LEBANESE AMERICAN UNIVERSITY

Auto-Indexing Arabic Texts Based On
Association Rule Data Mining
By

Rouba G. Nasrallah

A thesis
Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

School of Arts and Sciences
March 2015

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

Lebanese American University

School of Arts and Sciences - Beirut Campus

Thesis Approval Form

Student Name: Rouba Nasrallah I.D. #: 201104357

Thesis Title: Auto-Indexing Arabic Texts based on Association Rule Data Mining

Program / Department: Computer Science / Computer Science and Mathematics

School: Arts and Sciences

Approved by:

Thesis Advisor: Ramzi A. Hara

Committee Member: Rony Touma

Committee Member: Issam Kouatli

Signatures Redacted

Date: March 20, 2015

cc: Department Chair
School Dean
Student (original copy)
Thesis Advisor

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants to Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: Rouba Nasrallah

Signature: Signatures Redacted

Date: 11/03/2015

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

- I have read and understood LAU's Plagiarism Policy.
- I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
- This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Rouba Nasrallah

Signature: Signatures Redacted

Date: 11/03/2015

Acknowledgments

I would like to thank my advisor, Dr. Ramzi Haraty, for his help and constructive encouragements. As well, I would like to thank Mrs. Maha Zaarour, a former Arabic Teacher, for helping me with the manual indexing of the Arabic texts used in the experimental phase. Also, a special thanks to Ms. Nadine Chamoun and Ms. Hala Daouk. Finally, I would like to thank Walid K. Daher and Manal Houri who worked previously on the issue of auto-indexing Arabic texts.

Auto-Indexing Arabic Texts Based On Association Rule Data Mining

Rouba G. Nasrallah

Abstract

In this work, we propose a new model to enhance auto-indexing Arabic texts. Our model denotes extracting new relevant words by relating those chosen by the previous classical methods, to new words using data mining rules. The model uses the Apriori Algorithm - an association rule algorithm for extracting frequent sets containing related items - to extract relations between words in the texts to be indexed with words from texts that belong to the same category. These associations of words extracted are illustrated as sets of words that appear frequently together. Our results show significant improvement in terms of accuracy, efficiency and reliability when compared to previous works.

Keywords: Arabic texts, Auto-Indexing, Rule based Data Mining, Apriori, Frequent Sets, Pre-requisite.

TABLE OF CONTENTS

List of Tables

Chapter One

Introduction

Indexing text documents consists of analyzing the content of the text in order to retrieve its subject.

It is a very important task related to information retrieval; a field that is considered to be very important in the computer science domain due to the need of exploring different topics in our daily lives. As an instance, we all use ‘google’, a search engine used to search and retrieve information about different topics.

Manual indexing is very difficult and needs a big human effort from the person performing it, since one needs to read the whole text and analyze it. It becomes more difficult and needs when the length of the text is larger. Of course, the people responsible for performing document indexing are specialists, well-trained to do this kind of job. They should be experts in the documents language regarding grammar or vocabulary.

Indexing texts is needed in many domains and for different types of texts such as articles in newspaper and magazines as well as online articles. It is used for archiving, documenting and helping in e-mail spam detection, web page content filtering, and automatic message routing. But most importantly it is used for information retrieval.

1.1. Indexing Documents

There are two types of indexing according to Daher (2002): Thesaurus based indexing and Full-Text based indexing.

1.1.1. Thesaurus Based Indexing

In the first type (Thesaurus based indexing), the words chosen to represent the document do not need to exist in the text; however, their synonyms exist. The synonyms might be chosen by the documenter if the search for the text is usually done by using the synonym and not the exact word. Notice that the synonym is not only the correspondent of the term in the dictionary. For example, if we are talking about a certain football player, we can consider his name as a good index even if this name does not exist in the document. The problem of Thesaurus based indexing is the difficulty of implementing it due to the need of human intervention. The implementation should depend on a file containing the synonyms of the terms. This file should be always updated manually. This topic can be considered as further work that enhances the auto-indexing mechanism.

1.1.2. Full-Text Based Indexing

In Full Text based indexing, the indexing relies on choosing terms or phrases that already exist in the text. This type is much easier in concept.

1.1.3. Building Subject Headings

For both types, the result of indexing is a set of key words used to construct the subject of the text. Also, they can be used to construct many subject headings. The same document may have many subject headings. This will help in information retrieval systems because it will make it more accurate and raise the hit rate.

Documenters follow specified rules to build subject headings. The following fields should be included:

- *Name* – The name of person or organization the document is about. For example "Angela Merkel".
- *Position* – Social position. For example, “Chancellor” or “President”.
- *Location - Country/City/Town/Place*. For example, “Germany”.
- *Activity*. For example, “Meeting with president”.
- An example of a subject heading is the following:

  Angela Merkel>Chancellor>Germany>Meeting

The subject headings should contain at least one keyword. The search engines now can look at the subject headings instead of the whole text. Subject headings can help searching in different categories as well.

So far we have been describing indexing documents in general. But the difficulty remains in the details, especially in the difference between one language and another's degree of sophistication of rules. For example, the Arabic or Chinese languages have very complicated grammatical rules as well as a very rich vocabulary. This makes auto-indexing more complicated and difficult since many cases need to be taken into consideration in order to have accurate results. Therefore, and since our study aims to find a solution for auto-indexing, Arabic texts specifically, there is a need to implement an algorithm that handles the complicated syntax and logical structure of the Arabic language.

The idea behind this work is instead of having the documenter read the document, choose the key words and build the subject of the document or the subject headings, a computer program picks the key terms of the text automatically and returns them as output to the documenter. This will facilitate the work and spare the documenter time

and effort reading the document and analyzing it in order to find the right subject. Therefore, the goal is to automate a part of the work. The proposed solution, as we mentioned earlier in this report, is to extend the set of relevant words extracted from the text based on relations of words extracted from the prerequisite texts of the same category of the text to be indexed. The method used will be described later in this report in chapter four.

1.2. Organization of the Thesis

The second chapter of this report tackles the related work to the topic of Auto-indexing Arabic text documents, including the previous work done by Daher (2002), in which the words of the text are assigned a weight based on their frequency and spread in the text.

The third chapter presents the pre-processing phase in which the terms of the text are stemmed and the original words are extracted. In this chapter, some of the grammatical rules will be stated briefly. Many details will be mentioned in this chapter, all explaining the stemming phase which is considered as one of the most important and difficult phases in the auto-indexing issue due to the difficulty of the Arabic language. Also, this chapter tackles the weight calculation phase that was used in the work done by Daher (2002). In our work we depend on this phase to choose index words out of the text to be indexed and out of the pre-requisite texts so we can use them in our proposed solution.

Chapter four is considered to be the core chapter of this report since it presents the proposed solution. This chapter explains the mechanism used to choose the index words and extract the relevant terms out of the text to be indexed.

In chapter five, we describe how candidate index words are selected.

Chapter six presents the implementation phase in details.

In chapter seven, the experimental phase is described, and the results are analyzed and discussed.

Finally, the work is summarized in the conclusion. The progress that was made will be mentioned as well as the enhancements proposed and the further work.

The report also includes three appendices. The first appendix lists stop list terms and phrases categorized by their type. Appendix B contains a case study text as well as its analysis. Finally, appendix C contains the files to be accompanying the thesis in the submitted CD.

At the end of this report, the references are presented.

Chapter Two

Related Work

As was mentioned in chapter one, the aim of this work is to build an Auto-Indexing system for Arabic texts. This system is used to extract index words that represent the subject of the text. The work takes into consideration the relation between words and uses association rule based data mining techniques to tackle the auto-indexing problem.

2.1. Text Classification and Rule Based Data Mining

Some work has been done in the literature of the domain of text classification related to the issue of the report.

A number of methods have been proposed related to classification algorithms for text categorization. These methods are based on classification algorithms such as Naïve Bayes proposed by McCallum and Niagam (1998.), decision trees by Sahami, Dumais, Heckerman and Horvitz (1998), neural networks by Joachims (1998), and SVM by Dumais (1998). In Yang and Liu (1999) some of these methods were compared. The result shows that SVM is the best among the others regarding document classification.

As for the Arabic language, work has been done to classify texts. Al-Harbi, Almuhareb and Al-Thubaity (2008) discussed automated document classification considering it an important text mining task. Text classification aims to automatically

assign the text to a predefined category. The authors proposed a solution based on linguistic features. They generated the feature frequency of the lexical features that they have extracted and then they calculated the importance of each feature locally (for each class) based on Chi Square. Seven datasets were used including writers, poems, web, forums and others and each dataset contained different classes. The assembled corpus comprised 17,658 texts with more than 11,500,000 words. A tool was implemented to extract feature using SVM and C5.0 (Classification algorithms) and to decide to which class the tested texts belong. The results showed that in general C5.0 classifier is more accurate.

Khoja (2001) proposed a tagger that uses rule-based techniques as well as statistics to solve the problem of auto-indexing. The Arabic parts-of-speech studied are divided into verbs, nouns and particles. All affixes are removed using a stemmer to produce the stem word of each term in the text.

Gawrysiak, Gancarz and Okoniewski (2002) presented the shortcomings of the unigram and *n*-gram document representation techniques used in text mining, as well as proposing a solution that takes into consideration the position of a word in a text document. In Larkey and Connell (2001) as well as Billhardt, Borrajo and Maojo (April, 2000), approaches to handle co-occurrences were implemented.

In work done by Rahman, Sohel, Naushad and Kamruzzaman (2010) a set of text documents belonging to three categories (computer science, electrical engineering and mechanical engineering) were preprocessed, constructing a set of stem words for each text. The Apriori algorithm is then applied on the extracted sets. The results are sets of words that appear frequently together. Then they preprocessed the new document that they aim to classify and extracted out of it the frequent sets of words. The authors calculated the probability of each of these sets in each category. Then

they calculated the probability of having the text in a specific category by multiplying the probabilities of the frequent sets in every category. The category with the highest probability was that to which the text belongs.

In the work done by Daher (2002) a new method was proposed consisting of conceptual vectors based on the semantic relations (synonymy, homonymy...) between words. These fields are used to construct a semantic field database. Then, the meaning of the textual segments is calculated in the semantic fields. A conceptual vector tries to represent a set of associated ideas to textual segment (word, sentence, syntagm, etc.). Every term and every concept is projected by a conceptual vector. So, a vector corresponds to a linear combination of the terms meaning. They constructed automatically a database that contains common words in the same semantic fields. The synonyms of the words can be collected and used in this case. Then they calculated the angular distance between different vectors to find proximity between meanings.

Sharef, Omar and Sharef (2014) used a new classifier called FRAM for Arabic text categorization. Their experiments were divided into two stages, the training stage and the test stage. During the training stage, they pre-processed the documents that initially belong to a category and then extracted the important keywords. After that, they formed a database of the feature terms and used it in the testing stage. In the testing stage, they considered a set of categorized documents as uncategorized, and they classified them using a classifier. This is how they measured the categorization performance by using several standard techniques of performance evaluation. The purpose of using the classifier FRAM was to calculate the Frequency Ratio for each feature of the new document based on the candidate features of the training phase. Then they assigned the document under the category that obtained the maximum

value of the summation of frequency calculated previously. They concluded that FRAM was better than three Bayesian learning classifiers; MBNB, MNB and NB.

Hattab and Hussein (2012) proposed a three-model system. The first deals with the pre-processing phase. Then the result of the first model is cleaned while going through the second model consisting of error detection and correction of the corpus used. After that comes the classification model where it is decided to which domain the text belongs. In the classification model, each document of the training corpus will be classified. Then, the N-gram profile will be generated. This is done by splitting the text into tokens consisting only of letters and removing all digits. All possible N-grams, for N=3 will be computed. The frequency of occurrence of each N-gram is calculated and sorted from the highest frequency to the lowest. The similarity of the N-Grams profile of each document is compared against the profiles of all the documents in the training classes. For each N-gram in the document profile, search must be performed for the N-gram in the class profile, and then the difference between their positions should be calculated. For N-grams that are not found in the class profile, a maximum value is assigned. After that, all N-grams in the document profile have been exhausted. The second measure, after all N-grams in the document profile have been exhausted, is computing the sum of the distance measures. The detection and correction algorithm outperformed the Bayes algorithm by about 10%. Without checking misspelling errors, accuracy is 68.85%, while the average accuracy for the classification system with misspellings detection and correction is 71.77%.

El-Shishtawy and El-Ghannam (2012) proposed an accurate Arabic Root-Based lemmatizer for information retrieval purposes. The aim was to build a lemmatizer with minimum sufficient resources. Lemmatizer transforms inflected word form to its dictionary lemma look-up form. For example, in the case of nouns and adjectives,

the lemma form is the singular indefinite (masculine if possible) form. In the case of verbs, it is the perfective third person masculine singular form. This way, they can collect more information about the word to be stemmed and its context to generate more accurate word features. The system generates accurate lemma form and its relevant morpho syntactic features that support information retrieval purposes. Morpho syntactic features are required also to capture the important semantic senses of the language. Collected information about words also include word pattern. During the implementation of the approach the authors took into consideration the following features: 1) the power and generality of rule-based stemmers, and the accuracy of dictionary based approaches, 2) Arabic context morpho syntactic rules are used to expect the correct word category and then verified. For example, the algorithm uses the word pattern and the category of its previous word for identifying the current word, 3) simple rules used to re-categorize nouns as adjectives. The presented algorithm proves that accurate results for POS tagging (part of speech tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context), can be achieved when using inherent features and rules of Semitic languages like Arabic. It is shown that ambiguity can be resolved using metadata about patterns, roots, and infixes' indications. Analysis is aided with auxiliary dictionaries and syntax rules to produce a lemmatizer, which outperforms exciting up to date Arabic learning algorithms.

2.2. Auto-Indexing Based on Frequency and Spread of Words

Previously, a very important work has been done in the work of Daher (2002) on the issue of auto-indexing Arabic texts. As we have already mentioned, our work relies on that of Daher (2002), and is considered as an enhanced continuation of it.

In this section, we will present a summary of the model proposed, and all the details can be found in Daher (2002).

The model consists of four layers, each independently implemented. The layers are somehow connected since they exchange information. It is a process of phases, where the first one is reading the document. The second is extracting stem words, but this phase can easily be plugged off the system and replaced by another stemming technique. The third and fourth phases consist of assigning the weight and selecting the index words. In the first phase, the model uses a Stop-List containing terms (words/phrases) that should not be taken into consideration while choosing the index words since they do not contribute to the meaning of the text. After removing the Stop-List terms, the second phase begins. In this phase, the remaining words are considered as candidate words that need to be stemmed. The stem words will form an array that are used as input of the third phase in which the weight calculation is done. In the fourth phase, the index words are selected.

All the phases of the model proposed by Daher (2002) are important, but the phase in which the weight is calculated is considered the core of the system. Three factors influence the weight of the stem word. As we mentioned before, the frequency of appearance of the word in the text is one of the factors. The second one is the count of the stem words for that word. Daher (2002) proposed adding another factor which is the spread of the word in the text. This is to prevent choosing a word as an index if

it appears in a specific part of the document and not in all of it because in this case it is less likely that the word represent an index word of the text.

In chapter three, the stem word extraction phase as well as the weight calculation phase will be described briefly.

Chapter Three

Stem Word Extraction and Weight Calculation

This section gives a brief summary of one of the most important steps in our work that tackles the problem of stem word extraction. All the details can be found in the work done by Daher (2002). The second phase of the model proposed by Daher (2002), and on which we rely in our work, consists of extracting the stem words. After reading the document, the analysis of each word starts. This means that all the stop-words should be removed and not taken into consideration while extracting the index terms. Also, this phase differentiates between the two types of terms, verbs and nouns to be able to perform the right stemming technique that changes from one type to another.

- But what does stemming mean?
  - ✓ Stemming a word is extracting its root.
- Why is stemming an important step?
  - ✓ Stemming allows the auto-indexing algorithm to treat different forms of the same term as one word and not consider them independent. Notice that different forms of the word mean, for example, singular or plural form, adjective or subject etc. This will improve the indexing mechanism by reducing redundancy; for example, a word and its plural have the same meaning and contribute equally to the subject of the whole text. Therefore, they shouldn't be treated differently.

We shouldn't forget that the count of stem terms of a certain word in the text is one of the factors of the weight calculation; therefore, stemming is a preliminary phase of the auto-indexing work.

3.1. Stem Word Extraction

3.1.1. Rhyming

Rhyming is the first step in the stemming mechanism. In this step, we need to take into consideration that in the Arabic language we have two types of words, verbs and nouns. Each type has different rhyming; that we should rely on to be able to differentiate between verbs and nouns. In this step, each word is compared to a set of rhythms predefined in the Arabic grammar. In addition, we are able to identify through rhyming the tense of the verb (past, present or future) and whether a noun is singular or plural. It is also used to identify attached pronouns so they can be removed. We need to state that in the Arabic language all the verbs rhyme with different standard validation of the word 'فعل' (i.e., did).

3.1.2. Removing stop-list terms

The second step consists of removing stop-list terms that are words that do not contribute to the meaning of the text. A predefined stop-list is used to remove all the stop-words from the text to be indexed. As an example, in the English language we can consider the words 'it', 'the', 'never', 'where', 'that', 'numbers', etc. as stop-list terms. However, we should mention that these can also be used to identify whether the word that follows is a verb or a noun.

On a deeper level, we can also talk about stop-list phrases. These are sentence with no contribution to the meaning of the text. "To all my dearest readers", for example

is a sentence that does not add any value to the text meaning. The auto-indexer should not take into consideration the stop-list terms and phrases.

After defining what stop-list words and phrases are, we will describe briefly how stop-list terms are removed from a text document.

As mentioned above, the auto-indexing program uses a predefined stop-list document. If a word in the text is contained in the stop-list set of words, then it should be removed.

Concerning stop list phrases, the first word in every sentence of the text is checked for its belonging in a set containing the starting words of commonly used stop-list phrases. If the first word matches, then the rest of the phrase is compared to each of the stop-list phrases that begin with the first word.

3.1.3. Identifying nouns and verbs

This is a very important step in the stemming phase, in which verbs are differentiated from nouns, so that stem words can be extracted from them based on their type.

Identifying whether words are nouns or verbs consists of two steps:

1. Checking and testing the term that precedes that word. Some of the stop-list terms are always followed with verbs, while others precede nouns. So if a word is preceded by any of the stop-list terms that are always followed by a verb, then it is a verb. The same thing is applied to indentifying nouns. For example, the word ‘لم’ that means never, is a stop-list term that is always followed by a verb.
2. Distinguishing between verbs and nouns can rely on rhythm of the word itself. Rhyming is a very important thing on which the grammar of the Arabic language is built. Some of the used rhymes are specific for verbs; for

example, ‘يفعل’ (i.e, does) or ‘افعل’ (i.e, do). Others are for nouns, for instance ‘فاعل’ (i.e, doer) or ‘مفعول’ (i.e, done).

These two methods are our first trial to indentify verbs and nouns. In the case of words that cannot be identified using these two methods, we examine the pronouns attached to the word. Some of the pronouns are only attached to nouns, while others are attached to verbs.

3.1.4. Extracting stem words

In this section, we describe how extracting stem words is done. To extract stem words, an algorithm called "stemming algorithm" is applied to each word in the document.

Different stemming procedures need to be taken into consideration due to the fact that we have different types of verbs in the Arabic language. The output of any algorithm is the original form of the verb. In the following section, we will describe the phases of the algorithm we are using by starting with removing attached pronouns as suggested by Gillman (1990).

3.1.4.1. Checking Attached Prefix/Suffix Pronouns

The first step in the stemming algorithm is to remove the pronouns that are attached to a word. One of the characteristics of the verbs in the Arabic language is having two forms of pronouns: attached and discrete.

The discrete pronouns are stop-list terms and are eliminated in the phase that handles removing them. A word may consist of attached pronouns that are very important to identify in order to separate them from the word. The place of the attached pronouns can be at the beginning or the end of the word and sometimes on both sides. We have

a defined set that contains a list of all the pronouns. We should mention that this list is finite and it includes prefixes, suffixes, and possible combinations. In this step we have an algorithm whose job is to do a pattern matching by looping over the set of attached pronouns, and in case of existence of an attached pronoun, it removes it.
The set of the different forms of prefix pronouns and suffix pronouns can be found in the work done by Daher (2002).

3.1.4.2. Checking Verbs against the 'Five Verbs'

Another characteristic of the Arabic language is having five standard verbs known as the 'five verbs' and can be found in Deeb (1971) as well as Kindery, Rajihy and Shimry (1996). These verbs are special forms of the Arabic verbs and have special properties. One of the important things that we need to know about them is that they are always in the present tense and they usually end with 'ن'. Also, non-essential letters are attached to these verbs but not considered as pronouns. Therefore, these attached letters are not detected in the phase in which attached pronouns are removed. As an example, the verb 'يفعلون' (i.e., they do), where 'ي' and 'ون' are non-essential; however, they are not removed with the attached pronouns. In this phase, pattern matching is not used. Instead, rhyming is used to detect whether a verb belongs to the 'five verbs' set. The list of the 'five verbs', as well as the result of the 'five verbs' when preceded with either 'أدوات نصب' or 'أدوات جزم', can be found in Daher (2002).

3.1.4.3. Checking Verbs against the "Ten-Verb-Additions"

The third characteristic of the Arabic verbs is the ten different derivation formats that come from the original three letter verb. These derivations, similarly to the 'five

verbs', have essential letters and non-essential letters. Three of the ten-derivations are obtained by adding one letter to the original stem verb, five are obtained by adding two letters, and the other two derivations are obtained by adding three letters.

Rhyming is also used in this case to detect the verbs that are considered a derivation. Once the algorithm detects the verb, it will remove the non-essential letters. Therefore, the stem verb is obtained.

The 'ten derivations' list is found in Daher (2002).

3.1.4.4. Extracting Stem Words from a Noun

So far we have seen the methods used to extract the stem word of the verbs, but what about the nouns? In this section we will describe the way the stem word of a noun is extracted. However, we should mention that, although stemming nouns are more complicated than stemming verbs, they have some common steps.

Different factors affect the difficulty of stemming a noun. The first is the different forms in which a noun might come in the text, whether singular, double or the plural form. Needless to say that these forms also differ when the noun addresses a male or a female. The steps of extracting stem word from a noun are: check attached prefix and suffix pronouns; check the noun against the ‘five nouns’; restore a noun to its singular form; compare the noun to the ‘common derivations’: M-derivations, T-derivations, and miscellaneous derivations. The details as well as the rhyming algorithm can be found in Daher (2002).

After the stemming stage is over, it is time to calculate weights of the words in the text. The weight of a term relies basically on three factors, as shall be stated in the following sections of this chapter. The formula for weight calculation will be described in this chapter as well.

3.2. Weight Calculation

We mentioned earlier in this report that our work depends on the one done in Daher (2002), and is considered an enhanced continuation of it. This is why the weight calculation phase is considered a very important step, on which our work relies to extend the list of index words; and therefore, will be described in the following sub-sections.

3.2.1. Factors affecting the weight

The weight of a word is calculated depending on three factors: the count of the term in a text document, the count of the stem words for that word in the text document, and the spread of that word along the document.

The last factor was introduced based on the fact that if a word appears frequently in a specific part of the text, it is not necessarily that it reflects the subject of the text. The more the word is spread along the text, the more it should be considered as an index word. These three factors are used in formulas to calculate the weight of the words.

3.2.2. Formulas and terms used

Calculating the count of the word and of its stem isn't of any difficulty since it can be done by counting the occurrence of each term in the text.

The calculation of the spread factor takes more effort, and needs to be calculated using a complex formula.

The weight calculation is done using the following formula:

$$w = m \times sm \times f$$

In this formula, *m* is the count of a certain word in the text, *sm* is the count of stem words for a certain word in the text. The last factor which is the spread of the word within the document is represented by *f*.

The formula was developed in Daher (2002) and all the details can be found there.

This formula takes into consideration the following factors:

- *d* is the word's position in the text, i.e., its distance. It is found by counting the words preceding it in the text.
- *ad* is the average of all distances for each stem word of each term in the text, defined as

$$ad = \left\lceil \frac{\sum_{i=1}^{sm} d_i}{sm} \right\rceil$$

- *id* is the ideal distance between every two occurrences for each stem word.

$$id = \left\lceil \frac{N}{sm+1} \right\rceil$$

  Notice that *N* is equal to the count of all terms in document.

- *aid* is the average of all ideal distances for a stem word and is given as

$$aid = \left\lceil \frac{\sum_{i=1}^{sm} i \times id}{sm} \right\rceil$$

Since $\sum_{i=1}^{sm} i = \frac{sm \times (sm+1)}{2}$ and $id = \left\lceil \frac{N}{sm+1} \right\rceil$, *aid* becomes $\left\lceil \frac{N}{2} \right\rceil$.

- *g* is the difference between *aid* and *ad* called gap, i.e., $g = aid - ad$.

Note that *g* may have a positive or a negative value. The smaller *g* is, the more the word is spread in the text.

This section gives a small summary of all the factors that were used in the calculation of the weight of the words in a text.

Our work relies on the key factors used to calculate the weight of terms in the text that needs to be indexed.

We will not go into further details regarding the calculation of the weight. Further details, as well as, the general algorithm can be found in Daher (2002).

Chapter Four

Proposed Solution

Auto-indexing Arabic texts consists of automatically extracting index keywords that represent the text and that are related to its subject. The aim of this thesis is to improve the auto-indexing mechanism, and to enhance the results of the extraction of relevant index words.

To enhance the work done in this field, we propose taking advantage of the relations between words in a text. This is due to the fact that some words might not appear frequently in the text however they are somehow related to its topic and should be considered as index words. Based on relations of words, the number of indexes extracted out of the text will be expanded, which makes the auto-indexer more accurate.

For example, if we have a text from the category of "استخراج النفط". The word "البترول" might not be considered as an index word of the text, having taken into consideration the count of the word and of its stem, as well as the spread of the word in the text. But, why not taking advantage of the fact that this word is related to the term " النفط"and they both appear together frequently in texts of the same category? In this case, the use of relations between words and its integration in an auto-indexing system will allow the extraction of new relevant words related to the subject of the text. In our example, new relevant index word will be "البترول", not appearing frequently in the text, however contributing to its meaning.

To generalize, we take TEXT1 as the text to be indexed. TERM1 is not an index of TEXT1. However, TERM2 is an index.

Many texts from the same category of the TEXT1 are used. If TERM1 and TERM2 appear frequently together in these texts; they are related together.

We assume that since TERM1 is related to TERM2 that is an index of TEXT1, then TERM1 should be considered as a relevant word of the text. This means that TERM1 contributes to the meaning of the text and should be considered as an index word.

The solution proposed in this thesis is considered to be innovative, since the use of relations of words to enhance the auto-indexing mechanism hasn't been tackled before.

In addition, the experimental results, presented later in this thesis in chapter 7, showed a remarkable improvement.

Below, the proposed solution is explained in details.

4.1. Phases of the proposed solution

The proposed solution is based on four phases: collecting the pre-requisite texts, generating item-sets of frequent patterns using Apriori, extracting relevant frequent sets, and selecting the candidate index words. The first three phases will be described in details in the following sections. The fourth step will be discussed in the following chapter.

4.1.1. Collecting the Pre-Requisite Texts

First, the work relies on choosing texts from the same category of the document to be indexed. They are called pre-requisite texts that help identifying relations between words in text of a certain category.

The text to be indexed is extracted from a certain categorized corpus. Out of the same category, many texts should be extracted to be used in our model as pre-requisite information. Based on these pre-requisites, relations of words are formed. These texts are considered to be the fuel of the engine that represents the proposed model.

The pre-requisite texts will be indexed using the model proposed by Daher (2002). Out of the index words of each pre-requisite text, we need the words that represent the text the most. This is why the top five terms with the highest weights are chosen.

We will end up having a number of sets each corresponding to a text and containing five terms. These sets are considered the representatives of the pre-requisite texts. Also, words in these sets appear more likely in other texts of the same category. This is the main reason behind using relations of words in texts of the same category. For example, all texts belonging to the category of "استخراج النفط" have words in common such as "البترول", "النفط", "الدولار", "برميل" and others. These words appear frequently together in a text from this category and are considered to be related to each other.

The more we have pre-requisite texts, the more we have relations of words; and therefore, the results become more accurate.

The pre-requisite sets of words will be used in the second phase of our model in which the relation between words is extracted and the words that appear frequently together will be stated. This is done using the "Apriori" algorithm.

4.1.2. Generating Item-Sets of frequent pattern using Apriori

Second, we need to build sets of items appearing frequently with each other, and therefore related. The Apriori algorithm is used to achieve this task.

But, what is this algorithm and why do we use it? Apriori is an algorithm that generates frequent item sets and association rules over transactional databases. After identifying the frequent individual items in a database, Apriori extends them to larger item sets that appear frequently often in the database. According to Bhalodiya, Patel and Patel (2013), Apriori is an algorithm used to generate the candidate item-sets of frequent patterns that appear together. It is used to extract association rules of items in different sets and generalize the trends of the database.

In our case we will use this algorithm to extract associations between words of the different sets, extracted from the pre-requisite texts, which will highlight the relations between words in texts of the same category. Based on this, we can extend the candidate indexes of a text.

Briefly, the steps of the Apriori algorithm will be mentioned, but we will not go in details since they can be found in Rahman et al. (2010).

As we mentioned before, the main goal of using this algorithm is to extract frequent sets of words that are related together. The relations between words are used in the indexing mechanism that we are proposing in order to extract more relevant words out of the text.

The algorithm takes as an input the sets of words extracted out of the pre-requisite texts. Out of these sets a list of items is formed. This list contains all the words that appear in the pre-requisite sets but with a condition that the word will not appear twice in the list.

Next, lists of transactions are formed. Each transaction corresponds to one of the pre-requisite sets. Every transaction contains the indexes of the words of the set it represents in the list containing all the pre-requisite words.

To make this clearer, the following example is presented:

- S_1, S_2 and S_3 three pre-requisite sets as follows:

  $S_1 = \{term_1, term_2, term_3, term_4\}$

  $S_2 = \{term_2, term_4, term_5, term_1\}$

  $S_3 = \{term_5, term_3, term_7, term_6\}$

- L is the list containing all the pre-requisite words in a distinct way:

  $L = \{term_1, term_2, term_3, term_4, term_5, term_6, term_7\}$, notice that the first term in the list $term_1$ has the index 0, etc.

- T_1, T_2 and T_3 are the three transactions corresponding respectively to S_1, S_2 and S_3:

  $T_1 = \{0, 1, 2, 3\}$ since $term_1$ in S_1 is the first in L, $term_2$ is the second etc.

  $T_2 = \{1, 3, 4, 0\}$.

  $T_3 = \{4, 2, 6, 5\}$.

These transactions are then used as input of the algorithm responsible of extracting the frequent sets of words related to each other. The output will be sets of words appearing frequently together in these transactions.

4.1.3. Extracting relevant frequent sets

Third, not all the frequent sets extracted in the previous phase by Apriori should be considered as relevant. Some of them might not contain any of the indexes of the document we are trying to index. In this case the words in these sets are not related to any of the index words extracted of the text to be indexed. That is why this phase is introduced to extract only the frequent sets that are relevant.

After dealing with the pre-requisite texts and extracting the frequent sets of words that are related together, we should deal with the text to be indexed.

We apply the auto-indexing model proposed Daher (2002) to extract the index words out of the document that we wish to index. We considered that the words with the highest weights represent the text the most. The top five words with the highest weights are considered in the following work.

Only the frequent sets that contain one of these index words are used. This is, as well as the sets that contain a combination of the index terms extracted from of the text to be indexed. In other words, if a set of frequent items contains at least one of the index words of the text to be indexed, this means that it presents new terms that are related to this index. Therefore, the new words introduced in frequent sets are considered as important as the index term to which they are related, and can be considered as candidate relevant index words. This is helpful to the auto-indexing mechanism since it expands the number of candidate relevant index words. Some of the words that were not considered as index words of the text to be indexed are now being taken into consideration.

Notice that to find the combinations of the index words we apply an algorithm that extracts the power sets. The five index words are given as an input to an algorithm that generates power sets or in other words combinations of these indexes.

In the following section, the selection of the relevant sets is described.

4.2. The Output of the proposed Solution

Our proposed solution relates the outputs of previous solutions, to new words that are considered to be different outputs that gives more relevant words.

The relevant frequent sets returned in the third phase of the proposed solution are the output of our work. These are sets of related words that appear frequently together in texts of the same category of the text to be indexed.

But how is this going to help us in the indexing mechanism? This will be presented in the following chapter.

4.3. General Algorithm

In this section we present a high level algorithm of the model of the proposed solution discussed above. This algorithm describes briefly the implementation. The detailed implementation of this algorithm as well as its integration in the software programmed by Daher (2002) will be presented in Chapter 6.

```
// for text to be indexed
1. Apply software proposed by Daher (2002) to auto-index the document to be
indexed

2. Transaction T = Select top 5 index words of docToBeIndexed
//(indexed words sorted from highest weights to lowest)

// for each pre-requisite text
3. Transaction t_x = Select top 5 index words of pre-requisite doc //(indexed words
sorted from highest weights to lowest)

//using Apriori
4. Array[] ar = getfrequentsets(t_1,...., t_x)

// Select all the combinations of the index words
5. Array[] ar1 = PowerSetPart(T)

// Select all the relevant frequent sets that contains index words
6. Array[] result
int i = 0
Foreach set in ar1 // for each combination of indexes
        {
                Foreach set1 in ar // for each frequent set
                {
                        if set1.contains(set)
                        {
                                result[i] = set1
                                i++
                        }
                }
        }
```

First, the top five weighted index words are selected out of the text to be indexed. These indexes form transaction T.

Then, the top five weighted index words are selected out of each of the pre-requisite texts chosen. These indexes form t_1, ... ,t_x each corresponding for a text of the pre-requisites.

Apriori is applied taking as input t_1, ... ,t_x. The output will be the sets of words that appear frequently together.

The frequent sets that contain any of the index words of the text to be indexed by T are chosen represented.

This will allow us to expand the set of words that are more likely to be indexes of the indexed text, which will produce more relevant words extracted out of this text.

4.4. Selecting the candidate words

Selecting the candidate words is considered to be the first step in finalizing the work. The second step is selecting the index words out of these candidates.

The output of our proposed model is a collection of word sets, each containing words related to one of the index terms of the text to be indexed. These index words are selected based on their calculated weight that relies on the count of the word, the count of the stem of the word, and the spread of the word along the text. At the end of our method, these words are associated to a set of new words. This will expand the number of candidate words selected for a certain text, and gives more options of words to be considered as indexes.

A word might not be chosen as a candidate index word by previous models that only considers the count of the words and their stems as well as the frequency of

appearance of a word in a text. However, the relations with new words presented by our work make the word candidate to be considered as an index of the text.

Our model expanded the number of relevant words that are extracted out of the text and that can be considered as its indexes.

4.5. Index selection

The work proposed in this thesis relies on choosing five index words from the text to be indexed, as well as the pre-requisite texts and using them as an input of our approach. But how these indexes are chosen?

We mentioned earlier that we select the top five indexes produced by Daher (2002) with the highest weights, this means that the indexes were simply chosen by selecting the words with the highest weights. The calculation of the weight is described in Chapter 3. Our proposed solution associates new words to these indexes. There are many other index selection mechanisms that vary according to the target of the auto-indexing algorithm.

In the following section, we will present different index selection mechanism.

4.5.1. Various index selection mechanisms

The index selection mechanisms differs based on the task that the auto-indexing system aims to accomplish. This means that the index selecting mechanism in the case of a newspaper archiving system is different from that of an internet search engine system.

Among the interesting mechanisms used to select indexes, we have the one used in the case of book indexing mechanism. The index of a book is a set of keywords that are sorted alphabetically, along with the page number where they occurred within the

book. After stemming the words of the text and assigning each a weight, not all words that occur in the book will be chosen to appear in the index. A threshold is set, and words with a greater weight are chosen in the index.

Another index selection mechanism is the one used in the information retrieval systems. In this case, all the words of the document are considered to be indexes. Every document is assigned a unique ID. This ID as well as the name of the document and its path, are stored in the database of the system. Every word is assigned a unique ID, also inserted in the database along with its weight ID. This will create an association between the documents and the terms.

Obviously, each document contains so many terms, and each term may appear in more than one document. This association will present how relevant each term is to the document.

The user will type a few words, the information retrieval system will extract the IDs of the terms out of the database and search for their corresponding document ID in which they occur. The relevant words are sorted based on their weight and returned to the user.

Chapter Five

Experimental Results

All the work we have done so far will not be important if it is not useful. To achieve this goal, it needs to be implemented as a computer program especially that the whole idea is to computerize the auto-indexing work. The implementation of this work produces a software used in the field of auto-indexing, is one of the most important phases of the work.

In this chapter we will give a small summary about the previous implementation specification of the software produced by Daher (2002) due to the fact that we depend in our proposed solution on it. Then we will describe how we implemented and integrated our proposed solution. The auto-indexing software produced by our work is given on a CD with this report. Appendix C describes the content of the CD accompanying the report. This implementation part will lead us to a more important part of this thesis; the Experimental results. The remarkable results will prove the usefulness of our method. The main goal of the auto-indexing mechanism is to retrieve words that weren't considered by any of the previous auto-indexers; however relevant to the subject of the text to be indexed. The more relevant words are, the more accurate the auto-indexer is.

5.1. Previous Implementation

The work done by Daher (2002) was considered to be a big improvement of the auto-indexing mechanisms proposed. It was implemented to consider, other than the count of the word and the count of the stems of the word, the spread of an item along the text.

5.1.1. Data structure

The data structure of the software consists on arrays and classes. The software was designed in an object-oriented way.

The first class is the '*Word*' class composed of two attributes: the '*word*' which contains string of characters for each word in the text; and the '*distance*' attribute - an integer that represents the distance of the term from the beginning of the text.

The second class is the '*Document*' class. It has the entire text. The first attribute is an array containing objects of the class '*Word*' called '*Words*'. Every word of the document is read and inserted into the latter array. The second attribute of the '*Document*' class is the '*CountWords'*. This attribute holds the number of occurrence of the word in the text document. Notice that all the words of the text are still included at this stage and the stop-list terms are not yet removed. The last attribute for the '*Document*' class is the '*AverageIdealDistance'*. This last attribute equals half the '*CountWords'* attribute, and it is used to calculate the gap value of a certain term. Other than these attributes, this class has three methods: '*OpenFile*', '*CloseFile*', and '*ReadFile*'.

Another two classes that inherits from the class '*Word*' are: '*CandidateWord*' and '*StemWord*'. These two classes inherit the '*Word*' string attribute. However, instead of the '*distance*' attribute, they have the '*count*' attribute that represent the count of occurrence of a word in the text. Two additional attributes in the '*CandidateWord*'

class are stated. The first is called '*StemWord*' attribute and contains the stem word of each '*CandidateWord*' instance. The second is the '*Weight*' attribute that holds the weight of the corresponding candidate term. The '*StemWord*' class only has one single additional attribute called '*AverageDistance*' that contains the average of all distances for the corresponding candidate words of this stem word.

In addition to the classes, we have the arrays of objects. The '*CandidateWords*', '*StemWords*', and '*StopListTerms*' are arrays of objects.

5.1.2. Software Description

We have finished with the data structure of the software implemented by Daher (2002). Now it is time to describe the software.

The software presented by Daher (2002) as an auto-indexer is like a wizard. The user is directed in more than one step.

In the first step, the user chooses the path of the Arabic text that needs to be indexed.

In the second step, the stop-list terms are excluded and we end up having all the candidate words stemmed using the stemming algorithm. At the end of this second step, the user will be able to see all the candidate words with all their stems as well as their count of occurrences.

In the third step the weights are calculated. The user can choose to view in the fourth step the candidate words or the stem words along with their weights. The user has the option to check the stop-list terms. This is done by pressing a button that opens a screen of stop-list terms in which the user can add, delete or modify a stop-list term. In this screen the user can check the type of the stop-list term and modify it. The type of the stop-list terms is very important in the stemming algorithm since according to it the algorithm can choose if the following word is a noun or a verb.

This software provides what the user needs in a software, the "functionality", since it presents as a final output the index words of the text along with their weights. However, the software is more likely to be used as testing software and not as a commercial product.

5.2. Current Implementation

In the previous section, we have seen an overview at the previous implementation done Daher (2002). This implementation was integrated in our software. How this integration was done and what is presented as new in our software? This is what we are going to see in this section.

An extra step was integrated in which the frequent sets of words that relate the index words of the text to be indexed to other words from the pre-requisite, are generated.

5.2.1. Collection of Input Data

First of all, the text to be indexed goes under the auto-indexer that generates index words based on the count of the words, the count of the stems of each word, and the spread of words along the text. Of course all the pre-processing phases mentioned in Chapter 3 are applied to the text first. The index words are put into a list of strings, which should be sent to the part that we implemented as an input.

After finishing with the text that we wish to index, it is time for collecting the pre-requisite lists of words that appear frequently in texts from the same category of the text to be indexed. The auto-indexer will run again taking as an input, each time one of the pre-requisite texts.

At least 30 texts are needed to have better results (the more pre-requisite words we have the better the result is). A new list of strings is created that will contain the

index words with the highest weights of each pre-requisite text. Every text chosen from the same category of the text to be indexed will undergo the steps of the software implemented in Daher (2002) and at the end we will have the candidate words weighted. The top five words with the highest weight will be inserted into the pre-requisite list mentioned before.

After finishing with all the pre-requisite texts, the inputs of our method are ready; we have the list that contains the selected index words out of the text to be indexed along with the list of pre-requisite words.

Now we can move to the new step that we integrated and that is considered to be the implementation of our proposed solution.

A new button is added to the auto-indexer of Daher (2002) called "index". We press this button to open a new module represented by a screen in which all our work we have proposed is implemented.

When pressing the button, the list that contains the five index words of the text to be indexed as well as the list containing the pre-requisite indexes are passed as inputs to the new module.

5.2.2. Extracting Frequent sets

All the work that we have done to enhance the auto-indexing mechanism is based on the frequent sets extraction and the association of words. As we mentioned in Chapter 4, we chose to use the implementation of the Apriori algorithm to be able to extract the sets of the pre-requisite words that appear frequently together; and therefore, related to each other.

The five index words of every pre-requisite text form one transaction. The total is a number of transactions equal to the number of pre-requisite texts.

Apriori is applied to these transactions, which will produce a collection of sets, each containing words that are associated frequently together in texts from the same category of the text to be indexed.

These words will expand the number of relevant words that can be considered to be indexes of the text to be indexed. This is the purpose of our work; we are trying to maximize the number of candidate words to be considered as index words and that are not chosen when only considering the count of the words and their stems as well as the spread of the word. In the results of the experiments in which the implementation was tested, we noticed a good improvement. The experiments as well as the details will be explained in details in the following chapter.

5.2.3. Power sets

We need to pay attention to the fact that not all the sets that Apriori produces are useful. That is why we need to filter these sets and only choose what is needed. What is needed is what is related to the index words extracted out of the text to be indexed, or those that are related to a combination of the index words of the text to be indexed. We chose the implementation of the power-set algorithm to extract the combinations of the index words of the text to be indexed.

But first, what do we mean by power sets? We will start by taking the following example that will clarify things up: if we have the following elements: {x}, {y}, {z} and we need to form all the power sets out of these elements, the power set will be {{x}, {y}, {z}, {x, y}, {x, z}, {y, z}, {x, y, z}}.

The power sets algorithm is applied to the list containing the indexes of the text to be indexed. This will produce all the combinations possible of the index words chosen from the text to be indexed.

5.2.4. Choosing the relevant Frequent Sets

All the frequent sets returned by Apriori that contain an index word or a combination of index words are considered to be relevant and returned to the user as lists of strings. If we have a frequent set that does not contain any of the index words of the text to be indexed then it should not be considered to be related to the text we wish to index. Therefore we do not have to consider this frequent set as relevant. This is because we aim to expand number of index words of the text to be selected using a previous method by relating them to new words. If we have sets that do not contain any of the index words of the text to be indexed, then these sets are not related to the text to be indexed.

This will give the user more relevant words that are related to the index words given by the software proposed by Daher (2002) and will maximize the number of possible index words.

Since we used the software done in Daher (2002) and integrated a new module to it, the same data structure of this software used is mentioned in the previous paragraph, with the lists of strings containing the pre-requisite words and the index words of the text to be indexed.

5.2.5. Functionality

The extra module represented by a new screen was implemented in a user-friendly manner. However, this software is used more for testing and not for commercial purposes.

The main target of this software is to test whether the integration of the use of frequent item sets and association rule learning gives better results in the issue of auto-indexing Arabic texts.

In the following chapter, we will present our experimental results and analyze them to prove the usefulness of our proposed method.

5.3. Case studies

First of all, we selected 37 arbitrary texts from the same category of "استخراج النفط". The category is specified since we need to have access pre-requisite text from the same category of the text to be indexed to help achieve the work based on relations of words. The texts have different length so we can have various and not similar case studies on which we test the validity of our approach. The variety of scenarios make the results more reliable and trustable.

The texts of the chosen category were used as pre-requisite texts needed in our approach in order to extract the frequent item sets of words related to each other to help extend the sets of relevant index words. Notice that these index words are selected based on the frequency of words, their stems, as well as the spread of the words along the text to be indexed.

The experiments were conducted on 25 texts selected arbitrarily out of this category.

5.4. Calculations

The experiments are based on calculating two factors used to validate the usefulness of the approach proposed in this thesis. These factors are: the Precision and the Recall. The Precision measures the percentage of relevant index words retrieved programmatically out of all the words retrieved as candidate index words whether relevant or irrelevant. The Recall measures the percentage of relevant words retrieved programmatically out of the index words retrieved manually. This means it measures the completeness of retrieval of relevant index words.

First, we will start by defining some terms used in the calculations:

- N: Total number of words in a text.
- I: The most useful (relevant) index words found by manual indexing.
- RR: Retrieved Relevant index words using the software we implemented.
- RI: Retrieved Irrelevant index words using the software we implemented.
- NRR: None Retrieved Relevant words. NRR = I – RR.
- Precision = RR / (RR + RI).
- Recall = RR / (RR + NRR) = RR / I.

Any word that is returned as an output by the automatic auto-indexer, as well as the manual indexing, is considered to be relevant. Others that are returned programmatically but not manually are irrelevant. As we explained before, our approach retrieves relations between words presented as item sets. Each word appearing in an item set with a relevant word is checked to be relevant as well. If it's selected by the manual indexing, therefore it's relevant.

The calculations of Precision and Recall will be affected by the new relevant words returned as an output by our software.

The results are shown below in Table 5.1 and will be analyzed in the following section:

Table 5.1. Experimental Results

Results							
Text#	N	I	RR	RI	NRR	Recall	Precision
1	413	71	57	17	14	0.80	0.77
2	443	90	64	19	26	0.71	0.77
3	459	84	62	19	22	0.74	0.77
4	466	92	58	21	34	0.63	0.73
5	475	95	73	18	24	0.77	0.80
6	477	89	69	24	20	0.78	0.74
7	484	103	83	20	20	0.81	0.81
8	488	71	57	20	14	0.80	0.74
9	502	82	66	33	16	0.80	0.67
10	508	87	68	21	20	0.78	0.76
11	509	75	53	27	22	0.71	0.66
12	519	88	64	27	24	0.73	0.70
13	531	92	76	31	16	0.83	0.71
14	584	80	67	34	13	0.84	0.66
15	592	91	61	36	28	0.67	0.63
16	613	86	71	36	15	0.83	0.66
17	643	97	63	44	34	0.65	0.59
18	680	116	87	29	31	0.75	0.77
19	799	95	80	39	15	0.84	0.69
20	856	118	90	59	28	0.76	0.6
21	888	130	98	59	32	0.75	0.62
22	890	120	91	64	29	0.76	0.59
23	981	155	113	74	42	0.73	0.60
24	1157	202	126	66	76	0.62	0.62
25	1349	183	155	121	28	0.85	0.56
Average					**25.72**	**0.76**	**0.69**

5.5. Results Analysis

The results in table 5.1 show that our approach retrieves 69% of relevant index words out of the whole number of words selected; this is the Precision. Also it retrieves 76% of the words manually determined as index words, represented by the Recall.

We compared our results to the results calculated if the software implemented by Daher (2002) as auto-indexer is used alone, without taking into consideration the

relations between words. The Recall was equal to 70% in the previous work, the Precision was equal to 67%.

This means that the integration of the relation of words feature enhanced the recall by 6% and the precision by 2%.

Also, the NRR value was reduced by approximately 17.35%. This means that the number of non-retrieved relevant words was reduced by our method which is considered a very important enhancement.

After presenting the results of our proposed solution, and comparing them with the results of the work done previously, we can say that the use of associations of words and frequent sets of terms related to each other, improved the work done in the field of auto-indexing Arabic texts in a good way.

Some of the relevant words that do not appear frequently in the texts or that might not be spread along the text, but are somehow related to the subject of the text are now associated with other relevant words in the texts; and therefore, these words are considered by our approach as relevant.

This was not the case in any of the previous work related to the auto-indexing topic. The number of relevant candidate terms selected is increased; therefore, the user will have more variety of words related in a more efficient way to the subject of the text he's working on, which allows him to extract the subject of the text more accurately.

Now we can fairly say that choosing a subject for a certain text is proved to be more reliable after using the relations between words feature, presented in the proposed solution of this thesis and implemented by our work.

Chapter Six

Conclusion

To conclude, we need to mention that the auto-indexing mechanism is very important and can be used in many fields such as information retrieval.

We should not forget that the language has a huge impact on the difficulty of developing any auto-indexing software.

In our thesis we proposed a solution for the “auto-indexing Arabic texts” problem. The proposed solution is based on Association Rule Data Mining, or in other words, relating terms that frequently appear together in texts of the same category of the text to be indexed. This solution depends on pre-requisite texts from which the highest weighted index words are extracted and put into transaction, each corresponding to a text.

These transactions are given as an input to the Apriori algorithm that returns as an output frequent sets containing words related to each other.

This will increase the number of relevant words extracted out of the text. Some words that were not considered to be relevant when using the software, that only considers the frequency of the words and their stems as well as the spread of the word in the text, will be considered relevant by our work.

The association and relation of words are considered to be a new idea introduced to enhance the auto-indexing work proposed before particularly by Daher (2002).

In fact, the proposed solution by this report can be considered as a major innovative contribution to any auto-indexing system that deals with Arabic texts.

Three appendices will be submitted with this report. Appendix A contains the stop-list words. A case study is presented in Appendix B as well as the analysis. Appendix C describes the structure of the CD accompanying the submitted report.

At the end the references are also presented.

6.1. Further Work

The work done in this report can be considered to be a big improvement in the field of auto-indexing; however, new enhancements can be introduced. New proposed enhancements will be proposed in the following sections.

6.1.1. Extracting Stem Words

Many changes can be applied to the stemming algorithm proposed in Chapter 3. In the Arabic language, we have a very big number of grammatical rules. The more we integrate rules to the stemming algorithm, the better the stemming will be.

First, in the Arabic language we have certain symbols called 'الحركات' that change the pronunciation of the words. According to these symbols, the decision whether the word is a noun, a verb or a stop-list term can be done. The stemming will be better if the algorithm takes into consideration the 'حركات'.

Second, the more we add words to each of the M-derivations, T-Derivations and irregular plural tables, the better the stemming is.

Other enhancements can be done to the stop-list. This is by adding more words to it that shouldn't be taken into consideration in the indexing mechanism. We should make sure that none of the pointer nouns ('إسـم إشـارة'), discrete pronouns ('الضمائر المنـفصلة') and attaching nouns ('إسـم موصـول') is missing. Also the number should be added to the stop-list terms.

In addition, the words that belong to the list of 'الأسمـــاء الممنـــوعة من الصـــرف' should not be stemmed. Such words are names of countries, cities, human beings, mountains, rivers, seas, planets and many others. Numbers should be added to this list as well.

Also, numbers should be considered as stop-list terms and shouldn't be considered in the indexing algorithm.

6.1.2. Thesaurus Integration

Any auto-indexing algorithms cannot be considered as complete unless it is associated with a thesaurus system. The thesaurus has a good impact on any auto-indexer because it links different phrases or words with the same meaning to each other, which improves the indexing mechanism. Let us take the following example: consider having in the same text document the phrase '*رئيس البرلمان*' and the phrase '*رئيس مجلس النواب*'. Both phrases mean '*Head of Parliament*'; therefore, the occurrence of '*Head of Parliament*' in the text should be equal to both '*رئيس البرلمان*' and '*رئيس مجلس النواب*'. An auto-indexer that does not use the thesaurus will miss this feature. Also, the thesaurus can be considered to be relating words as well as phrases together, and using it can contribute enormously to the performance of an auto-indexer. The integration of thesaurus in any auto-indexing system is a very important task that ought to be taken into consideration in any further work regarding the auto-indexing field.

6.1.3. Enhance the proposed solution

We need to mention some of the enhancements that can be applied as further work to our proposed solution.

First, the more pre-requisite text we have the better the results are; this is due to the fact that we will have more relations between words; therefore, the number of new relevant words will increase. With the increase of the number of relevant index words extracted programmatically, the Precision and Recall will also increase. We propose to have a database of predefined lists each related to a certain category of texts. These lists are integrated into the system similarly to the stop-list terms and updated every now and then using a separate computer program. This is to prevent repeating the extraction of pre-requisite words every time new texts from the same category are indexed.

Second, as we explained before, we take the top five highest weighted index words out of the text to be index and out of the pre-requisite. We assume that if we maximize the number of extracted index words, the results will be better, because this will maximize the possibility of having more frequent sets of words related together. The more we have sets of words appearing frequently together, the tighter the relations between the words are. This will make our work more reliable and efficient because the relations between the words are stronger and more accurate.

References

Abdelali, B. & Tlili-Guiassa, Y. (2012). Meaning representation for automatic indexing for automatic indexing of Arabic texts. *International Journal of Computer Science Issues, 9*(6), 173-178.

Al-diabat, M. (2012). Arabic text categorization using classification rule mining, *Applied Mathematical Sciences, 6,* 4033 - 4046.

Al-Harbi, S., Almuhareb, A., & Al-Thubaity, A. (2008). *Automatic Arabic text classification*. Paper presented at the 9es Journées Internationales d'Analyse Statistique des Données Textuelles, Saudi Arabia (pp. 77-83).

Al-Molijy, A., Hmeidi, I., & Alsmadi, I. (2012). Indexing of Arabic documents automatically based on lexical analysis. *International Journal on Natural Language Computing,1*, 1-8.

Alsaleem, S. (2011). Automated Arabic text categorization using SVM and NB. *International Arab Journal of e-Technology*, *2,* 124-128.

Al-Shammari, E., & Lin, J. (2008). A novel Arabic lemmatization algorithm. *Proceedings of the second workshop on Analytics for noisy unstructured text data.* USA. Doi:10.1145/1390749.1390767

Bhalodiya, D., Patel, K., M., & Patel, C. (2013). An efficient way to find frequent pattern with dynamic programming approach. *Proceedings of the 2013 Nirma University International Conference on Engineering (NUiCONE), Ahmedabad,* 1-5. Doi:10.1109/NUiCONE.2013.6780102

Billhardt, H., Borrajo, D., & Maojo, V. (2000, April). Using term co-occurrence data for document indexing and retrieval. Paper presented at the proceedings of BCSIRSG 22nd annual colloquium on information retrieval research (England), Sidney Sussex College, Cambridge (pp. 105-117). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.214.6912&rep=rep1&type=pdf

Daher, W. (2002). *An Arabic auto-indexing system* (Unpublished Master's thesis). Lebanese American University, Lebanon.

Deeb, E. (1971). *New Arabic Grammar*. Beirut, Lebanon: Lebanese Book Publishing.

Dumais, S. (1998). Using SVMs for text categorization [Abstract]. Retrieved from: https://www.google.com.lb/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0CDIQFjAC&url=http%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fum%2Fpeople%2Fsdumais%2Fieee98-tc.doc&ei=SjvwVNObMc_OaIuXgrAP&usg=AFQjCNE0ufn62Qx7OpiUevgoCLdvle_fLQ&bvm=bv.87269000,d.d2s

El-Shishtawy, T., & El-Ghannam, F. (2012). An accurate Arabic root-based lemmatizer for information retrieval purposes. *International Journal of Computer Science Issues(IJCSI), 9*(1), 58-66. Retrieved from http://arxiv.org/ftp/arxiv/papers/1203/1203.3584.pdf

Gawrysiak, P., Gancarz, L., & Okoniewski, M. (2002). Recording word position information for improved document categorization. Retrieved from http://bolek.ii.pw.edu.pl/~gawrysia/publ/taipei.pdf

Gillman, P. (1990). Text retrieval: the state of the art: *Proceedings of the Institute of information scientists text retrieval conferences: The user's perspective (1988) and Text management (1989).* London: Taylor Graham.

Harrag, F. & El Qawasmeh, E. (2009). Neural network for Arabic text classification. P*roceedings of the Second International Conference on the Applications of Digital Information and Web Technologies, London,* 778 - 783. Doi: 10.1109/ICADIWT.2009.5273841

Hattab, A., & Hussein, A. (2012). Arabic content classification system using statistical Bayes classifier with words detection and correction. *World of Computer Science and Information Technology Journal, 2,* 193-196.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features [Abstract]. *In proceedings of the European Conference on Machine Learning. Berlin.* Retrieved from http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

Khoja, S. (2001). APT: Arabic part-of-speech tagger. Retrieved from http://archimedes.fas.harvard.edu/mdh/arabic/NAACL.pdf

Kindery, A., Rajihy, F., & Shimry, F. (1996). *Arabic Grammar Book (in Arabic)*. Kuwait: Rissala Publishing.

Larkey, L. & Connell, M. (2001). Arabic information retrieval at UMASS in TREC-10. Retrieved from http://ciir.cs.umass.edu/pubfiles/ir-254.pdf

Mayfield, J., & McNamee, P. (1998). Indexing using both N-grams and words. Proceedings of the Seventh Text Retrieval Conference (TREC-7) (pp. 419 – 424). Gaithersburg, MD: National Institute of Standards and Technology. Retrieved from http://old-lipn.univ-paris13.fr/~rozenknop/Cours/MICR_REI/articles-RI/JHUAPL.pdf

McCallum, A., & Niagam, K. (1998). A comparison of event models for naïve Bayes text classification [Abstract]. *In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization* (pp. 41-48). Retrieved from http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf

Rahman, C., Sohel, F., Naushad, P., & Kamruzzaman, S. (2010). Text classification using the concept of association rule of data mining. Retrieved from http://arxiv.org/ftp/arxiv/papers/1009/1009.4582.pdf

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian approach to filtering junk E-mail [Abstract]. *In proceedings of the AAAI-98 Workshop on learning for Text Categorization* (pp. 55-62). Retrieved from http://www.searchforum.org.cn/dataflowgroup/Reading/2004Paper/SpamPaper/ContentandStatistical/A%20Bayesian%20Approach%20to%20Filtering%20Junk%20E-Mail.pdf

Sharef, B., Omar, N., & Sharef, Z. (2014). An automated Arabic text categorization based on the frequency ratio accumulation. *The International Arab Journal of Information Technology, 11,* 213-221.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods [Abstract]. In proceedings of SIGIR-99, 22nd ACM International Conference of Research and Development in Information Retrival, New York, (pp. 42-49). Retrieved from http://www2.hawaii.edu/~chin/702/sigir99.pdf

Appendix A

Stop List Terms/Phrases

The following list contains most of the common stop list terms that are available in Arabic. The terms are categorized according to their type Deeb (1971) as well as Kindery et al. (1996).

أحرف الجر
عن
من
إلى
على
فـي
بـ
لـ
كـ

أدوات النصب
أن
لن
إذن
كـي
لام التعليل
لام الجحود
حتى
فاء السببية

أدوات الجزم
لم
لمَا
لام الأمر
لام الناهية
إن
مَن
ما
مهما
أي
متى

شكَان
شتان
بطآن
آه
أوَه
أوَاه
أفٍ
واه
وا
ووي
بَجَل
قد
قط
بخ
زه
صه
مه
هَيت
هلمَ
هيا
حيَ
حيَهل
ويها
بلهَ
إيه
رويد
تيد
تيدح
آمين
أمامك
وراءك

الفاء
ثـم
حتـى
أو
أم
بـل
لـكن

أحرف القسم
الباء
التاء
اللام
الواو

أحرف النداء
يا
أيا
هَيَا
أ
آ
أي
آي
وَا
أيُها
أيَتها

أدوات الإستثناء
إلا
غير
سوى
عدا

سـاء
حبَذا

كان و أخواتها
كان
أمسى
أصبح
أضحى
ظلَ
بات
صار
ليس
ما زال
ما انفك
ما فتئ
ما برِحَ
ما دامَ

إنَ و أخواتها
إن
أنَ
كأنَ
لكنَ
ليتَ
لعلَ

كاد و أخواتها
كاد
كرب
أوشك
عسـى

إبتدأ

ليس و أخواتها
ما
لا
إن
لات
ليس
إلا

أسماء الكناية
كم
كأي
كذا
كيت
ذيت
بضعة
فلان
بمن

أسماء الأصوات
هيد
هاد
جه
ده
عاه
هس
هش
دج
حاحا
طق

كيفما
أينما
حيثما
إذما
أيان
أنَى
أين

أسماء أفعال
هيهات
سرعان

إسم الموصول
الذي
الذان
الذِينَ
التي
التان
اللواتي
اللائي

مكانك
ها
هاك
دونك
عندك
لديك
إليك
عليك بنفسك

أحرف العطف
الواو

أسماء الإشارة
هذا
هذان
هؤلاء
هذه
هاتان
ذلك
ذانك

خلا
حاشا

أدوات التحذير
إيَّاك
حسبك
حذار

أدوات المدح و الذم
نِعم
بِئس
ذلكم
تلك
تانك
تلكم
أولئك

الضمائر المنفصلة
هو

حرى
إخلولق
شرع
أخذ
أنشأ
جعل
طفِقَ
هبَ
قام
إنبرى
بدأ
هما
هم
هي
هما
هن
أنتَ
أنتُما
أنتُم

دو
هم
كخ

أحرف الجواب
نَعَم
أجَل
إي
بلَى
كلا
لا
أنتِ
أنتما
أنتنَ
أنا
نحن

Appendix B

Case Study

Consider the following Arabic document:

مليون برميل إنتاج البحرين من النفط

المنامة - الهيئة الوطنية للنفط والغاز

بلغ إجمالي إنتاج النفط في مملكة البحرين والذي يشتمل على (حقل البحرين وحقل أبوسعفة) للفترة مابين يناير/كانون الثاني و سبتمبر/أيلول الماضي، 49.445 مليون برميل، ويلاحظ، أن إنتاج النفط خلال شهر سبتمبر من حقل البحرين وحقل أبوسعفة هو الأفضل مقارنة مع معدل الإنتاج في الأشهر الأخرى من العام 2010.

وبلغ المعدل اليومي لإنتاج حقل البحرين خلال شهر سبتمبر من العام الجاري (2010) 32 ألف برميل وهو المعدل اليومي نفسه للإنتاج لشهر سبتمبر من العام الماضي (2009.(

جاء ذلك، في تقرير الإحصاءات النفطية الذي أصدرته الهيئة الوطنية للنفط والغاز، الخاصة بإنتاج النفط والغاز والواردات النفطية وكذلك المشتقات النفطية والمبيعات المحلية للفترة مابين يناير وسبتمبر/أيلول 2010.

ويلاحظ، أن معدل الإنتاج اليومي لشهر سبتمبر من العام 2010، أفضل من المعدل اليومي لشهر يناير من العام الجاري الذي بلغ 30 ألف برميل يومياً.

وفيما يخص النفط الخام المستورد من المملكة العربية السعودية فقد بلغ 64.253 مليون برميل في العام 2010، بالمقارنة مع 60.171 مليون برميل للفترة نفسها من العام 2009؛ أي بزيادة قدرها 4.082 ملايين برميل وهي تعادل 6.8 في المئة. وقد بلغ المعدل اليومي للنفط الخام المستورد 236 ألف برميل يومياً، مقارنة مع 220 ألف برميل يومياً خلال الفترة نفسها من العام 2009م. وتعود أسباب الزيادة في الكمية المستوردة من النفط إلى الطلب على المشتقات النفطية المكررة بمصنع التكرير بالمملكة. أما النفط الخام الذي تم ضخه إلى مصفاة التكرير خلال التسعة شهور من العام فقد وصل إلى 72.730 مليون برميل، بالمقارنة مع 69.751 مليون برميل خلال العام 2009؛ أي بزيادة نسبتها 4.3 في المئة.

ويتمثل النفط الخام الذي تم ضخه إلى المصفاة في النفط المستورد من المملكة العربية السعودية والذي يشكل 89 في المئة وكذلك المنتج من حقل البحرين والذي يشكل 11 في المئة. وقد بلغ المعدل اليومي للنفط الخام الذي تم ضخه إلى مصنع التكرير 266,410 براميل، مقارنة مع 255,501 برميل خلال الفترة نفسها من العام 2009.

وتعود أسباب الزيادة إلى زيادة إنتاج مصفاة البحرين لتلبية الطلب على المشتقات النفطية.

أما إنتاج مصفاة البحرين خلال التسعة شهور من العام 2010، فقد بلغ 74.313 مليون برميل مقارنة مع 70.721 مليون برميل خلال العام 2009، وبمعدل يومي قدره 272 ألف برميل، مقارنة مع 259 ألف برميل المعدل اليومي للفترة نفسها من العام 2009؛ إذ تبلغ نسبة الزيادة 5.1 في المئة.

إن المعدل اليومي لإنتاج المصفاة قد تجاوز الطاقة الاستيعابية للمصفاة التي تبلغ 260 ألف برميل في اليوم؛ إذ حققت مصفاة البحرين معدلاً قياسياً للإنتاج خلال الفترة من العام، ويعتبر هذا المعدل من الأرقام الكبيرة التي تحققها المصفاة في تاريخها.

أما المنتجات النفطية المصدرة، فقد بلغ المعدل اليومي للصادرات 236 ألف برميل، مقارنة مع 223 ألف برميل للفترة نفسها من العام 2009، وبلغ إجمالي الصادرات خلال التسعة شهور 64.547 مليون برميل، مقارنة مع 60.941 مليون برميل، بزيادة قدرها 3.606 ملايين برميل؛ أي أن الزيادة تعادل 5.9 في المئة. وتعود أسباب الارتفاع في الصدرات إلى الطلبات الخارجية على المشتقات النفطية التي تنتجها مملكة البحرين من مصفاة التكرير؛ إذ تنتج المصفاة عدة أنواع من المشتقات النفطية تمتاز بجودتها العالية المطابقة للمواصفات العالمية. وهذه الأنواع هي: النفثا، الغازولين، الكيروسين، وقود الطائرات، الديزل، زيت الوقود، الكبريت بالإضافة إلى الأسفلت.

أما فيما يخص المبيعات المحلية فقد بلغ المعدل اليومي 26 ألف برميل خلال التسعة شهور من العام 2010، مقارنة مع 25 ألف برميل خلال الفترة نفسها من العام 2009؛ أي بزيادة تصل نسبتها إلى 4.5 في المئة. وهناك عدة أسباب لهذه الزيادة، وأن أهمها تطور حركة السفر عن طريق جسر الملك فهد؛ إذ ساهم الجسر في زيادة حركة السياحة العائلية الخليجية وبالتالي زيادة السيارات القادمة إلى مملكة البحرين عبر الجسر؛ فضلاً عن حركة العمران التي تشهدها المملكة؛ ما ساهم في زيادة استهلاك المشتقات النفطية محلياً، إضافة إلى ارتفاع عدد السيارات البحرينية في المملكة.

وفيما يتعلق بمبيعات مطار البحرين الدولي من وقود الطائرات خلال العام 2010، فقد بلغ المعدل اليومي 14 ألف برميل يومياً مقارنة مع المعدل نفسه خلال العام 2009، وقد بلغ الإجمالي 3.723 ملايين برميل، وتشرف على هذه المبيعات شركة البحرين لتزويد وقود الطائرات (بافكو.(

إنتاج واستهلاك الغاز

بلغ المعدل اليومي لإنتاج الغاز في مملكة البحرين خلال التسعة شهور 1.544 مليار قدم مكعب، مقارنة مع 1.497 مليار قدم مكعب خلال الفترة نفسها من العام 2009. أما إجمالي إنتاج الغاز فقد بلغ 421.439 مليار قدم مكعب خلال التسعة شهور من العام الجاري، مقارنة مع 408.757 مليارات قدم مكعب خلال الفترة نفسها من العام 2009، بزيادة قدرها 12.682 مليار قدم مكعب، وهي تعادل ما نسبته 3.1 في المئة. وترجع أسباب الزيادة إلى الطلب على الغاز من قبل المحطات الكهربائية في المملكة والتي تستخدم الغاز في إنتاج الكهرباء، وكذلك استخدام الغاز في المصانع الكبرى في المملكة مثل شركة نفط البحرين (بابكو) وشركة الخليج لصناعة البتروكيماويات (جيبك) والصناعات الأخرى.

ويأتي استهلاك هيئة الكهرباء وشركات الكهرباء الخاصة الأخرى في المرتبة الأولى (أي بنسبة 35 في المئة)، تليها شركة ألمنيوم البحرين (ألبا) ثم شركة نفط البحرين (بابكو) وشركة الخليج لصناعة البتروكيماويات (جيبك) وعدد من الشركات الصناعية العاملة في مملكة البحرين. ويستخدم الغاز كلقيم لتوليد الطاقة وإعادة حقن الآبار النفطية وللأغراض الصناعية الأخرى.

The number of words in this text Document denoted by N is 799 words.

The top five index words with the highest weights after running the software implemented by Daher (2002) are:

بحرين
عام
مقارنة
نفط
برميل

The relevant sets of words related (associated) frequently together are:

نفط	برميل 5	نفط حقل عام
برميل	برميل خاصة	نفط عام طاقة
بحرين	برميل فترة	نفط شركات طاقة
عام	برميل سبتمبر	نفط شركات والت
نفط إنتاج	برميل ماض	نفط طاقة والت
نفط ألف	برميل مصدرة	نفط خاصة فترة
نفط مليون	برميل أخرى	برميل إنتاج ألف
نفط خام	إنتاج عام	برميل إنتاج مليون
نفط ملايين	بحرين يومياً	برميل إنتاج خام
نفط بحرين	بحرين حقل	برميل إنتاج سعود
نفط يومياً	بحرين زيادة	برميل إنتاج سبتمبر
نفط خليج	بحرين غاز	برميل ألف مليون
نفط دول	حقل عام	برميل ألف ملايين
نفط عالم	عام طاقة	برميل مليون خام
نفط تشهد	نفط إنتاج مليون	برميل مليون ملايين
نفط حقل	نفط إنتاج خام	برميل مليون 5
نفط عام	نفط إنتاج يومياً	برميل خام ملايين
نفط شركة	نفط إنتاج عالم	برميل دول خاصة
نفط زيادة	نفط إنتاج حقل	برميل دول فترة
نفط شركات	نفط إنتاج عام	برميل عالم عام
نفط طاقة	نفط إنتاج سعود	برميل عالم ماض
نفط والت	نفط إنتاج أخرى	برميل عالم مصدرة
نفط سعود	نفط ألف ملايين	برميل زيادة سعود
نفط 3	نفط مليون خام	برميل خاصة فترة
نفط منتج	نفط مليون ملايين	برميل ماض مصدرة
نفط خاصة	نفط مليون زيادة	إنتاج حقل عام
نفط فترة	نفط مليون 3	بحرين حقل غاز
نفط ماض	نفط مليون منتج	نفط إنتاج حقل عام
نفط مصدرة	نفط خام ملايين	نفط مليون خام ملايين
نفط أخرى	نفط خام عالم	نفط مليون زيادة 3
برميل إنتاج	نفط ملايين منتج	نفط خليج دول عالم
برميل ألف	نفط بحرين يومياً	نفط خليج دول تشهد
برميل مليون	نفط خليج دول	نفط خليج عالم تشهد
برميل خام	نفط خليج عالم	نفط دول عالم تشهد
برميل ملايين	نفط خليج تشهد	نفط دول خاصة فترة
برميل دول	نفط دول عالم	نفط عالم ماض مصدرة
برميل عالم	نفط دول تشهد	نفط شركات طاقة والت
برميل حقل	نفط دول خاصة	برميل مليون خام ملايين

برميل عام	نفط دول فترة	برميل دول خاصة فترة
برميل شركة	نفط دول أخرى	برميل عالم ماض مصدرة
برميل زيادة	نفط عالم تشهد	نفط خليج دول عالم تشهد
برميل سعود	نفط عالم عام	
برميل مئة	نفط عالم ماض	
برميل 3	نفط عالم مصدرة	

Apparently, we have in some of the sets that relate index words of the text to be indexed with other words numbers. This is because the numbers are considered to be index words for some of the pre-requisite texts. We mentioned before that it would be better if the numbers are considered as stop-list terms.

Also, all the frequent sets contain at least one of the index words of the text to be indexed, and they are related to other words as well. We consider this to be very helpful in the indexing mechanism. We chose the index words with the highest weights out of the text to be indexed. We considered that these are the words that represent the text the most, due to the fact that based on their count, the count of their stems, and their spread along the text. These index words were the most weighted. We did not stop at this stage; to enhance the work, we selected other words that are related frequently to these words. Even if the new words presented by the frequent sets aren't chosen based on the factors considered by Daher (2002) to find the index words of a text; these new words are selected by our work what makes the possibility of finding the most accurate index words higher.

Appendix C

Accompanying CD

The CD accompanying this report contains the following directory structure:

- **Report –** Contains two folders.
  - **Document Files –** Ten Word documents where this report resides.
  - **Sample Docs –** Two samples of Arabic documents for testing.
- **Software –** Contains four folders.
  - **Source Code –** Source code (Visual Basic code) for the testing software.
  - **Setup 9x –** A setup program for 9x machines.
  - **Setup NT –** A setup program for NT machines.
  - **DLLs –** containing the dlls of the implementation of the proposed solution
- **References –** Contains three folders.
  - **Association rule based data mining –** Three pdf papers and two power point presentations.
  - **Miscellaneous Papers -** Sixteen papers in pdf format.
- **Presentation –** PowerPoint presentation.