Diacritic Oriented Arabic Information Retrieval System

Khalid S. R. Aloufi koufi@taibahu.edu.sa
College of Computer Science and Engineering,
Taibah University,
Madinah, KSA

Abstract

Arabic language support in search engines and operating systems is improved in recent years. Searching in the Internet is reliable and can be compared to the excellent support for several other languages, including English. However, for text with diacritics there are some limitations. For this reason, most Information retrieval (IR) systems remove diacritics from text and ignore it for its complexity. Searching text with diacritics is important for some kinds of documents, such as those of religious books, some newspapers and children stories. This research shows the design and development of the system that overcome the problem. The proposed system considers diacritics. The proposed system includes the design complexity in the retrieving algorithm rather than the information repository, which is database in this study. Also, this study analyses the results and the performance. Results are promising and performance analysis shows methods to enhance design and increase the performance. The proposed system can be integrated in search engines, text editors and any information retrieval system that include Arabic text. Performance analysis of the proposed system shows that this system is reliable. The proposed system is applied on database of Hadeeth, which is religious book includes the prophet action and statements. The system can be applied in any kind of data repository.

Keywords: Search Engine, Information Retrieval Systems, Diacritics, Arabic Language, information Retrieval Performance Analysis

1. INTRODUCTION

In the early days of the Internet Arabic letters were not defined. However, when the Unicode standard becomes applicable in the Internet browsers and the hyper text markup language (HTML), several non Latin languages have been supported.

Arabic section in the Hex reference of the Unicode standardization that include characters, such as letters, symbols and diacritics Ranges from 0600 to 06FF [6].

The Arabic letters in Unicode range from 0621 to 063A and from 0641 to 064A [6]. The Arabic diacritics range from 0618 to 061A and from 064D to 0656 [6]. In Unicode, There are 36 Arabic letter and 11 diacritics.

Every spoken language has its own features that are applied in reading, writing, listening and speaking. Arabic language is root based language. Arabic language consists of 28 letters. Each letter is pronounced with different vowels depending on the diacritics used. Some of the 28 letters are written with different shapes depending of their order in the word. These features are important for information retrieval systems, either for sound processing or pattern recognition. This study develops an information retrieval methodology for Arabic language.

The features of the any spoken language increase the information find-ability when considered in those concerned information retrieval systems. Any language with diacritics can get applied by the result of this study. The language features is included in the information systems analysis and design to increase the retrieval reliability.

There are present difficulties for Arabic users in searching for information in the Internet using search engines [2]. The amount of information returned is far less than available.

Current search engines are failing to support non Latin languages in the same level of Latin language support [10]. The design of search engine philosophy considers the features of Latin languages but it is not working with non Latin languages such as Arabic [2].

There are different methodologies for AIRS. Methodologies can be classified in two categories. The first category makes no change to the text. The second category removes all diacritics from the text.

The second category cannot be used with text sources that require being with diacritics. Diacritics are important from some sources because changing diacritics for similar words might change the meaning. Usually, as mentioned earlier this is important for some text sources, such as religious and educational books for children.

Compared to the second category, the first category increases the complexity of database design and the retrieval algorithm. However, this requires further investigation and AIRS development. This study keeps the diacritics as in the original document as in the first category of the methodologies. The complexity of database design is out of the scope of this study. The database mainly is the repository of the information of Arabic text with diacritic. The main contribution of this research is to investigate the basic approach of retrieving diacritized text.

The methodology and approach of this study was not found to be applied by any study according to the knowledge of this research. However, several studies removed diacritics from text, which applied in information retrieval system of Quran [7] [1] [9] [10].

The paper is organized as follows. The next section describes the system model. Third section is the section for discussing the results. The paper finding and further research suggestions are summarized in the section Conclusion and Future work. This paper ends with the references used in this research.

2. THE SYSTEM MODEL

This section presents the Arabic information retrieval system (AIRS). This system uses the Database as the information repository. Database is one of the examples of data repositories used by search engines [11].

The SQL is the language that supports reading, writing and other functions with databases. However, SQL is found to not fully support non Latin languages, such as Arabic [4]. The SQL commands such as LIKE and regular expression command does not support diacritics and multi byte characters [4]. For this reason this study propose AIRS simplified model.

The system model consists of four processes, which are input processing, input terms, searching and result set.

Input processing is the process of defining the characters of word. The proposed model assumes single word without diacritics. Usually the user is expected to enter the word without diacritics. Input terms are the step of the definition of all the possible words with diacritics. Searching is the SQL query to search for all the words. Result set is the last process to return the group of records from the database.

The following algorithm shows the above processes:

Variables:

Term: entered by user

Generated terms: contains words that will be forwarded to the search engine.

The algorithm:

1. *User enters the Term.*
2. *Generate all possible Terms*
3. *Search for generated term*
4. *Return result set for all generated terms*
5. *finish*

Assuming the user will enter the word $x_1x_2x_{3,}$ the algorithm execute as follows:

1. User enters the term $x_1x_2x_3$.
2. The generated terms will be $x_1x_2x_3,x_1\text{-}x_2x_3,x_1x_2\text{-}x_3$ and $x_1\text{-}x_2\text{-}x_3$.
3. Search for all the generated terms in step 2
4. The result set is returned
5. finish

The user enters a word without diacritics to search for. The system generates all the possible words with diacritics using the simple collection generation as shown in the algorithm. Any "-"is counted if it is any diacritics only because the system uses regular expression. After that, the system in step 3 search for the generated words in the database. Step 4 will return the search match of any of the generated terms in one query result without any repetition of the records returned in the result set. Searching is a result set of the SQL query for all the generated terms. SQL query replaces any dash "-"with any character, mainly for diacritics.

The searching algorithm is developed using JAVA programming language [3]. The database is designed using MySQL [5]. The operating system used is Windows 7 Home premium. Processor if Intel ® Core™2 Do CPU P9300 @ 2.26GHz 2.27GHz and the Ram is 3.00 GB. System type is 64-bit Operating System.

This research is applied as information retrieval of database of Hadeeth. The information of the Hadeeth can be organized according to the section found in most Hadeeth books.

Each Hadeeth consists of Sanad and Matin. The Hadeeth book in this study is mukhtasar Saheeh Muslim [8].

The Hadeeth book is organized in chapters, sections and items. Each item is single Hadeeth. Each Chapter consists of several Sections and each section contains a group of Hadeeth. Chapter is called Book or Ketab in Arabic. Section is called Door or Bab in Arabic.

3. RESULTS AND PERFORMANCE ANALYSIS

The section presents the experiments results and performance analysis and applied improvement methods. Each experiment returns a number of generated words, the number of returned results and the time required.

Each experiment includes the generated words, the number of the results returned for this word, the time required to get the result set. Table 1 presents one experiment example.

Table 2 summarizes all the experiments performed on AIRS in this study. The table shows the word, the number of letter for each word, the number of generated words, the waiting time, number of words without results, number of words with results and the total number of returned results.

Word	Number of Results
رسول	3
ر_سول رس_ول رسو_ل ر_سو_ل رس_و_ل ر_س_و_ل	0
ر_س_ول	1768
Time	1553 *ms*

TABLE 1: search for the word رسول

The mathematical notations used in this study are as follows. The number of generated words is n_t. The waiting time is w_t. The total number of returned records is n_r.

Equation (1) computes the time required to get a result for all generated words which have no results, where x_{nwrt} is a constant; n_t is the total number of generated words. From the experiments, x_{nwrt} is found to be 38.7 *ms* for each word without returned result.

Computing the time required to get a record is performed by Equation (2), where x_{nwr} is a constant, n_r is the total number of returned records. From the experiments, x_{nwr} is found to be approximately 0.6 *ms*. The waiting time can be calculated as a function of the time required to search for a result of a word and the time requiered to get a result.

The total waiting time is computed using equation (3), where n_r is the total number of returned records and n_t is the total number of generated words.

$$t_{nwrt} = x_{nwrt} . n_t, \text{ , where } x_{nwrt} = 38.7\ ms \quad (1)$$

$$t_{nw} = x_{nwr} . n_r \text{ , where } x_{nwr} = 0.6 \quad (2)$$

$$w_t = t_{nwr} + t_{nwrt} = x_{nwr} \cdot n_r + x_{nwrt} \cdot n_t = 0.6 \cdot n_r + 38.7 \cdot n_t \quad (3)$$

Total number of records	Number of words with results	Number of words without results	Waiting Time	Number of Generated Words	Number of letters	Word
1771	2	6	1553	8	4	رسول
2073	3	1	982	4	3	قال
22	1	7	382	8	4	خالد
0	0	16	722	16	5	هريره
451	2	14	872	16	5	هريرة
0	0	64	2353	64	7	المدينه
94	1	63	3217	64	7	المدينة
0	0	64	2117	64	7	الاسلام
53	1	63	2221	64	7	الإسلام
0	0	64	2110	64	7	ألإسلام
0	0	64	2790	64	7	ألاسلام
349	2	2	488	4	3	أحد
120	3	1	256	4	3	احد
5	1	3	165	4	3	صلى
1	1	3	167	4	3	صلي
0	0	8	286	8	4	شجره
23	1	3	209	4	3	عجل
13	1	7	296	8	4	شجرة
3	1	15	524	16	5	قيامه
89	1	15	614	16	5	قيامة
55	1	7	318	8	4	خرجت
0	0	4	164	4	3	حتى
4	1	3	165	4	3	حتي
486	3	1	486	4	3	إذا
49	2	2	217	4	3	اذا
0	0	64	2971	64	7	الايمان
17	1	63	2171	64	7	الإيمان
0	0	64	2150	64	7	ألإيمان

TABLE 2: Summery of a group of experiments

Figure 1 shows the time to search for words that has no result. Figure 2 shows the time required to return record from database. Figure 3 shows the time required for generated words when n_t is 1.

The comparison between figure 1 and figure 2 shows that time consumed in searching is much higher than the time required for getting the results from the information repository.

The comparison between figure 2 and figure 3 shows that the time required for getting the results increases as the number of returned results increases without regards of the number of letters of the word.

Figure 4 shows the time required for generated words when n_t is 8. Figure 5 shows that the w_t increase by a factor of searching time, which is 38.7 *8. Figure 5 shows the time required for searching and getting results of generated words when n_t is 8.

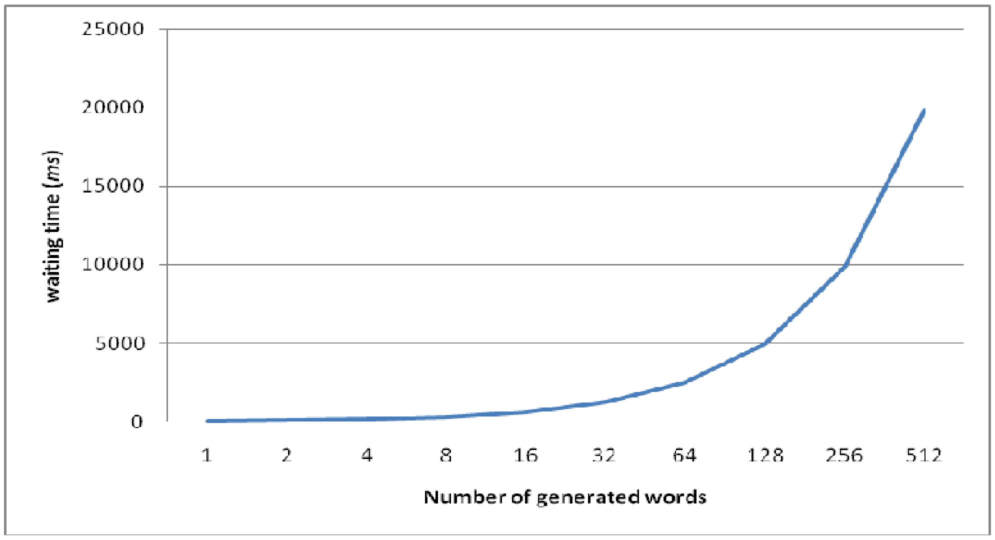

FIGURE 1 : the time required to search for words that has no result.

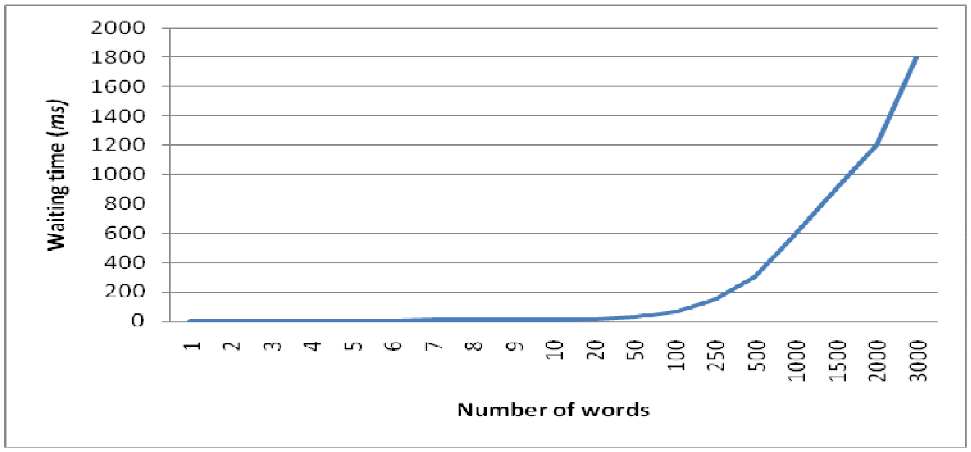

FIGURE 2 : the time required to return record from database.

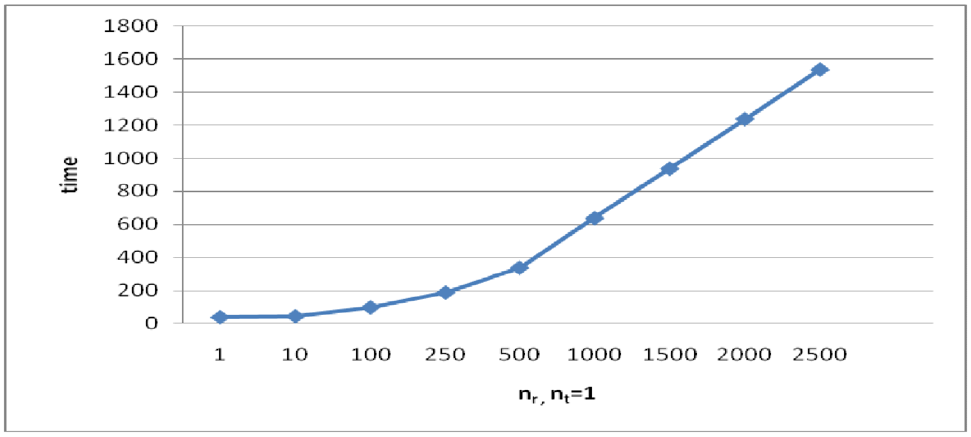

FIGURE 3 : The time required for generated words when n_t is 1.

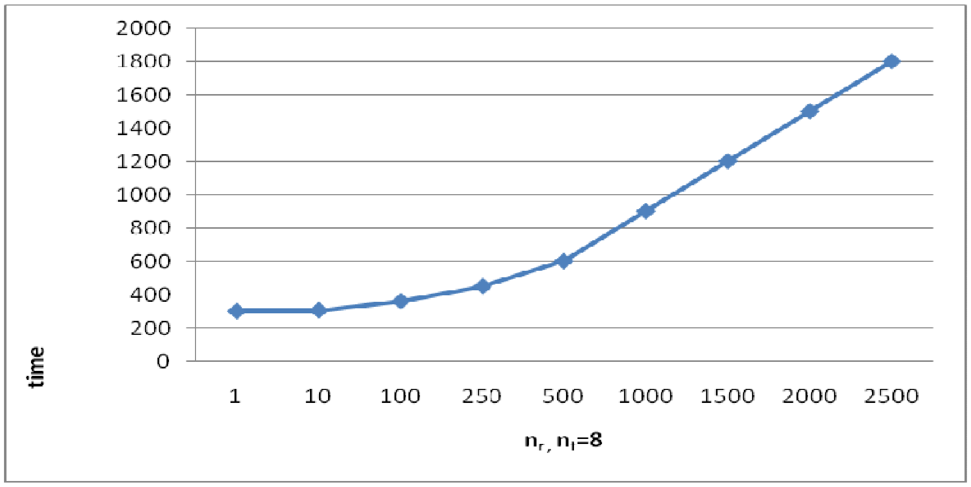

FIGURE 4 : The time required for generated words when n_t is 8.

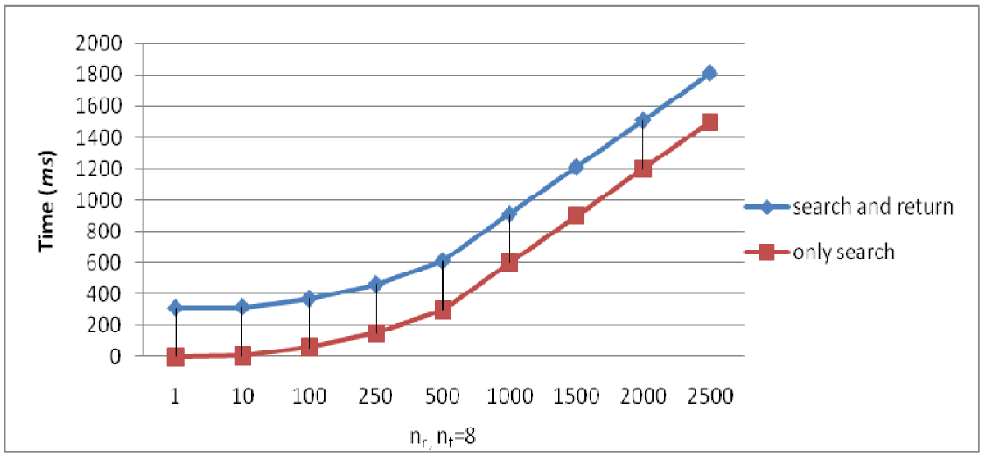

FIGURE 5 : The time required for searching and getting results of generated words when n_t is 8.

4. PERFORMANCE DEVELOPMENT

The system performance can be increased by decreases the waiting time when excluding the words that has no results. The time required to search for a word with no result is found to be 38.7 *ms.*

The average time to return one record is 0.605063 *ms.* In some cases, which is expected, some results were performed with high performance even with a large number of words without results. These results have low number of returned results and can be considered as words with no results as an assumption in this study.

According to Equation (3), there is a waiting time increase of 38.7 *ms* for each search. The worst case is small number of result or no result for the search.

The best case is where there is a group of results for each search. The small number of result can be defined as the number of results with waiting time close to the waiting time for searching for the same number of generated words with no results.

According to figure 5, there is 8 generated words. The waiting time increases as the number of returned words increases. The increase wating time is constant becasuse of searching is constant with 8 * 38.8.

Figure 6 shows that the percentage of the time used for searching is decreasing as the number of the returned results increases. This can lead to a conclusion that the percentage of searching time is much less than the returning time when the returned results is increasing. However, when the returned results are decreasing, most of the waiting time is in searching.

When n_t is 512 words , figure 7 shows that more than 90 % of the waiting time is searching either for large or small number of returned results.

Earlier figures of figure 3 and figure 4 shows that the waiting time increases as the number of generated words increases. By equation (1), figure 8 shows exponential increase of wating with the increase of the number of genreated words.

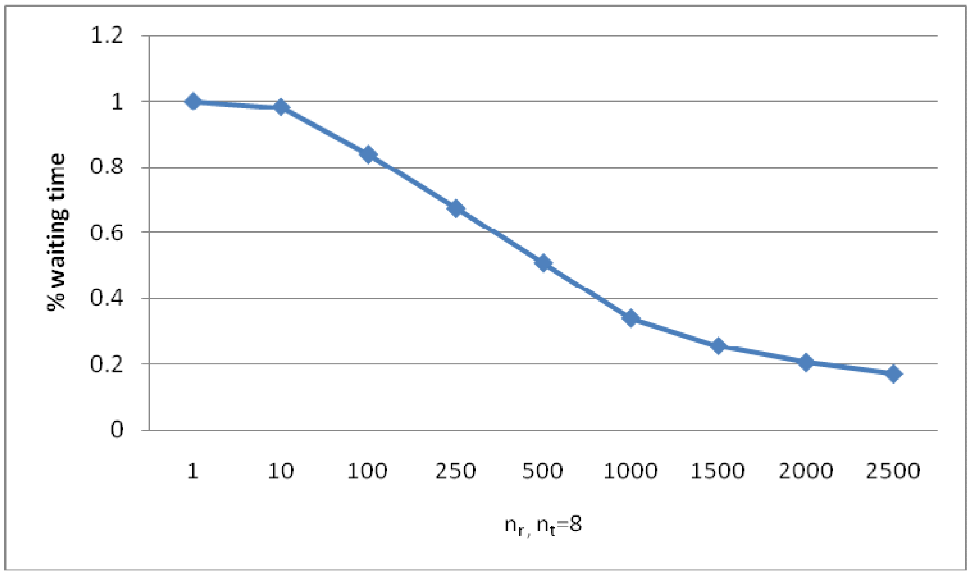

FIGURE 6: the % w_t for different n_r ,n_t=8

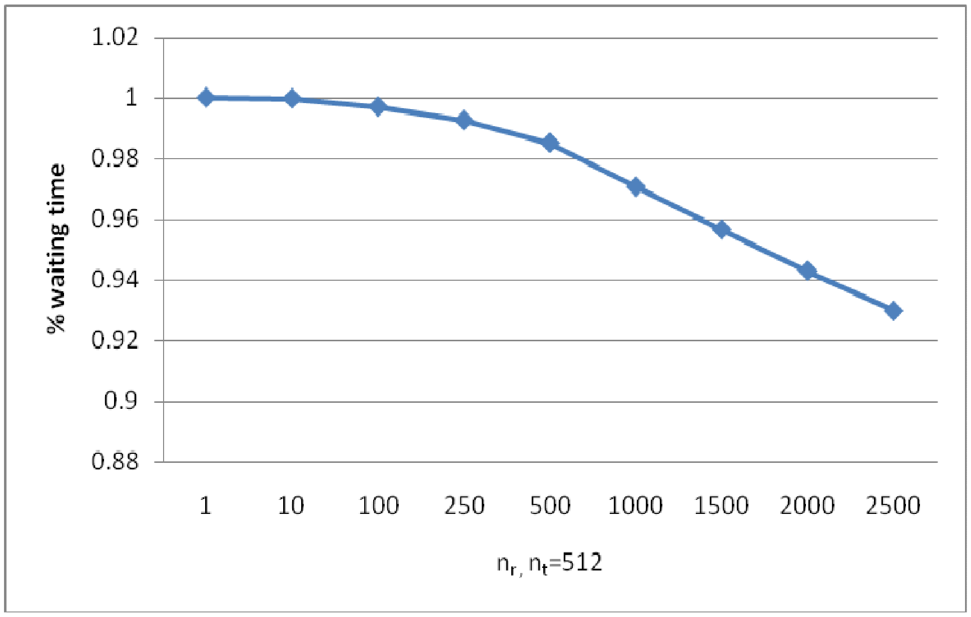

FIGURE 7: the % w_t for different n_r ,n_t=512

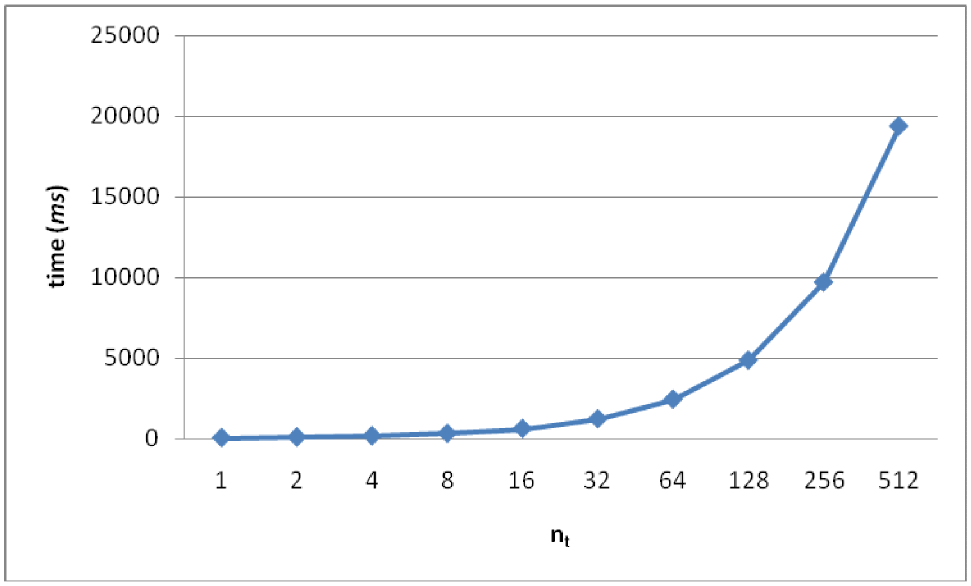

FIGURE 8: the w_t for different n_t

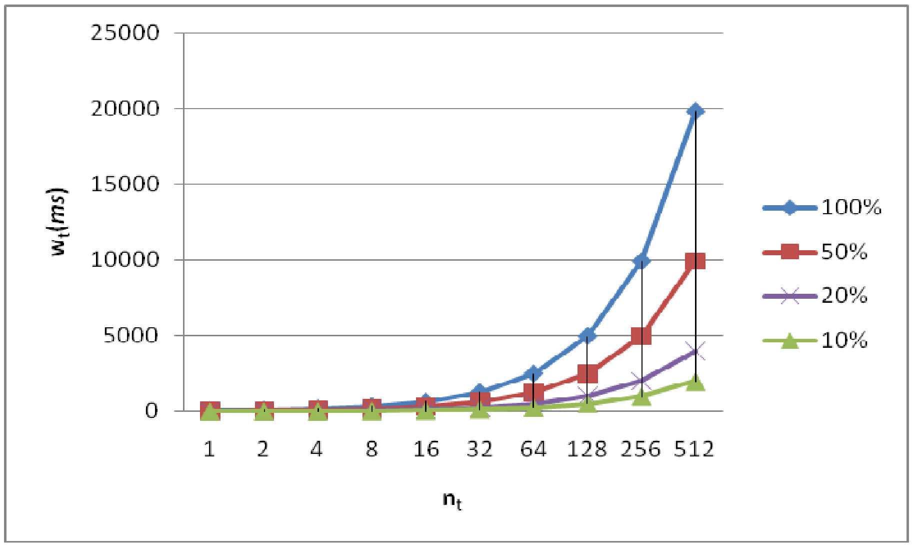

FIGURE 9: the w_t for different n_t

In conclusion, the percntage of waiting time used for search is increasing as the number of generated words increases. Not having every search return results, it is performance development advantage by excluding the words that have no results.

The system performance can be increased by having a list of words with no results to decrease waiting time. Figure 9 shows the w_t increases as the n_t increases. If 50%,20%,10% of the generated words has no results, 50%,20%,10% of the w_t is saved, consequently.

The detailed result is listed in table 3. The system will gain a great perfromance by exluding the words with no results. There is no tradeoff by using a table to list the excluded words with no results because this will be small table and time for searching is assumed to be zero time.

Furthermore in perfomance analysis, if a word has no result at all, the AIRS can be developed to exclude another table for words from the first step befor other steps befor generating other words. In general AIRS can be developed to work according the following algorithm:

Variables:

term: entered by user

t1: table of excluded terms

t2: table for exuded generate terms

Generated terms: contains terms that will be forwarded to the search engine.

The algorithm:

1. *User enters the Term.*
2. *If the term is in t1, then return no results and move to step 7*
3. *Generate all possible Terms*
4. *Remove any term found in t2*
5. *Search for generated term*
6. *Return result set for all generated terms*
7. *finish*

Number of letters	n_r	% w_t , 10% of n_r with results	% w_t , 20% of n_r with results	% w_t , 50% of n_r with results	% w_t , 100% of n_r with result
1	1	3.87	7.74	19.35	38.7
2	2	7.74	15.48	38.7	77.4
3	4	15.48	30.96	77.4	154.8
4	8	30.96	61.92	154.8	309.6
5	16	61.92	123.84	309.6	619.2
6	32	123.84	247.68	619.2	1238.4
7	64	247.68	495.36	1238.4	2476.8
8	128	495.36	990.72	2476.8	4953.6
9	256	990.72	1981.44	4953.6	9907.2
10	512	1981.44	3962.88	9907.2	19814.4

TABLE 3: % w_t , different percentages of n_r with results

5. PERFORMANCE ANALYSIS

Some words can be without meaning but returns results because it is part of other words. This is the result of using regular expressions. For example the word حتي is not meaningful in Arabic but because it can be part of other words, there were a result set for it. The word حتى is meaningful but there was no result set for it.

The search term, for example, قال has some of the possible generated words such as قّال , قال قُال , قّالْ , قالْ , قّال , قالْ , قال , قّال , قَالَ , قَال , قالَ , قَال , قالَ , قال , .. etc. There are 11 possible diacritics and this means this will results in 1331 different terms of قال. However, because the system uses regular expression, there will be only 4 words which are ق-ا-ل ,قا-ل ,ق-ال ,قال, where the dash or "-" is any possible diacritic.

Also, for performance reasons, some Arabic information retrieval system evaluates some Arabic letters to be equal, such as ا, أ, and إ.

Some words cannot be found in Arabic because the combination of diacritics does not exist and this is not a problem of processing because the system uses regular expression.

6. CONCLUSION AND FUTURE WORK

In conclusion, AIRS is a system model to retrieve information from Arabic text that includes diacritics. The system design, analysis and performance development is presented in this study. Results shows that the system overcomes the problem found in current information retrieval systems. Results show that the performance can be improved by extra algorithm steps.

The developed system assumes the user will not enter any diacritic. The user could select from the possible inputs to refine the search and the system considers the order of results. The system uses the discrete processing of input characters of the search term.

It is expected that some users may be interested in writing some diacritic for exact matches of the terms. The system can be developed for this purpose.

Future studies will include the addition of the stemming and lemmatization to the system, such that the searched term will be refereed to its lemma using a stemming engine. This will include more results for a search action. However, this will increase the number of terms that will be searched for each search action**.**

Future studies will include the addition of VSP model to the system. However, this will increase the number of terms that will be searched for each search process. Performance issues will meet challenges and will need advance information retrieval system to minimize words with no weight or no records at all.

7. REFERENCES

1. Hammo , Bassam, Mahmoud EL-Haj, Azzam Sleit (2008), Enhancing Retrieval Effectiveness of Diacritisized Arabic Passages Using Stemmer and Thesaurus: *The 19th Midwest Artificial Intelligence and Cognitive Science Conference* (Cincinnati, OH, USA).

2. Lazarinis , Fotis, Jesus Vilares Ferro, John Tait (2007), Improving Non-English Web Searching (iNEWS07), *SIGIR Forum* 41(2) 72-76

3. Oracle (2010),http://www.java.com(accessed 17 April 2010).

4. Sudeshna Sarkar (2007), Regular Expression Matching for Multi-script Databases, *Bulletin on the Technical Committee on Data Engineering* 30(1) 17-29.

5. Sun Microsystems (2010), MySQL 5.5 Reference Manual (2010). Available at: www.mysql.com (accessed 16 Mar 2010).

6. The Unicode Consortium (2006), *Unicode Standard, Version 5.0*, Addison-Wesley, 5th edition.

7. Thabet , N. (2004), Stemming the Qur'an, *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*.

8. Zaki Aldeen Almonthery (2003), Mukhtasar Saheeh Muslim.

9. Alhajjar , A., Mohammad Hajjar, Khaldoun Zreik (2009), Classification of Arabic Information Extraction methods, *2nd International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

10. Beitzel, S., U. Syed, E. Jensen, O. Frieder and D. Grossman (2006), "On the Development of Name Search Techniques for Arabic", *Journal of the American Society for Information Science and Technology*, 57(6), pp.728– 739.

11. Carol Lundquist, Ophir Frieder, David O. Holmes and David Grossman (1999), parallel relational database management system approach to relevance feedback in information retrieval, Journal of the American Society for Information Science (JASIS), 50(5);413-426.