 nodes	8 nodes	12 nodes	16 nodes
Reports	1463	1219	947	621	328	148
Politics	868	723	640	470	273	121
Local	750	641	597	430	261	119
International	523	486	355	294	249	117

At every iteration, the Mapper reads one line from HDFS and send it to the Reducer. The Reducer reads individuals in pairs to run the crossover operation in parallel and compute the new fitness of the individual after the crossover. The execution time of MapReduce iterations for all iterations is described in Table (5.7) and is illustrated on Figure (5.1).

The results in Table (5.7) show that as the size of category increased, the execution of the proposed approach become better that is execution of small size categories, this is because of MapReduce prefer processing small number of large files rather that processing large number of small files. We should notice that the results in Table (5.7) considered as the basic input for measuring the speedup, efficiency and scalability in Section 6.2, Section 6.3 and Section 6.4.

Each iteration produces two HDFS files, the first contains the whole individuals stored with the fitness and the second is a file contains the individuals with maximum fitness of every Reducer. At the end of every iteration, the file contains the maximum individuals compared with the output of the previous iteration and decides starting new iteration or no based on the improvement ration of maximum fitness of each iteration as explained in Section 5.2.4.

The output of every iteration considered as the input of the next MapReduce iteration. Finally, if the MapReduce iterations stop successfully, we read an individual from the small file from the output of the last iteration that contains the individuals that have the maximum fitness as the output summary. As described in Section 4.4, the sentences of the summary are ordered in chronological order based on the publish date of the original news.

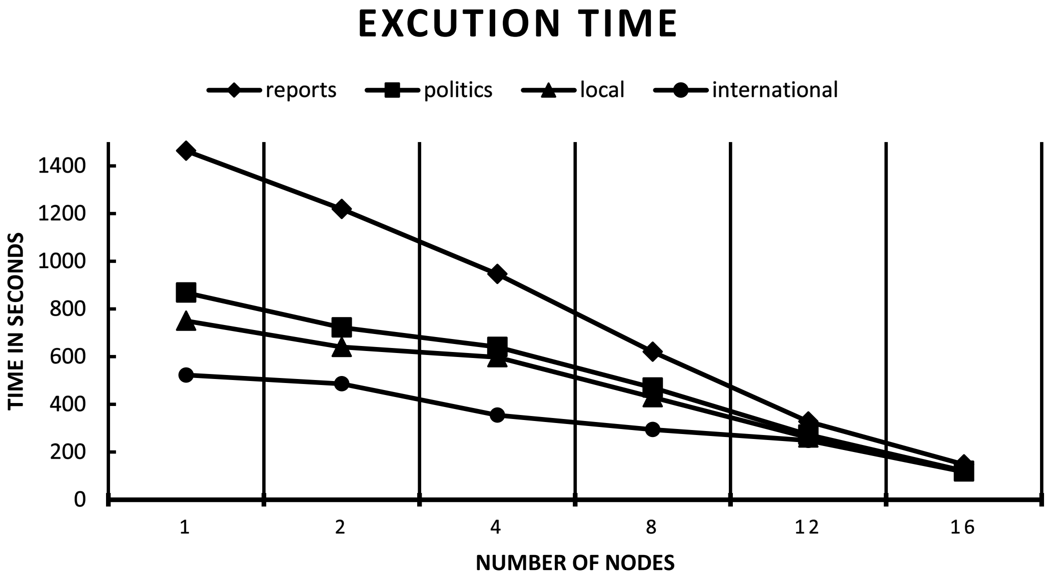

Figure (5.1): Summarization Execution Time.

We should notice that the results in Table (5.7) considers as the basic for computing the speed up score for the proposed approach for text summarization as explained later in Section 6.2.

5.4 Summary

In this Chapter, we described the details of the implementation of our proposed approach for making automatic text summarization of large-scale multi Arabic documents using MapReduce and Genetic algorithm. We give short description about the collected data and its characteristics and the NLP tools used in the implementation. In addition to that, we presented the environments of Hadoop cluster and the nodes, which runs MapReduce model. Finally, we presented the set of experiments of the proposed approach.

Chapter 6

Evaluation

In this chapter, we present an evaluation for the proposed summarization approach that consists of measuring the quality of the generated summaries in terms of precision and recall and measuring the parallel computation speedup, efficiency and scalability.

6.1 Summarization Quality

There are an important measure from measuring the quality of document summarization, Recall-Oriented Understudy for Gisting Evaluation (ROUGE) proposed by (Lin, 2005). ROUGE uses peer review or human summary to compare the system-generated summary with the human summary using several sub measures like (ROUGE-N, ROUGE-L, and ROUGE-W). One disadvantages with ROUGE is it inability to deal with large summarization approach, it suitable for small datasets where human effort is involved in the summary of the original text and comparing it to the system summary.

Since the proposed multi-document summarization approach deals with large-scale Arabic text documents, ROUGE measure would be ineffective and time consuming due to the need to perform manual summarization of the whole data by a human. Therefore, using the ROUGE measurement in our proposed approach is a challenging task.

Additional important measures for measuring the quality of document summarization are based on precision, recall and F-measure (Gong & Liu, 2001). Precision is the fraction of retrieved documents that are relevant to the query.

$$\text{Precision (P)}=\frac{|\text{system}-\text{human choice overlap}|}{|\text{sentences chosen by system}|} \qquad \text{Equation 6.1}$$

While recall is the fraction of the documents that are relevant to the query that successfully retrieved (Nenkova, 2006).

$$\text{Recall (R)}=\frac{|\text{system}-\text{human choice overlap}|}{|\text{sentences chosen by human}|} \qquad \text{Equation 6.2}$$

F-measure = $\frac{2*P*R}{P*R}$ Equation 6.3

Therefore, we propose selecting small and random samples of the produced summaries and then preform manual summarization for them. This procedure follows: select random and small samples from the system-generated summaries, fetch the original text of news items of these summaries sample sets and preform the traditional precision, and recall measurements to assess the quality of the summary. The same samples are provided to a three human experts in managing news website to make summary of sample consisted of 10 news items for each data set which mean 40 news item in aggregate. The human experts asked to select the most important sentences from the sample original news items, which can represent the whole sentences regardless the number of chosen sentences.

The results of the three human expert summarization are shown in Tables (6.1), (6.2), and (6.3) respectively.

Table (6.1): System Results vs. Human Expert 1 Results

	International	Local	Politics	Reports
System Summary	12	11	13	10
Expert 1 Summary	26	24	26	23
Correct sentences	4	5	3	4
Precision	0.33	0.45	0.23	0.4
Recall	0.15	0.21	0.12	0.17
F-measure	0.21	0.29	0.16	0.24

Table (6.2): System Results vs. Human Expert 2 Results

	International	Local	Politics	Reports
System Summary	12	11	13	10
Expert 2 Summary	28	30	34	33
Correct sentences	6	7	10	7
Precision	0.5	0.64	0.77	0.7
Recall	0.21	0.23	0.29	0.21
F-measure	0.3	0.34	0.42	0.32

Table (6.3): System Results vs. Human Expert 3 Results

	International	Local	Politics	Reports
System Summary	12	11	13	10
Expert 3 Summary	25	27	28	23
Correct sentences	9	7	8	6
Precision	0.75	0.64	0.62	0.6
Recall	0.36	0.26	0.29	0.26
F-measure	0.49	0.37	0.4	0.36

In document summarization approaches, the precision score represents the correct summarized sentences from a collection of sentences. The results show that there is variance in the specialists' summarization. The first human expert results show that low score for precision, recall and f-measure. The results of the second and the third experts were very similar to each other. The main reason of this variance in the results is that the human experts summarized the news items based on their vision of the importance of each sentences in the news item.

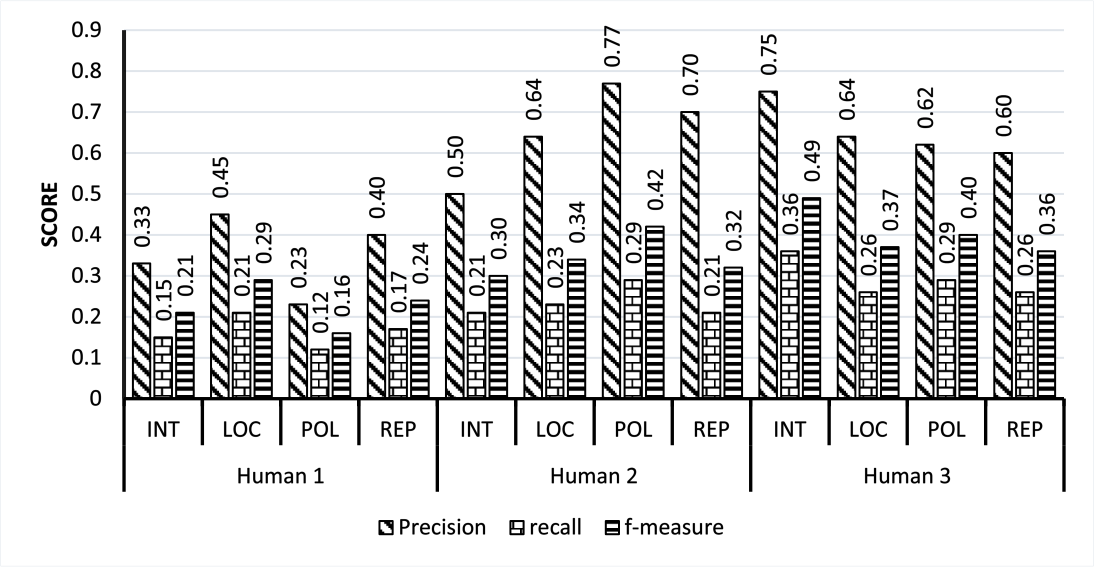

Figure (6.1): Summarization Quality Based On the Selected Sample Text.

Figure (6.1) shows the results of summarizing the datasets for the three human summaries of all datasets. The results show highest precision scores: local data set by expert 1, the politics data set by expert 2. While international data set, get the highest score by expert 3. The figure also shows the lowest precision score especially in summarizing the politics data set by human expert 1. Figure (6.2) illustrates the average of the precision, recall and f-measure for the four datasets using the three human experts. The results show that the local data set has the maximum precision score while the reports dataset has the lowest recall score.

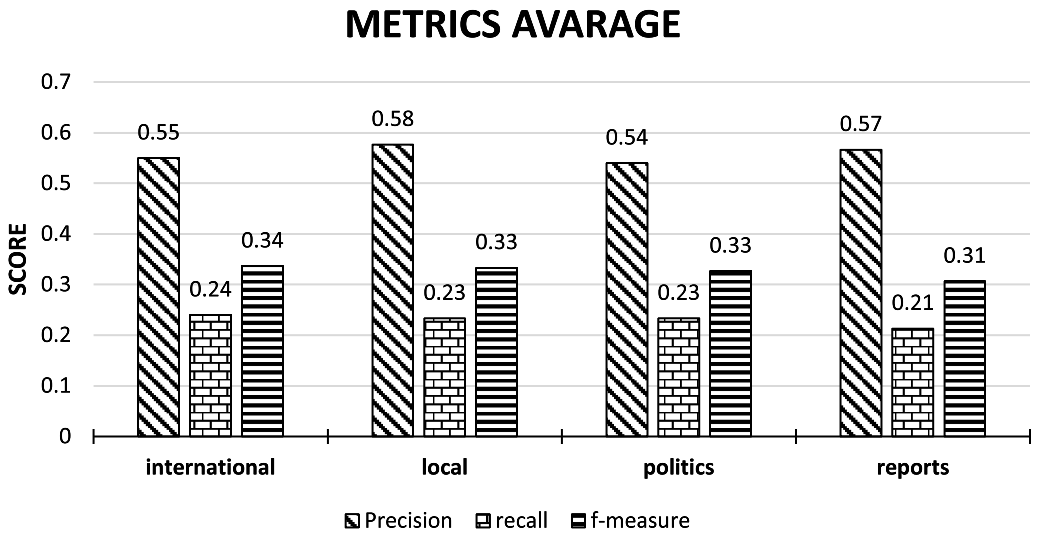

Figure (6.2): Evaluation Metrics Average.

6.2 Speedup

Speedup is an important measure for the performance of parallel computations. Speedup (***S***) is the ratio of required taken time to solve a problem using single processing (t_s) to the required time to solve the same problem using several parallel (t_p) computers (Grama, 2003). Measuring the speedup metric is important in our research; firstly, should run the same MapReduce jobs on different node start from 2 nodes to 16 nodes of our cluster as described in Section 5.3 and compare the results to the ideal speedup. The speedup measured using the following equation:

$$S_{n=} t_s/t_p \qquad \text{Equation 6.4}$$

where ts (time of serial) is the time of execution the multi document summarizer on one node, while tp (time of parallel) is the time of execution the same program using parallel computers.

The results in Table (6.4) show that the speedup of the summarization for the datasets is not close enough to the ideal speedup when using less than 12 computer nodes. On the other hand, when we use 16 nodes, the speedup gets more close to the ideal speedup. There are many reasons that prevent reaching a speedup close enough to the ideal speedup. In parallel computations systems, there are a lot of time lost during executing programs in parallel fashion like communication time, idle and waste computation.

Table (6.4): Summarization Speedup.

Dataset	Speedup (S)				
	2 nodes	4 nodes	8 nodes	12 nodes	16 nodes
Reports	1.200164	1.544879	2.355878	4.460366	9.885135135
Politics	1.200553	1.35625	1.846809	3.179487	7.173553719
Local	1.170047	1.256281	1.744186	2.873563	6.302521008
International	1.076132	1.473239	1.778912	2.100402	4.47008547

In addition to that, MapReduce runs Map and Reduce jobs in parallel but, Reduce method should wait all Map methods on all nodes to complete in order for it to start and therefore can increase the overall execution time. Another important reason is that MapReduce trends to process large input files rather than process small files.

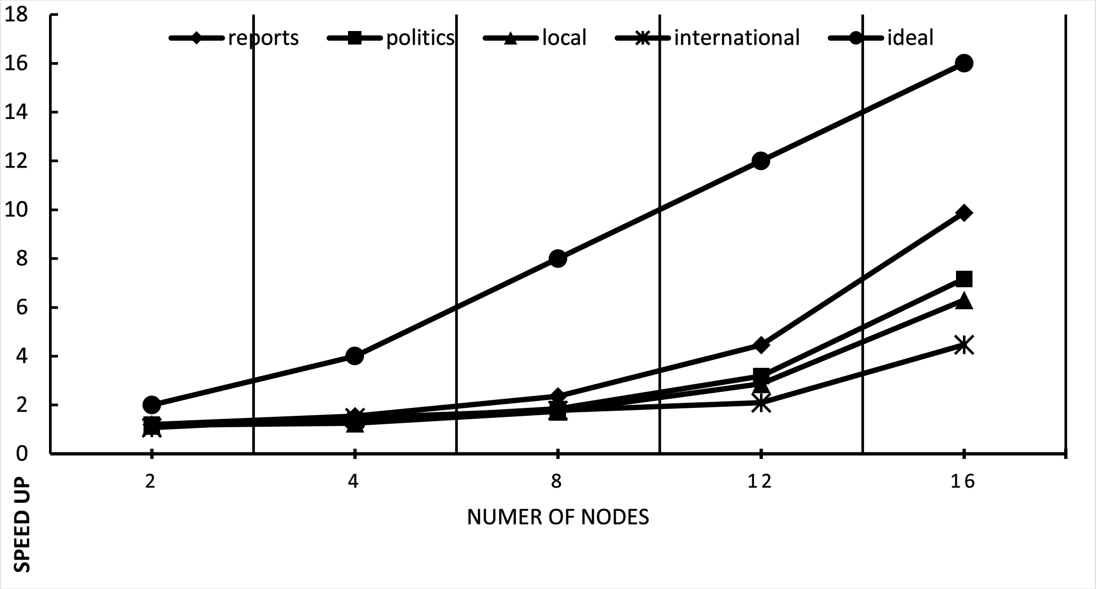

Figure (6.3): Summarization Speedup.

In our case, the size of the largest dataset is about 1.2 GB, which is considered small for ideal processing by MapReduce, which prefer large amount of data to exploit its advantages. The results Figure (6.3) show that using small datasets, which have size less than 1 GB, not result in high speedup score. While Reports dataset get more improvement in the speedup score when using 8, 12 and 16 nodes respectively because it is the largest data set with size more than 1.2 GB. However, the other data sets size is less than 1 GB therefore they did not result in high speedup score like the Reports data set.

The results illustrated in Figure (6.3) are show that using more dataset size and more processing nodes make our summarizer approach more efficient and the speedup getting more linearly when adding more resources to the cluster nodes.

6.3 Efficiency

In parallel systems, there are many available resources available for solving computation problems. Efficiency is measure how the available resources in parallel system is utilized (Grama, 2003). Efficiency can be computed as the speed up of the parallel system to the number of processing units. In general, the value of efficiency is between zero and one, therefore the best efficient and ideal parallel system have value one and the worse have zero. We can compute the efficiency using the following equation:

$E = S / P$ Equation 6.5

where S is the speed up of the system and P is number of processing units.

Table (6.5): Cluster Efficiency

Dataset	Efficiency (E)				
	2 nodes	4 nodes	8 nodes	12 nodes	16 nodes
Reports	0.6	0.386	0.294	0.372	0.618
Politics	0.6	0.339	0.231	0.265	0.448
Local	0.585	0.314	0.218	0.239	0.394
International	0.538	0.368	0.222	0.175	0.279

For our proposed approach, we can find the efficiency from speed up values in Table (6.4) where the processing units is the number of cluster nodes and the speed up is the corresponding speed up for category and cluster nodes as shown in Table (6.5).

Figure (6.4) shows that the efficiency of the proposed system increased for summarizing the reports dataset. The main reason is MapReduce model require larger data sets sizes which not exists in our data sets. The reports data set size is about 1.2 GB therefore it gets the highest efficiency among other data sets when using 16 nodes in the cluster. In addition to that, when using 2 nodes all data sets efficiency is around %62. Nevertheless, when using more resources, the efficiency is decreased which mean the resources is not utilized. The main reason of low efficiency score is the data sets except reports data sets size not large enough for MapReduce and it should not processed using more than two nodes.

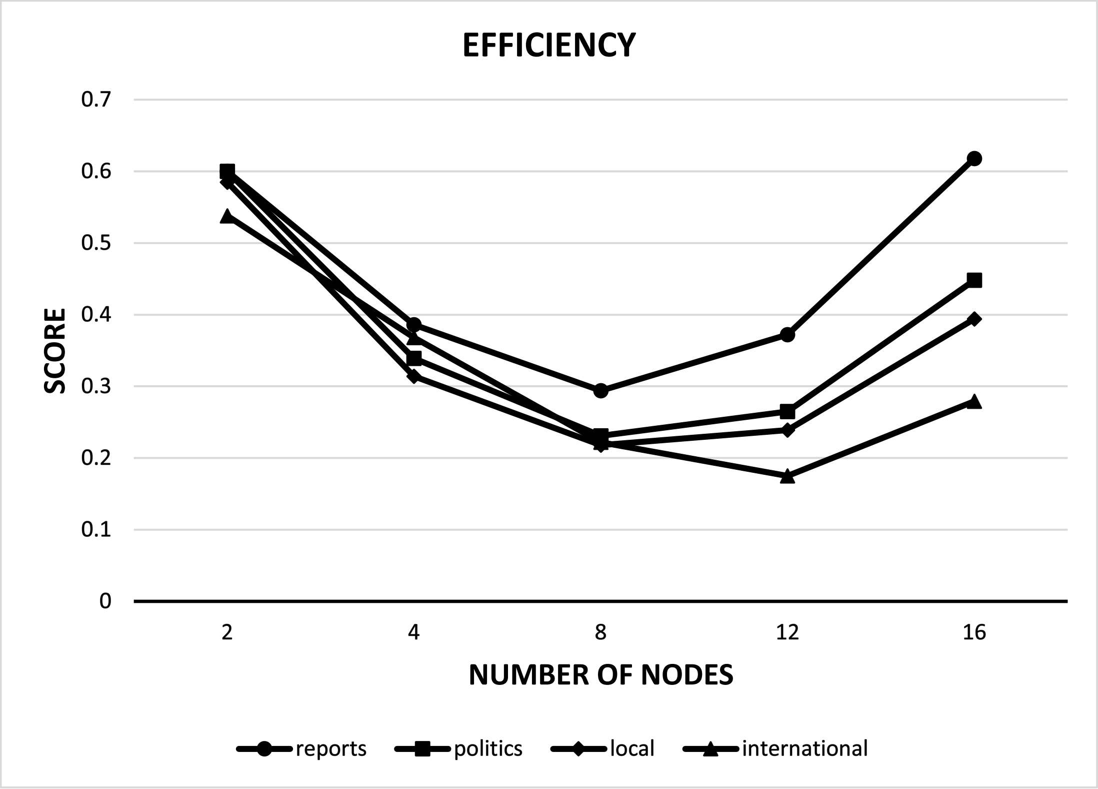

Figure (6.4): Cluster Efficiency.

6.4 Scalability

Scalability is evaluated rather than computed. A parallel system is said to be scalable when the efficiency can be kept constant as the number of processing units increased, provided that the problem size is increased (Grama, 2003). We can conclude that the proposed system is scalable, where the efficiency is kept (near to) constant while adding more nodes (up to 16 in the experiment) to the MapReduce cluster and at the same time increasing the size of data set.

6.5 Summary

In this Chapter, we presented the evaluation of the parallel Genetic algorithm using MapReduce model. The main result is that the system can correctly select the most important sentences from the datasets to construct the readable summary. The quality of the summaries is presented in terms of precision, recall and f-measure scores. The approach results into acceptable speedup and efficiency scores. We show that increasing the datasets size can improve the speedup and efficiency scores significantly.

Chapter 7
Conclusion and Future Work

7.1 Conclusion

In this research, we built a new approach for making automatic text summarization for large-scale multi Arabic documents using Genetic algorithm and MapReduce model. We choose Genetic algorithm technique because its ability to automatically evolve to the potential summaries during its iterations. However, Genetic algorithm suffers from major a problem, the high computation requirement. We used MapReduce model for the automatic distribution of a Genetic algorithm-based summarization computation across a set of machines connected together.

Our proposed approach guarantees acceptable quality of produced summaries, speedup score for using Genetic algorithm, efficient computation usage and provide high scalability. Therefore, it is adaptable to make summarization of any desired text documents sizes if there are enough resources for MapReduce.

We evaluated the proposed approach using the traditional text summarization metrics, which are precision and recall. The results show that the precision score indicated that the approach successfully identified most of the correct summary sentences similar to the human specialists' summaries. In addition to that, the proposed approach provides up to 10x speedup score, which is faster than executing the same code on single machine. Therefore, it can deal with large-scale datasets successfully. Finally, the efficiency score of the proposed approach indicates that the largest data set utilize the available resources up 62% which is a satisfying result taking into account the available data set sizes.

It is worth noting that we tackle some obstacles during this research. One obstacle is the lack of large organized Arabic text corpus. This make collecting these resources from online newspaper resources a challenging task. In addition, in Arabic NLP area there are no accurate tools for processing Arabic text such as accurate sentence tokenizers, automatic cue-words identifiers, accurate named entity recognizers and absence of a complete Arabic WordNet equivalent, which is required for semantic similarity for intelligent detection of redundant sentences.

7.2 Future Work

Various future efforts may improve the proposed approach. One such effort is to add more features to the list of used features like identifying the named entity in the text, identifying cue-words, which can be used for giving strong meaning for the text like the words: “emphasize”, “incidentally”, “for example” and many other cue words. Additional feature is using semantic similarity measures for accurately detecting the redundant sentences and the degree of similarity between words. These features are likely to increase the score of the sentences so these sentences are likely to be included in the summary, and therefore, the quality of produced summary will be improved significantly.

Finally, increasing the size, domain-diversity and sources of the dataset while performing the experiments on a larger MapReduce cluster would increase the speedup score, respectively the efficiency and scalability of the proposed approach and therefore measure the adaptability of text summarization on cloud computing. We test our proposed approach for making text summarization for other domains and other text resources like books and social network blogs and tweets.

References

Abdallah, S., Shaalan, K., & Shoaib, M. (2012, March 11-17). *Integrating rule-based system with classification for Arabic named entity recognition*. Paper presented at the 13th International Conference on Computational Linguistics and Intelligent Text Processing, New Delhi, India.

Al-Barhamtoshy, H., & Al-Jideebi, W. (2009, December). *Designing and Implementing Arabic WordNet Semantic-Based*. Paper presented at the the 9th Conference on Language Engineering.

Al-Dawsari, M. (2004). *The assessment of readability books content (boys-girls) of the first grade of intermediate school according to readability standards* (Technical Report). Sultan Qaboos University, Muscat.

Al-Hashemi, R. (2010). Text Summarization Extraction System (TSES) Using Extracted Keywords. *International Arab Journal of e -Technology*, *1*(4), 164-168.

Al-Maimani, M. R., Naamany, A. A., & Bakar, A. Z. A. (2011, February 19-22). *Arabic information retrieval: techniques, tools and challenges*. Paper presented at the GCC Conference and Exhibition (GCC) IEEE, Dubai, United Arab Emirates.

Al-Shalabi, R., Kanaan, G., Jaam, J. M., Hasnah, A., & Hilat, E. (2004, April 19-23). *Stop-word removal algorithm for Arabic language*. Paper presented at the Proceedings of 1st International Conference on Information & Communication Technologies, Syrian.

Althobaiti, M., Kruschwitz, U., & Poesio, M. (2014, May 26-31). *AraNLP: A Java-based library for the processing of Arabic text*. Paper presented at the proceedings of the 8th International Conference on Language Resources and Evaluation, Rekjavik, Iceland.

Aliguliyev, R. M. (2009). A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Systems with Applications Journal*, *36*(4), 7764-7772.

Aristoteles, H. Y., Ridha, A., & Julio, A. (2012). Text Feature Weighting for Summarization of Documents in Bahasa Indonesia Using Genetic Algoritm. *International Journal of Science Issues*. *9*(3), 1694-0814.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. (1st ed.). New York: Oxford university Press.

Barzilay, R., & Elhadad, N. (2002). Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, *17*(1), 35-55.

Baxendale, P. B. (1958). Machine-made index for technical literature: an experiment. *IBM Journal of research and development*, *2*(4), 354-361.

Bhandari, D., Murthy, C., & Pal, S. K. (2012). Variance as a stopping criterion for genetic algorithms with elitist model. *Journal of Fundamenta Informaticae, 120*(2), 145-164.

Borthakur, D. (2008, April). *HDFS architecture guide*. Retrieved June 6, 2016, from: Hadoop apache project http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

Bossard, A., & Rodrigues, C. (2010, October). *Combining a Multi-document Summarization System with a Genetic Algorithm*. Paper presented at the International Workshop on Combinations of Intelligent Methods and Applications, France.

Bouras, C., & Tsogkas, V. (2008, September). *Improving text summarization using noun retrieval techniques*. Paper presented at the International Conference on Knowledge-Based and Intelligent Information and Engineering System, Berlin, Germany.

Brill, E. (1992). A *simple rule-based part of speech tagger*. Paper presented at the Proceedings of the 3rd workshop on Speech and Natural Language, pp. 112-116. Association for Computational Linguistics, Trento, Italy.

Chowdhury, G. G. (2003). *Natural language processing*. Annual review of information science and technology, 37(1), 51-89.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Journal of Communications of the ACM, 51*(1), 107-113.

Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM (JACM)*, *16*(2), 264-285.

El-Ghannam, F., & El-Shishtawy, T. (2014). Multi-Topic Multi-Document Summarizer. *International Journal of Computer Science & Information Technology (IJCSIT)*, *5*(6), 77-90.

El-Haj, M., & Rayson, P. E. (2016, Mar 08). *OSMAN–A Novel Arabic Readability Metric*. Paper presented at the Language Resources and Evaluation Conference, Slovenia.

El-Shishtawy, T., & El-Ghannam, F. (2012, May 14-16). *Keyphrase based Arabic summarizer (KPAS)*. Paper presented at 8th International Conference on Informatics and Systems, Egypt.

Mohamed, E., & Kübler, S. (2011). Part of speech tagging for Arabic. *Natural Language Engineering Journal*, *18*(4), 521-548.

Ene, A., Im, S., & Moseley, B. (2011). *Fast clustering using MapReduce*. Paper presented at the Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. San Diego.

Erkan, G., & Radev, D. R. (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, *22*(1) 457-479.

Farghaly, A., & Shaalan, K. (2009a). Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP) Journal*, *8*(4), 14.

Feng, L., Elhadad, N., & Huenerfauth, M. (2009, 30 March – 3 April). *Cognitively motivated features for readability assessment*. Paper presented at the Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, Athens.

Ferreira Cordeiro, R. L., Traina Junior, C., Machado Traina, A. J., López, J., Kang, U., & Faloutsos, C. (2011, August 21–24). *Clustering very large multi-dimensional datasets with mapreduce*. Paper presented at the Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, USA.

Froud, H., Lachkar, A., & Ouatik, S. A. (2013). Arabic text summarization based on latent semantic analysis to enhance Arabic documents clustering. IJDKP *International Journal of Data Mining & Knowledge Management Process*, *3*(1), 79-95.

Geronimo, L. D., Ferrucci, F., Murolo, A., & Sarro, F. (2012, April 17-21). *A parallel genetic algorithm based on hadoop mapreduce for the automatic generation of junit test suites*. Paper presented at the IEEE 5th International Conference on Software Testing, Verification and Validation, Downtown Montreal, Canada.

Giannakopoulos, G. (2013, August 9). *Multi-document multilingual summarization and evaluation tracks in ACL 2013 multiling workshop*. Paper presented at proceedings of the MultiLing 2013 Workshop on Multilingual Multidocument Summarization, Sofia, Bulgaria.

Golberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. (1st ed). USA: Addison-Wesley Reading Menlo Park.

Goldstein, J., Mittal, V., Carbonell, J., & Kantrowitz, M. (2000). *Multi-document summarization by sentence extraction.* Paper presented at the Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization, Stroudsburg, USA.

Gong, Y., & Liu, X. (2001, September 09 - 12). *Generic text summarization using relevance measure and latent semantic analysis.* paper presented at the proceedings of the 24th annual international ACM conference on Research and development in information retrieval. New Orleans, USA.

Grama, A. (2003). *Introduction to parallel computing.* (2nd ed). USA: Pearson Education.

Gunarathne, T. (2015). *Hadoop MapReduce v2 cookbook: Explore the Hadoop MapReduce v2 ecosystem to gain insights from very large datasets.* (2nd ed). USA: Packt Publishing.

Gunning, R. (1969). The fog index after twenty years. *Journal of Business Communication*, *6*(2), 3-13.

Hans, N., Mahajan, S., & Omkar, S. (2015). Big Data Clustering Using Genetic Algorithm On Hadoop Mapreduce. *International Journal Of Scientific and Technology Research*, *4*(04), 58-62.

He, Q., Zhuang, F., Li, J., & Shi, Z. (2010, October 15-17). *Parallel implementation of classification algorithms based on MapReduce*. Paper presented at the 5th International Conference on Rough Sets and Knowledge Technology, Beijing, China.

He, Y.-X., Liu, D.-X., Ji, D.-H., Yang, H., & Teng, C. (2006, August 13-16). *Msbga: A multi-document summarization system based on genetic algorithm*. Paper presented at the International Conference on Machine Learning and Cybernetics, China.

Hewahi, N. M., & Kwaik, K. A. (2012). Automatic Arabic Text Summarization System AATSS Based on Semantic Features Extraction. *International Journal of Technology Diffusion*, *3*(2), 12-27.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* USA: University of Michigan Press.

Jin, C., Vecchiola, C., & Buyya, R. (2008, December 7-12). *Mrpga: an extension of mapreduce for parallelizing genetic algorithms*. Paper presented at the IEEE 4th International Conference on EScience, Indiana, USA.

John, A., & Wilscy, M. (2015, December 3-5). *Vertex Cover Algorithm Based Multi-document Summarization Using Information Content of Sentences.* Paper presented at the International Conference on Information and Communication Technologies (ICICT), Kochi, India.

Kang, U., & Faloutsos, C. (2013, December). Big graph mining: algorithms and discoveries. *ACM SIGKDD Explorations Newsletter*, *14*(2), pp. 29-36.

Khoja, S. (2001, June). *APT: Arabic part-of-speech tagger.* Paper presented at the Proceedings of the Student Workshop at NAACL.

Kincaid, J. P., Fishburne Jr, R. P., Rogers, R. L., & Chissom, B. S. (1975). *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel*. Institute for Simulation and Training, University of Central Florida.

Lin, C. (2004, July). *Rouge: A package for automatic evaluation of summaries.* In Text summarization branches out: Proceedings of the ACL-04 workshop, Barcelona, Spain.

Litvak, M., Last, M., & Friedman, M. (2010, July 11-16). *A new approach to improving multilingual summarization using a genetic algorithm.* Paper presented at the Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, *2*(2), 159-165.

Mat Daud, N., Hassan, H., & Abdul Aziz, N. (2013). A Corpus-Based Readability Formula for Estimate of Arabic Texts Reading Difficulty. *World Applied Sciences Journal*, *21*(2), 168-173.

Mc Laughlin, G. H. (1969). SMOG grading-a new readability formula. *Journal of reading*, *12*(8), 639-646.

McKeown, K., & Radev, D. R. (1995, July 9-13). *Generating summaries of multiple news articles.* Paper presented at the Proceedings of the 18th international ACM SIGIR conference on Research and development in information retrieval, Washington, USA.

Meng, X., Bradley, J., Yuvaz, B., Sparks, E., Venkataraman, S., Liu, D., et el. (2016). Mllib: Machine learning in apache spark. *Journal of Machine Learning Research*, *17*(34), 1-7.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems Journal*, *9*(3), 193-212.

Momtaz, A., & Amreen, S. (2012). *Detecting document similarity in large document collecting using MapReduce and the Hadoop framework*. BRAC University.

Munawar, A., Wahib, M., Munetomo, M., & Akama, K. (2008, September 25-27). *A survey: Genetic algorithms and the fast evolving world of parallel computing.* Paper presented at the 10th IEEE International Conference on High Performance Computing and Communications, Dalian, China.

Nagwani, N. (2015). Summarizing large text collection using topic modeling and clustering based on MapReduce framework. *Journal of Big Data*, *2*(1), 1-18.

Najeeb, M. M., Abdelkader, A. A., & Al-Zghoul, M. B. (2014). Arabic Natural Language Processing Laboratory serving Islamic Sciences. *International Journal of Advanced Computer Science and Applications IJACSA*, *5*(3), 114-117.

Nandhini, K., & Balasundaram, S. R. (2013). Use of genetic algorithm for cohesive summary extraction to assist reading difficulties. *Applied Computational Intelligence and Soft Computing Journal*, *2013*(8), 1-11.

Nenkova, A. (2006, September 17-21). *Summarization evaluation for text and speech: issues and approaches*. Paper presented at the 9th International Conference on Spoken Language Processing, Pittsburgh, USA.

Oufaida, H., Nouali, O., & Blache, P. (2014). Minimum redundancy and maximum relevance for single and multi-document Arabic text summarization. *Journal of King Saud University-Computer and Information Sciences, 26*(4), 450-461.

Paniagua, C., Flores, H., & Srirama, S. N. (2012, August 26-29). *Mobile Sensor Data Classification for Human Activity Recognition using MapReduce on Cloud.* Paper presented at 10th International Conference on Mobile Web Information Systems, Paphos, Cyprus.

Poli, R., Langdon, W. B., McPhee, N. F., & Koza, J. R. (2008). *A field guide to genetic programming*. UK: Lulu Enterprises.

Qazvinian, V., Hassanabadi, L. S., & Halavati, R. (2008). Summarising text with a genetic algorithm-based sentence extraction. *International Journal of Knowledge Management Studies, 2*(4), 426-444.

Radev, D. R., Jing, H., Styś, M., & Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing & Management Journal, 40*(6), 919-938.

Rao, D., & Yarowsky, D. (2009, August 7). *Ranking and semi-supervised classification on large scale graphs using map-reduce.* Paper presented at the Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing, Suntec, Singapore.

Saha, S. (2014). Parallelization of Genetic Algorithms using MapReduce. *European Journal of Applied Social Sciences Research (EJASSR), 2*(1), 20-35.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management Journal, 24*(5), 513-523.

Shaalan, K. (2014). A survey of arabic named entity recognition and classification. *Computational Linguistics Journal*, *40*(2), 469-510.

Shukla, A., Pandey, H. M., & Mehrotra, D. (2015, February 25-27). *Comparative review of selection techniques in genetic algorithm*. Paper presented at the International Conference on Futuristic Trends on Computational Analysis and Knowledge Management, New Delhi, India.

Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas Journal*, *18*(3), 491-504.

Silva, C., & Ribeiro, B. (2003, July 20-24). *The importance of stop word removal on recall values in text categorization*. Paper presented at the Proceedings of the International Joint Conference on Neural Networks, Oregon, Portland.

Sivanandam, S., & Deepa, S. (2007). I*ntroduction to genetic algorithms*. (2008 Ed). Springer Science & Business Media.

Sripada, S., Kasturi, V. G., & Parai, G. K. (2005). *Multi-document extraction based Summarization*. Final Project. Stanford University.

Stat, I. W. (2015). *Arabic Speaking Internet Users and Population Statistics.* Retrieved August 29, 2016, from http://www.internetworldstats.com/stats19.htm

Suanmali, L., Salim, N., & Binwahlan, M. S. (2010). SRL-GSM: A Hybrid Approach on Semantic Role Labeling and General Statistical Method for Text Summarization. *Journal of Applied Science, 10*(3), 166-173.

Thakkar, K. S., Dharaskar, R. V., & Chandak, M. (2010, November 19-21). *Graph-based algorithms for text summarization.* Paper presented at the 3rd International Conference on Emerging Trends in Engineering and Technology, Goa, India.

Todirascu, A., François, T., Gala, N., Fairon, C., Ligozat, A.-L., & Bernhard, D. (2013, October 15-16). *Coherence and cohesion for the assessment of text readability.* Paper presented at the 10th International Workshop on Natural Language Processing and Cognitive Science, Marseille, France.

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., . . . Baldeschwieler, E. (2013, October 1-3). *Apache Hadoop YARN: yet another resource negotiator*. Paper presented at the Proceedings of the 4th annual Symposium on Cloud Computing, Santa Clara, California.

Vekaria, K., & Clack, C. (1998). Selective crossover in genetic algorithms: an empirical study. *In Parallel Problem Solving from Nature* (pp. 433-444). Netherlands: Springer-Verlag.

Verma, A., Llora, X., Goldberg, D. E., & Campbell, R. H. (2009, 30 November- 2 December). *Scaling genetic algorithms using mapreduce*. Paper presented at the 9th International Conference on Intelligent Systems Design and Applications, Pisa, Italy.

Waheeb, S. A., & Husni, H. (2014). Multi-Document Arabic Summarization Using Text Clustering to Reduce Redundancy. *International Journal of Advances in Science and Technology (IJAST), 2*(1), 194-199.

White, T. (2012). *Hadoop: The definitive guide*. (3rd ed). USA: O'Reilly Media, Inc.

Whitley, D. (1994). A Genetic algorithm tutorial. *Statistics and Computing Journal, 4*(2), 65-85.

Whitley, D., Rana, S., & Heckendorn, R. B. (1999). The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology, 7*(1), 33-48.

Wu, G., Li, H., Hu, X., Bi, Y., Zhang, J., & Wu, X. (2009, August 21-22). *MReC4. 5: C4. 5 ensemble classification with MapReduce*. Paper presented at the 4th ChinaGrid Annual Conference, Yangtai, China.

Zanoli, R., & Pianta, E. (2009, December 12) *A multistage PoS-tagger*. Paper presented at 11th Conference of the Italian Association for Artificial Intelligence EVALITA, Reggio Emilia, Italy.

Zhang, J., Sun, L., & Zhou, Q. (2005, 30 October- 1 November). *A cue-based hub-authority approach for multi-document text summarization*. Paper presented at the Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, Wuhan, China.

Zhao, W., Ma, H., & He, Q. (2009, December 1-4). *Parallel k-means clustering based on mapreduce*. Paper presented at the IEEE 1st International Conference on Cloud Computing, Beijing, China.