Extensibility in Arabic Full-Text Indexing

By
SAEED MOHAMMAD RAHEEL

Submitted in partial fulfillment of the requirements
For the Degree of Master of Science

Thesis Advisor: Dr. Haidar Harmanani

Department of Computer Science
LEBANESE AMERICAN UNIVERSITY
June 2003
I grant the LEBANESE AMERICAN UNIVERSITY the right to use this work, irrespective of any copyright, for the University's own purpose without cost to the University or to its students, agents and employees. I further agree that the University may reproduce and provide single copies of the work, in any format other than in or from microforms, to the public for the cost of reproduction.
Lebanese American University

GRADUATE STUDIES

We hereby approve the thesis of

Saeed Mohammad Saeed Raheel

Candidate for Master of Science Degree*.

(signed)
(chair) Prof. Haidar M. Harmanani

Prof. Walid T. Keyrouz

Prof. Iyad Ouesst

Prof. Albert Naccache

Date

*We also certify that written approval has been obtained for any proprietary material contained therein.
Extensibility in Arabic Full-Text Indexing

ABSTRACT

By

SAEED MOHAMMAD SAEED RAHEEL

The Internet has become a gateway for an abundant resource for information. Web search engines provide a useful mechanism through which we can look and access information. Most of the research has been dedicated to Latin languages. Using the Latin alphabet, this work proposes an extensible mechanism for indexing and Arabic search. The method expands the capabilities of Arabic search engines and indexing engines. In this work we propose a rules compiler that extends the Arabic indexing rules used by the indexer at runtime. We validate our approach using a prototype that was built using Java and MS SQL Server as a backend RDBMS. Tests have been made on a sample of Arabic documents and a starting set of indexing rules.
I dedicate this work to the most precious people for me on earth.
All the gratitude and respect are due to them.
All of this would not have been possible without their continuous encouragement and support.
This is why they are the first to deserve the best of my respect and love.
My life and my humble work are dedicated first and last...

To my parents
ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Haidar Harmanani for his guidance throughout my M.S. studies. Thanks are also due to Dr. Walid Keyrouz, Dr. Iyad Ouaiiss, and Dr. Albert Naccache for being on my Thesis committee.

I would like to express my sincere gratitude to the Lebanese American University whose financial support during my graduate studies made it all possible.

I would like also to express my most sincere and warm greetings to my true friend Walid Daher for all the support he has provided me with throughout the whole study.

I would like to express my dearest and warmest gratitude to Mr. Hasan Nahhas for his helpful insight into the Arabic language wishing for him the best. I would like to thank him very much because he has helped me out with my first steps.

I would like also to express my thanks and gratitude to Mr. Khaled Damaj for his very precious support especially with the Arabic information resources he has made available to me.

I would like also to express my thanks and gratitude to Dr. Dalal Abbas for her help and guidance regarding the Persian indexing and language and for Dr. Khalil Jaffal for his help and guidance regarding the Hebrew indexing and language.

Finally, I would like to thank my friends and family for their long support.
### Contents

**Chapter 1 Introduction** .......................................................... 1  
  1.1 Information Retrieval .......................................................... 1  
  1.2 Indexing Documents .......................................................... 2  
  1.3 Indexing Arabic Documents .................................................. 4  

**Chapter 2 Review of Literature** .............................................. 7  
  2.1 Automatic Text Retrieval Systems .......................................... 7  
  2.2 Document Indexing ............................................................ 9  
  2.3 The use of Single Terms and Term phrases in Automatic Indexing... 10  
  2.4 The Use of Syntactic Rules to Generate Indexing Phrases .......... 13  
  2.5 Lemmatization in Arabic .................................................... 15  

**Chapter 3 Arabic Stem Extraction** .......................................... 17  
  3.1 The need to perform stemming .............................................. 17  
  3.2 Determining the type of the word ....................................... 19  
  3.3 The Rhyming technique ...................................................... 21  
  3.4 Extracting the stem .......................................................... 22  
    3.4.1 Stemming techniques for verbs ................................... 22  
    3.4.2 Stemming techniques for nouns ................................... 26  

**Chapter 4 Solution Approach** ................................................. 34  
  4.1 A Proposal for Arabic Search ............................................. 34  
  4.2 Extensibility in Document Formats ...................................... 36  
  4.3 Extensible Arabic Full-Text Indexer .................................... 39  
    4.3.1 The Database component ........................................... 40  
    4.3.2 The Server Component .............................................. 45  

**Chapter 5 Rules Compiler** ................................................... 46  
  5.1 Introduction ................................................................. 46  
  5.3 Description ........................................................................ 47  
  5.4 Examples ........................................................................... 48  
    Example 5.4.1 ....................................................................... 48  
    Example 5.4.2 ....................................................................... 49  
    Example 5.4.3 ....................................................................... 49  
    Example 5.4.4 ....................................................................... 50  
    Example 5.4.5 ....................................................................... 50  

**Chapter 6 Experimental Results** ............................................. 54  
  6.1 Indexing Duration ............................................................... 55  
  6.2 Search Hits .......................................................................... 55  

**Chapter 7 Rules Compilation and Indexing in Hebrew and Persian Languages** .......................................................... 58  
  7.1 The Persian Language .......................................................... 58  
  7.2 The English Language .......................................................... 59  
  7.3 The Hebrew Language .......................................................... 61  

**Chapter 8 Conclusion and Future Enhancements** ................................. 64  
  8.1 Conclusion ........................................................................... 64  
  8.2 Future Enhancements ............................................................ 64  

References ....................................................................................... 68  

Appendix A Definitions for the Rules Compiler Conditions and Actions ........................................................................... 70  

Appendix B Transliteration Conventions ....................................................... 73  

Appendix C Parser Class .................................................................. 74  

Appendix D Rules List ...................................................................... 73
List of Figures

3.1 Stemming Process Flowchart ................................................................. 19
4.1 Graphical Representation of the System .................................................. 39
4.2 Database Diagram .................................................................................. 42
7.1 Indexing Durations ................................................................................. 47
7.2 Search Lists Accuracy ............................................................................ 48
List of Tables

Table 5.1 List of Prefix Pronouns ................................................................. 29
Table 5.2 List of Suffix Pronouns ................................................................. 29
Table 5.3 List of Combination of Suffix Pronouns ...................................... 30
Table 5.4 The Common Five Verbs ............................................................. 31
Table 5.5 List of Ten Derivations ................................................................. 32
Table 5.6 List of Attached and Detached Pronouns .................................... 34
Table 5.7 The five Common Nouns .............................................................. 35
Table 5.8 Standard Syntaxes for Plural Nouns .......................................... 37
Table 5.9 A Subset of Exceptional Plural Nouns ......................................... 37
Table 5.10 A Subset of M-Derivations ......................................................... 39
Table 5.11 A Subset of T-Derivations .......................................................... 39
Table 5.12 Miscellaneous Derivations ......................................................... 40
Table B.1 Transliteration Conventions ....................................................... 61
Chapter 1
Introduction

Public libraries were the main source of information. All research was done by going through all available resources in order to extract only the material relevant to the subject sought. As the amount of information grew bigger, the need for a scalable solution to this problem became essential. Therefore, researchers started moving towards computers and the area of Information Retrieval emerged. Since it was an automated process it was called Automatic Information Retrieval.

1.1 Information Retrieval

There is no unanimous agreement on the definition of the Automatic Information Retrieval term. However, Lancaster defines it in [1] as: “Information retrieval is the term conventionally, though somewhat inaccurately, applied to the type of activity discussed in title of volume. An information retrieval system does not inform (i.e., change the knowledge of) the user on the subject of his inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to his request.”

The above definition can also be used to define search engines, which simply pinpoint to locations inside a certain document. In this context, the term document is totally interchangeable with the term Information. Many specific features differentiate between search engines: the
indexer that is running behind every search engine, the rules used for indexing information, and the language of the documents being indexed.

1.2 Indexing Documents

There are two types of indexing: Thesaurus based indexing and Full-Text Indexing. Thesaurus based indexing relies on the synonyms of words in addition to the words themselves. Therefore, we might represent a certain document with a group of words that do not even exist in that document. The problem with this approach is that it is harder to implement, if compared to the full-text indexing approach, and it requires continuous monitoring and human intervention. One way to ease up its implementation is by using a dictionary or a thesaurus that would help in the indexing process but that does not eliminate the human intervention factor completely.

Full-text indexing, on the other hand, is easier to implement because it relies on the text itself rather than on a thesaurus or a dictionary. Whenever a document is subject to full-text indexing, the result of this process is a set of keywords that constitute the content of this document. However, not all the keywords describe exactly what this document is all about. Only words that add to the meaning of the text are considered. Other words such as the, a, on, at, etc are considered as noisy words which do not add to the meaning of the text and they are discarded by the indexer in the indexing process.

In their study about generating “Indexing Phrases” [8], Salton, Zhao and Buckley proposed a phrase generation method that helps in generating the so-called “content identification” of natural language texts. “The phrase generation method is based on a simple
language analysis system that determines the syntactic function of individual text words with a high degree of accuracy, and chooses indexing phrases based on weights assigned to the phrase components." [8]. Assigning a weight to a term relies on a number of factors: how many times this term appears in the text, what is the distance between the occurrences, how many times the stem of that word appears in the text, and last the distance between the occurrences of the stem. The more the term appears in the text the higher the weight will be. The closer the occurrences of that term are the higher the weight becomes. Similarly, the more that stem of that term appears in the text the higher the terms' weight will be and the closer the stem occurrences are the higher the weight will be.

Moreover, Walid Daher in his study about indexing Arabic documents [9] has referred to the same concept with what is so-called "subject headings." "Indexes, when extracted from the documents, are referred to as keywords." Thus, a "keyword" is the term used by documenters to signify an index. Keywords, in turn, are used to build "Subject headings." Usually, subject headings are phrases composed of more than one key-word. A single document may have as many subject headings as possible. "The more subject headings a document is assigned, the more likely that a user might hit that document upon searching for a topic. Composing subject headings is what documenters actually do." [9]

However, since in our case we are relying on computers, subject headings or indexing phrases will be extracted automatically. Weights that will be assigned to the terms will help us
maintain an acceptable range for the Recall and Precision measures\(^1\). Both of these measures help us define how relevant the results returned by the search engines are and thus detecting how efficient our indexing algorithm is in building the “Indexing Phrases.”

Once subject headings are identified, the search engine uses them to locate and identify documents instead of going through the whole text again.

### 1.3 Indexing Arabic Documents

Natural languages rely on grammatical, syntactical, and morphological rules. The level of difficulty and sophistication depends on the language itself. For example, Arabic is semantically very flexible yet morphologically complex. In other words, we can create from a stem any meaningful conjugation and it would give a different meaning depending on its composition. Therefore, we have to follow certain fixed and standard syntactic and morphological rules to do that. For example, let us take the basic three letters paradigm in the Arabic language. For it, one can count various conjugations: The active participle (faṣal equivalent to 'doer' in English), the present verb (yaf:al equivalent to 'doing' in English), the future verb (sayaf:al equivalent to 'will do' in English), the passive participle (esem el maṣoul equivalent to 'done to' in English), the plural conjugations such as (sayaf:alou equivalent to 'they will do', faːalou equivalent to 'they did', yaf:aloun equivalent to 'they are doing', faːaloun equivalent to 'going to do') and the singular feminine conjugations (safaːsoul equivalent to 'she will do', tafaːsoul equivalent 'she is doing', faːilal equivalent

\(^1\) Recall is the proportion of relevant records retrieved, and precision is the proportion of retrieved records that are relevant. In general, one would like to retrieve an acceptable number of relevant items while rejecting most of the extraneous material producing thereby both a high recall and high precision. [10]
to 'she is going to do', fa'alat equivalent to 'she did'), and the plural feminine conjugations such as
فِعْلُ، يَفْعَلُنَّ، يَفْعَلُنَّ فَعَالَات (fa'alat equivalent to 'they (feminine) are going to do in the present continuous', sayasalna equivalent to 'they (feminine) are going to do in the future', ya'salna equivalent to 'they (feminine) they are doing', fa-alna equivalent to 'they (feminine) did'). As we have seen, the Arabic language has the flexible morphological rule that allows the pronouns to be built in as part of the verb or noun or to be attached to it such as in the term يَفْعَلُهُ (ya'alouhou), where the ي has become a part of the verb فعل and the أ has been attached to the end of it.

Similarly, the الت has become part of the original verb فعل and the أ has been attached to the end of it to form the verb قِطْعَهُ. In addition, another morphological complexity in Arabic is that more than one pronoun can be added to any tense and to any gender related conjugation such as فَعَالَتَين (fa'alatouhonna equivalent to 'they (feminine) are doing more than one thing') where the original verb is فعل and the أ الت has become part of it and the أ ن has been attached after them as suffixes. Note that this complexity increases with the addition of prefixes also.

Therefore, what we need to do is either to make sure that we embed all the indexing rules into the system, which is a tedious task and renders the modification to these rules a cumbersome job, or we rely on the main subject of this study: *Extensibility in Indexing Rules through the use of a rules compiler*. For that purpose, we have created an extensible rules compiler. The rules compiler enables us to create and customize at anytime the indexing rules used in the indexing process of the Arabic documents in order to achieve a high level of indexing accuracy.

An additional extensibility-related issue is the proliferation of file formats. One can find an infinite number of file formats scattered all over the internet. For example, one document can be a pure text file, an Excel spreadsheet, a PowerPoint presentation, an HTML document etc.
The text file should be parsed simply from the beginning to the end. However, an Excel sheet or a PowerPoint document contains information that is specific to the software managing them, mainly Microsoft Excel and Microsoft PowerPoint. What the user is concerned with is just the text written in these documents. Therefore, parsers have to be built for this purpose. These parsers should be integrated into the system in such a way that ensures minimal effort and user intervention. Therefore, we have created a mechanism through which the user is able to add a new parser for any file format while maintaining as much as possible a minimum level of user intervention.

Chapter 2 reviews related work and sheds some light on background information in addition to a description of what has been already done before. Chapter 3 presents a description for the techniques used in Arabic stem extraction. Chapter 4 presents a detailed description for the prototype we've built in order to test the validity and the effectiveness of the solution proposed i.e. the parser for different file formats and the rules compiler. Chapter 5 focuses on the main purpose of this study: the rules compiler. Finally, chapter 6 and 7 we will present some of the test results done using the proposed system as well as a description for the applicability of the system to other languages.
Chapter 2
Review of Literature

Not much work has been done about Arabic search engines. Most of the efforts revolved around the English language and the various methods through which English documents can be indexed.

2.1 Automatic Text Retrieval Systems

Salton defines Automatic Text Retrieval Systems as: "an automatic text retrieval system is designed to search files containing natural language documents and to retrieve certain stored items in response to queries submitted by a user population." [6] In his definition, Salton, referred to a document by using the word ‘item’. Whenever a document is indexed and stored, the system identifies keywords that are contained in the document or sometimes supplemented by additional related information. The query supplied by users are composed of a set of words interrelated by the Boolean operators and, or, and not. The system responds to a query by looking for items containing combinations of these query words.

"It is customary to evaluate the effectiveness of a retrieval system by using a pair of measures, known as recall and precision. Recall is the proportion of relevant material actually retrieved from the file, and precision is the proportion of the retrieved material which is found to be relevant to the user's information needs" [6].
Two very important notions were presented in the above definition. Whenever a document is indexed, a set of keywords is extracted and set to be the “identifying keyword list” for that document or the “Indexing phrase” [8] as referred to in Salton, Zhao, and Buckley’s study. This keyword list is then compared to the query submitted by the user. The total number of documents, out of the whole collection of documents indexed, proven to be related to the query submitted is referred to as the Recall. The total number of results matching exactly what the user needs out of these retrieved items is what is known as Precision.

In order to achieve a high level of recall, a search would retrieve everything relevant to the query submitted. However, in order to maintain a high level of precision, at the same time, the search rejects a large proportion of any extraneous items. This will result in a very high level of recall and precision almost equal to one. However, it is practically known that recall and precision vary inversely since it is always difficult to retrieve everything wanted while also rejecting what is unwanted. The main problem lies in the diversity of the queries that the users submit. The more specific the query submitted is the highest the precision measure becomes, but also very few documents will be retrieved and thus achieving a low recall. The broader the queries submitted are the higher the recall is since more relevant records are retrieved but also more non-relevant ones, and thus decreasing the precision measure.

A compromise for this dilemma is to for users to submit queries that are neither too narrow nor too broad and thus achieving quite a reasonable level of recall and precision. Studies showed that if users are to decide which is more preferable, most users preferred precision over recall because then fewer items will be retrieved and thus less time the user has to spend browsing all that information [8].
Our study will follow a mechanism that allows us to present to the user the various search results related to the queries he/she submits. Regardless, whether his/her search queries are broad or narrow our system will still sort the results in their decreasing order of relevancy. The system uses the term's occurrence as well as its stem's occurrences and the terms positions at which it was found as well as its stem's positions at which it was found to calculate the term's degree of relevance. The terms with the highest relevance will appear at the top and the terms with the lowest relevance will appear at the bottom.

2.2 Document Indexing

Indexing can be done in many ways and its results can vary according to the mechanism applied. For example, meta indexing indexes documents based on meta data that exist in their headers. Full-text indexing, on the other hand relies on the text of these documents. One approach used in full-text indexing is to generate term phrases that identify the content of each document. Rather than storing the whole content of a document to be used later on when searching, the indexer generates terms based on some semantic and syntactic rules in an automatic way and assigns them for these documents for later use. Later on, when a user submits a query, the search would be then done on these terms rather than on the text of the documents indexed.

The representation of each document (i.e., its indexing phrase) affects directly the recall level since the broader that phrase is the more likely the document will be retrieved. Therefore, broadening or narrowing the index phrase is one way to affect the recall level. One method to broaden the indexing phrase is to use truncated terms (an approach that we will be using in our
study to generate what we will refer to later in our study as *stems*). Instead of using the entire original word for query or document identification, its stem will be used instead. For example, using a stem such as “analy” definitely hits upon so many other terms such as analysis, analyst, analysis, analyzer, etc. And thus a broader scope of words and thus documents is being covered. Another recall enhancing measure that Salton proposes is the use of synonyms or terms that are related to the original terms found in the documents. These synonyms can be obtained from available thesauruses and term hierarchies. (We shall not rely on the latter approach in this study.)

There are additional measures to enhance the precision level. A high precision level is achieved by assigning weights to terms since more important terms will be distinguished from other less important ones: “Such discrimination may help in ranking the output in decreasing order of presumed importance” [6]. Other precision enhancement measures include the use of indexing phrases rather than single terms. In this study, we shall rely on both measures by assigning weights to stems and their corresponding original words and we will generate indexing phrases and thus enhancing the precision level.

In our study, we follow the full-text indexing methodology to index documents. Terms will be ranked according to their degree of relevance. We will process terms and generate the stem for each. A weight will be assigned to each of them based on measures that were mentioned in the previous section.
2.3 The use of Single Terms and Term phrases in Automatic Indexing

Indexing is the task of storing content identifiers in order to be used later on by search queries. "In a retrieval environment, indexing is the task which consists in the assignment to stored records and incoming information requests of content identifiers capable of representing record or query content." [10] In order to achieve a high level of accuracy, content identifiers or index terms must be treated carefully. Terms that are too broad must be rendered specific by replacing them with term phrases, while narrow terms must be broadened by supplying synonyms and related terms normally extracted from a thesaurus.

Natural language terms are usually controlled by a dictionary or a thesaurus and are assigned to the stored information records. The search engine extracts by comparing the relevant records the assigned terms to the terms supplied by a user query.

In an information retrieval environment, various possible identifiers (terms, keywords) operate unequally and ineffectively. In other words, "some terms are used at the right level of accuracy while others are neither too broad nor too narrow" [10]. For example, a term like "computer" would be too broad to be assigned to a computer science collection and totally inappropriate for a general social science environment since no term can be fully identifiable with it.

The study done by C.T.Yu, G. Salton, and M.K. Siu under the title "Effective Automatic Indexing using Single Terms, Term phrases and Thesaurus Class Assignments" [10] revolves around achieving acceptable levels of recall and precision. "It is convenient to relate terms
specificity to the assignment frequency of each term to the items of a collection" [10]. The best terms are likely the ones that are neither too frequently nor too rarely found in the collection of stored records, for if a term occurs too frequently it leads to the retrieval of too many extraneous records and produces inadequate search precision. If it is too rare then a great number of many relevant items are likely to be missed leading to recall losses.

Therefore, an appropriate indexing policy then uses those indexing terms, which exhibit the appropriate level of specificity as reflected in the term frequency characteristics in the collection under consideration, while modifying at the same time terms that are used too frequently or too rarely:

a. Broad terms that occur in too many records and are therefore likely to produce inadequate retrieval precision should be combined with other appropriate terms to form term phrases.

b. Narrow terms which occur in too few records and are therefore likely to produce inadequate recall should be incorporated into thesaurus classes, and these classes should be assigned for content identification instead of the individual terms.

For example, a content identifier such as “programming language” will be assigned instead of the individual broad terms “programming” and “language” alone. The terms are then deleted and when using a thesaurus for identification, the synonyms or related terms are then added to the original terms set. Thus, a term such as “minicomputer” might be placed into a common thesaurus with "microcomputer", "hand calculator", "desk calculator", etc. The
assignment of the thesaurus would simultaneously supply all these terms as content identifiers for a particular document.

In our study the system will not generate term phrases for documents but rather will be indexing and storing single terms. The system will assign weights to these terms in order to determine the recall and precision for each search. These terms will be used to reply to the queries submitted by users. In order to reply to the phrases submitted by the users, the system will search for documents containing all or some of these terms. The documents with the most degree of proximity will be returned first while the ones with the least proximity will be displayed last.

2.4 The Use of Syntactic Rules to Generate Indexing Phrases

In order to be able to ensure quite an acceptable accuracy level for the term phrases generated, Salton, Zhao, and Buckley suggested the use of the syntactic rules of the English language. Weight assignment to individual phrase components was used also to determine the syntactic function of individual text words and choose the indexing phrases based upon these weights.

Many different language analysis procedures have been proposed to control the assignment of index terms to documents stored in information retrieval systems. “A standard method uses sets of properly weighted single terms for content representation, the term weights being used to distinguish the more important from the less important terms” [8]. Another method uses predefined thesaurus designed to recognize synonyms and other relations between terms.
and on the construction of term phrases consisting of combinations of single terms. Still another approach is based on the so-called knowledge base which provides a complete semantic characterization of the terms in a particular document.

However, the use of thesauruses and knowledge bases to specify the relevant semantic environment raises very substantial conceptual and practical problems when the subject area of interest is not severely circumscribed. Henceforth, the most immediately usable approach is the construction of term phrases to supplement the single term indexing technique. “Term phrases are sets of single terms that collectively carry meaning and represent more refined entities than the individual term components” [8]. For example, “presidential election” represents a concept quite apart from that of “presidential” or “election” alone.

There are many different ways to generate term phrases. One way uses a statistical term co-occurrence method where phrases are defined as two or more that occur frequently in close proximity to each other in the texts of a document collection. However, the statistical term co-occurrence approach generates a large number of potential phrases, many of which are semantically improper. In our study, we will be assigning term weights based on the occurrence of these terms. This is because the more sophisticated the morphological rules of the language becomes, especially if we’re talking about Arabic, the more difficult it would be for us to resolve them by a purely syntactic approach. For example, This is mainly due to the fact that relevant subject phrases cannot be generated by using a shallow text analysis without appropriate semantic rules, one problem that hampers most of the phrases generation systems from functioning accurately. An alternate approach could be achieved based on a dictionary search that would identify each text word as being a noun, and adjective, an adverb, and so on.
In our study, we will rely upon a mixture of these approaches. We'll be using a predefined, though not that accurate, so called thesaurus. However, the thesaurus will not contain full lists of words and relations between them. On the contrary, this thesaurus will start with a minimal set of keywords and their types, mainly adverbs and stop list terms. This system will gradually build the thesaurus to contain key terms and phrases that would each best describe the contents of the documents being indexed. The indexer will rely on a certain mechanism to differentiate between nouns and verbs. It will generate stems and then assign them weights so that indexing phrases can then be produced. The indexing phrases are then used to decide upon which document is the most relevant to the user's needs and which is the least.

2.5 Lemmatization in Arabic

"In the field of NLP [Natural Language Processing], lemmatization refers to the well-known process of relating a given textual item to the actual lexical or grammatical morpheme it goes back to in the system of the language" [11]. This is the definition that Dicy has given to lemmatization or what we will refer to in our study as stemming. Lemmatization is in the core of the indexing process. It is what takes back a textual item, what we refer to as a term in our study, to its original lexical or grammatical morpheme i.e. what we refer to as the stem.

Stemming in Arabic is rendered more difficult due to the non-vowelled standard writing. The majority of the documents lack the diacritical signs which indicate the short vowels, the doubling of a consonant, basic case-ending, etc. This difficulty is also due to the complex forms that a word can take. This complexity is due to the following reasons:
• The presence of prefixes.
• The presence of suffixes.
• The presence of patterns.

Dichy defines a general form for any word in Arabic as:

\[ \text{Word} = \text{prefixes} + \text{stem} + \text{suffixes} \]

Based on this definition and starting from any basic stem one can find various variations for this stem by adding suffixes and prefixes. For example, for the noun derivation of a certain stem such as the term \( \textit{kutub} \) (books), we add the suffix \( \textit{kutub} + iy \) which renders it to \( \textit{kutuby} \) (my books). Other derivations are ones that change the tense of a verb by adding a prefix such as \( \textit{ya'} \). For example, \( \textit{kataba} \) (wrote) + \( iy \) = \( \textit{yaktoubou} \) (is writing). Other derivations are ones that derive the active participle (\( \textit{ism al-fā'il} \)) or the analogous adjective (\( \textit{sifa musabbaha} \)).

Therefore, Dichy suggests that we build some sort of a lexical database where all the finite grammar-lexis relations are taken into account. These relations will be the basis of the lemmatization process.

Lexical Analysis of Arabic documents will help a lot in the lemmatization process. However, this renders the job more complex since the Arabic language is very rich in lexical rules. In this study, we will build a database for syntactical rules containing prefixes, suffixes, stems, patterns, etc. This database should form a good and solid ground for any further enhancements and should help in the future and pave the way for more complex lexical analysis.
In our study, we will perform lemmatization for stemming. All the prefixes and suffixes will be removed. What is left will be matched against patterns in order to remove the infixes. The result of these processes should be the stem of the term being processed. Not all word will go through this whole process. Stop List terms will be discarded directly while Non Derivable terms will be stored directly. Other terms will go through the whole process. The more accurate and numerous the rules is the higher the level of stemming accuracy will be.
Chapter 3
Arabic Stem Extraction

3.1 The need to perform stemming

In his study [6], Salton proposes the use of "truncated terms, or word stems, instead of the original complete terms, for query or document identification"[6] in an attempt to reach a higher level of recall. The example he used was the word "analy". Therefore, by using "analy" as an indexed term, the indexer would then be able to cover notions such as "analyst", "analysis", "analyzer", etc., since "it has a broader scope than any of the complete words"[6]. In addition, the use of term weights would balance the equation and help the indexer reach a higher level of precision "because the weights distinguish the better, or more important, terms from the less important ones"[6]. Such discrimination will help in ranking the output in decreasing order of presumed importance. Moreover, Pohlmann and Kraij [12] have found out that "Surprisingly, Krovetz finds that all stemmers yield a significant improvement" over no stemming. The derivational stemmer generally gives the best results. Krovetz notes that improvements from stemming increase at higher levels of Recall and that derivational morphology is responsible for improvement at high levels of Precision."[12] According to Pohlmann and Kraij, one of the techniques employed in Information Retrieval to improve performance is stemming of document and query terms. Some researchers have reported favorable results using linguistic stemming algorithms applied to languages morphologically more complex languages. In addition, researchers have noted that there are many factors affecting the performance and the recall and

\[\text{According to Pohlmann and Kraij, enhancement figures range from 1.3 to 45\% in average Precision at Recall 0.25, 0.50 and 0.75.}\]
precision levels such as the document length, the length of the query submitted and the morphological complexity of the language being indexed. Therefore, performing stemming in languages, which are more morphologically complex, yields to better results in Recall and Precision.

In the Arabic language one can find more than 150 basic morphological representations for the same term. If we are to add the various additional suffixes and prefixes to these 150 morphological representations then their count increases significantly. Therefore, the morphological representation of a term in Arabic is much more flexible and complex than that in English and this is one additional reason why we need to perform stemming.

In our study, we will rely on the above mentioned techniques. We will extract stems from the original terms and will assign weights to each of them in order for the system to present more results to the user, and thus reaching a higher level of recall, while sorting these results in decreasing order of relevance, and thus maintaining an acceptable level of precision.

Figure 3.1 presents a graphical representation for the flowchart of the stemming process. When the indexer receives a term it tries to determine its type. Depending on its type the indexer treats the term accordingly. Regardless whether the term is a noun or a verb still the indexer needs to remove the suffixes, the prefixes and the infixes from it and then match it against patterns to determine its type.
Prefixes + Stem (Infixes) + Suffixes

Determine Type

Noun
- Remove/Convert Noun Suffixes & Prefixes
  - Rhyme Against Noun Patterns
    - Stem Found?
      - YES
        - STEM
      - NO

Verb
- Remove/Convert Verb Suffixes & Prefixes
- Indecisive
- Remove/Convert Suffixes & Prefixes
  - Rhyme Against Verb Patterns
    - Stem Found?
      - YES

Figure 3.1 Stemming Process Flowchart
3.2 Determining the type of the word

Before extracting the stem of a certain word, the module tries to determine whether this word is a verb or a noun because the stemming techniques for both types differ from each other. One way to determine the type of a word is to examine its precedent word. By determining the type of the word being processed the stemming task becomes faster. For example, let us consider the case of stop list terms. Some stop list terms precede nouns only while others precede verbs only. For example, أدوات التصـيب والجزم (adawat al naseb waljazem) precede verbs only while الأسماء الموصولة (al asma' al mawsoula) and الأحرف الجر (ahrof al jar) precede nouns only. The technique applied to determine the type of the word is to rhyme it against patterns (Rhyming is a technique that will become clearer in the next section). If it matches with يفعل (yaf'al), for example, then it is a verb; however, if on the other hand, it matches with فاعلة (fa'ela) then definitely it is a noun. Moreover, if the preceding word was a verb then definitely it is a noun because no two verbs can succeed in the Arabic language. However, if none of the preceding approaches was able to determine the type of the word then the system will treat this term as unknown and will process it once as a noun and another time as a verb. The approach that is able to produce a stem will be adopted as the solution and is then the stem of that word and then its type relies upon which technique was successful.

Stemming a word to its root term is an important stage that a document has to undergo while performing indexing. Before stemming a word, the module has to decide upon its type because stemming techniques for nouns differ from those of verbs. Some suffixes are used with verbs only while others are used with nouns only. Moreover, some prefixes can be used with verbs only while others can be used with nouns only. Another difference is the patterns for each.
The patterns for nouns are different from those of the verbs. Therefore, after deciding upon the type of the word, the indexer uses Rhyming to extract the stem of the term being indexed.

3.3 The Rhyming technique

When we say that a word matches another we are referring to the Arabic terminology (ala wazen). In Arabic, the basis for all the stems is the three letters paradigm \(\text{ف ع ل} (\text{fā', cayn, lām})\) and from it other patterns are generated. For example, the \(\text{ا} (\text{alef})\) is added to it as an infix and the \(\text{ا} (\text{alef})\) as a suffix to form the rhyme \(\text{ع ل} \text{ا} \text{ا} \text{ا}\). All the words that rhyme with \(\text{ع ل} \text{ا} \text{ا} \text{ا}\) should have a \(\text{ا} \text{ا} \text{ا}\) as an infix after the first character and a \(\text{ا} \text{ا} \text{ا}\) as a suffix at the end and should be phonetically identical with it i.e. its syllables should match that of the rhyme. For example, the term ف ع ل دارسة since it contains a \(\text{ا} \text{ا} \text{ا} \text{ا} \text{ا}\) as the second character and a \(\text{ا} \text{ا} \text{ا} \text{ا} \text{ا}\) as the last character and is phonetically identical to it. In fact, all the basic patterns are mere derivations and additions to those three letters. What is common among all derivations is that they all preserve the existence of those three letters. None of these letters is missing in any of the majority of the derivations. It was mentioned previously that one way to decide whether a word is a noun or verb is to examine its preceding word. Another technique, of course if the latter failed, is Rhyming. Rhyming is performed by comparing all of the additions performed on the three basic letters ل ف ع (fa-ala) to the additions performed on the three basic letters that form the stem of the word being processed. If they match then we say that the word and the pattern “rhyme” and thus we extract the stem of that word by extracting only the three letters that fall exactly in the same positions as the three letters ل ف ع and ل (fā', cayn, and lām) in the pattern. We are sticking to

---

3 Rhyming in Arabic can be described as pattern matching in English i.e. we say that a word 'Rhymes' or 'matches' a 'rhyme' or a 'pattern' when it phonetically matches that pattern.
three letters only simply because all the verbs and nouns in Arabic originate from three letters words.

For example, let us consider the word بدرس (yadros) and the pattern يفعل (yaʕal). What the function does is that it ignores all the letters in the verb بدرس that fall in the same positions as the letters ل and compares all the others together, in this case the only letter that is compared is ي (yâ). The algorithm ignores these letters on purpose simply because if they do rhyme then the letters ignored are the stem of that word, in this case درس (darasa). The same applies to all the other patterns. Each and every word is checked to determine its type and accordingly is rhymed with a set of patterns until it matches one and afterwards the stem is extracted.

3.4 Extracting the stem

The main purpose behind determining the type of a word is to decide which stemming technique ought to be used. As it has been mentioned earlier, stemming techniques for nouns differ from those of verbs. We shall start by describing the stemming technique for verbs.

3.4.1 Stemming techniques for verbs

Verbs in the Arabic language come in three tenses: present, imperative, and past. Usually and in most cases, the past tense of the verb is considered to be the stem for all its other tenses.
What differs is that additional letters are attached to it so that it can be transformed into the present or the imperative. These attached letters are either derivations that form the letters of the word "سَلَّمَوْنِيْهَا" (salalhumuniha) or are pronouns. Pronouns in Arabic are of two types: attached and detached. Detached pronouns "الضمائر المنفصلة" (addama'er al monfasila) are stop list terms and thus they are discarded. However, attached ones "الضمائر المتصلة" (al dama'er al mottasila) have to be removed from the verb.

Attached pronouns come either in the form of suffixes or in the form of prefixes. The verb is matched against patterns and all the attached pronouns are removed and what is left of the verb is then returned back for further processing. The verb then goes through the rhyming process. It is matched against the pattern with which it has rhymed and all the additional letters are then removed.

In the Arabic language, patterns are categorized into several fixed categories. However, since our proposed solution should comply with these but not limit itself to them a description of the main verb categories shall be presented in what follows. One type of verbs that has to be considered during the rhyming process is the common five verbs, referred to in Arabic as "الSuffixes" (al af'al al khamsa). These verbs form a finite set that contain a specific type of additions. The main characteristic of these verbs is that they always end with a "ن" (noon) except if they are preceded by a "ن" (noon) except if they are preceded by a أداة جزم (adat jazem) or أداة نصب (adat naseb). In this case, the ن is removed and either an ًا (‘alif) or ی (yā') replaces it.
Table 3.1 contains a list of these verbs. Since it is a finite set of verbs and the additions are known beforehand, the module does not need to perform pattern matching to stem the original verb but rather simply removes all the letters that are underlined from the pattern that rhymes with the word at hand.

### Table 3.1 – The common five verbs

<table>
<thead>
<tr>
<th>Explanation</th>
<th>Another Morphological Representation</th>
<th>First Morphological Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>يستعمل مع جمع المصدر الغائب</td>
<td>يفعلوا</td>
<td>يفعلون</td>
</tr>
<tr>
<td>تستعمل مع جمع المصدر المخاطب</td>
<td>يفعلوا</td>
<td>يفعلون</td>
</tr>
<tr>
<td>تستعمل مع المئتي الغائب</td>
<td>يفعلن</td>
<td>يفعلن</td>
</tr>
<tr>
<td>تستعمل مع المئتي المخاطب</td>
<td>يفعلن</td>
<td>يفعلن</td>
</tr>
<tr>
<td>تستعمل مع المؤنث المفرد المخاطب</td>
<td>يفعلى</td>
<td>يفعلى</td>
</tr>
</tbody>
</table>

The second type of additions that the module has to care for in stemming verbs is a category called the *ten additions* or لزيادات العشرة (azziyadat al asara). One word that combines the letters which are used to perform the additions is مسالمتهمبها. These letters never come in the form of suffixes but rather as prefixes or in the middle of the word, which renders the stemming task somehow more difficult. Scholars agreed upon making the three letters ع ل as
the basis for representing verbs. Any other letters are mere additions and must be dropped. Again the set of combinations of these additions is finite and is represented in Table 3.2.

As it is the case with the common five verbs, the word is rhymed against any of these patterns. If a match is detected then the additional letters (the ones that are underlined) are removed.

<table>
<thead>
<tr>
<th>Stem</th>
<th>Example</th>
<th>Pattern</th>
<th>Stem</th>
<th>Example</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>أصل الفعل</td>
<td>مثال</td>
<td>الزيادات</td>
<td>أصل الفعل</td>
<td>مثال</td>
<td>الزيادات</td>
</tr>
<tr>
<td>هزم</td>
<td>إهزم الأداء</td>
<td>ضرم</td>
<td>ضرم النثيران</td>
<td>فعل</td>
<td></td>
</tr>
<tr>
<td>قرف</td>
<td>افترض خطأ فاحما</td>
<td>سرع</td>
<td>سرع البحث</td>
<td>فعل</td>
<td></td>
</tr>
<tr>
<td>زهر</td>
<td>إهزم الورد</td>
<td>قتل</td>
<td>قتل الأداء</td>
<td>فعل</td>
<td></td>
</tr>
<tr>
<td>غرق</td>
<td>إضروت عيناه</td>
<td>سبب</td>
<td>تسبب في وفاته</td>
<td>فعل</td>
<td></td>
</tr>
<tr>
<td>خرج</td>
<td>استخرج النغط</td>
<td>عطف</td>
<td>عطف مع صبيته</td>
<td>فعل</td>
<td></td>
</tr>
</tbody>
</table>

Whenever a word is detected to be a verb, it has to undergo the stemming process exactly as the previous sections were ordered. First of all, the algorithm checks for any attached pronouns. If any are detected then they are removed. If the verb was still composed of more than 3 letters then it is checked against the common five verbs. If it proved to be one of them then the

\[\text{The stressing character – known as } \text{٢} \text{ – is considered a letter by itself in the Arabic language.}\]
stemming is performed by removing the additional letters; else it has to rhyme with one of the ten derivations leaving the word in its original three letters format.

Again, the system does not limit itself to the values stated in the tables above and does not categorize them in that way but rather offers the user the flexibility to add any new pattern without worrying about the category to which this pattern belongs. Moreover, the system is built in such a way that it is even able to detect the pattern for an Unknown word type.

3.4.2 Stemming techniques for nouns

Next, stemming techniques for nouns will be discussed. Nouns can be singular, double or plural. Variations and exceptions for these forms also exist and the task becomes even harder when it comes to dealing with masculine or feminine terms.

Pronouns attached to nouns are only variations for the attached pronouns or al dama'er al mounfasila) which change in their form so that they become al dama'er al mouttasila. These pronouns come in the form of suffixes. Table 3.3 contains a list of detached pronouns and their attached form when they become attached.
Table 3.3 – List of Attached and Detached Pronouns

<table>
<thead>
<tr>
<th>Explanation</th>
<th>Suffix Form</th>
<th>Detached Form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In addition to the above table, examples of other letters can also be attached as prefixes. These letters are ل، ك، س، ف، م، ب (al, kal, syl, bal, lam). Again, pattern matching helps in detecting these attached letters and removing them. As it is the case with verbs, nouns also might belong to a category called the common five nouns or الأسماء الخمسة (al asma' al khamsa).
Similar to the tense in a verb, a noun has a state. This state depends upon its position in the sentence or its preceding word. These states are either مرفوع (marfou'a) or منصوب (mansoub) or مجرور (majrour). The Arabic language also contains special characters, the diacritical marks, that are situated either at the top of or at the bottom of a letter. These characters are called الحركات (al harakat). They can be either a فتحة (fatha) or a كسرة (kasra) or ضمة (damma) or a neutral character called سكون (soukoun).

The state of these verbs is identified by "letters" instead of by diacritical marks. So, instead of being مرفوع with a ضمة it is مرفع with a و (wāw). The same applies for the منصوب where the ي replaces the فتحة and for the مجرور (majrour) state where the ي replaces the كسرة. And the cases are shown clear in the above table.

When the system detects a three-letter noun, it performs pattern matching rather than rhyming. If the noun proves to belong to this set then the first two letters are extracted as the stem for this word. One last thing, among these nouns, only ذو (tou) is considered to be a stop list term.

As it was mentioned earlier in the process of dealing with nouns, one of the difficulties that the system faces is to detect whether a noun is in its singular or plural state. Similar to other languages, the Arabic language also has irregular plural form and an additional state called المثنى.
(al mouthanna), which is the case when dealing with two entities and the singular state occurs when we are dealing with one entity and the plural when dealing with three and above.

Moreover, within these states, the grapheme differs when it is addressing a male or a female. So, for example, كتبها (kitabouho) adds the singular masculine pronoun ها to the noun كتب to represent the singular masculine morpheme whereas it adds the ه to form كتبها (kitabouha) when we’re talking about a female. كتبهن (kitabouhonna) is for the plural feminine state whereas كتبهم (Kitabouhom) is for the plural masculine state. What makes the task even more challenging is when we’re dealing with exceptions or what is called in Arabic جمع تكسير (jamea taksiir) where not only additions exist but the whole noun changes sometimes as, for example, the plural of امرأة نسوة is شجرة (sajara)is شجرا (sajarat), the regular plural form and شجار (asajar) as the irregular plural form.

Table 3.5 lists the regular form for the plural state. Table 3.6 contains a subset of irregular plural forms.

Table 3.5 – Standard syntaxes for plural nouns.

<table>
<thead>
<tr>
<th>Explanation</th>
<th>Stem</th>
<th>Explanation</th>
<th>Stem</th>
</tr>
</thead>
<tbody>
<tr>
<td>المفرد المئون</td>
<td>فاعلة</td>
<td>singular</td>
<td>فاعل</td>
</tr>
<tr>
<td>ملائم المئون</td>
<td>فاعل</td>
<td>فاعلا</td>
<td>فاعليين</td>
</tr>
<tr>
<td>جميع المئون</td>
<td>فعلت</td>
<td>جميع المذكور</td>
<td>لسع</td>
</tr>
<tr>
<td>Stem</td>
<td>Singular Form</td>
<td>Example</td>
<td>Exceptional Plural Form</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>صحيح</td>
<td>مصباح</td>
<td>مصليج</td>
<td>مفاعيل</td>
</tr>
<tr>
<td>قسم</td>
<td>قسم</td>
<td>تقاسم</td>
<td>تقاسم</td>
</tr>
<tr>
<td>دخن</td>
<td>دخنة</td>
<td>بدأهن</td>
<td>بدأهن</td>
</tr>
<tr>
<td>جنح</td>
<td>جنحة</td>
<td>أجنحة</td>
<td>أجنحة</td>
</tr>
<tr>
<td>علم</td>
<td>علم</td>
<td>طلء</td>
<td>طلء</td>
</tr>
<tr>
<td>قتل</td>
<td>قتلى</td>
<td>فتلى</td>
<td>فتلى</td>
</tr>
<tr>
<td>كتب</td>
<td>كتابة</td>
<td>إلكيب</td>
<td>إلكيب</td>
</tr>
<tr>
<td>عمل</td>
<td>عمل</td>
<td>إصل</td>
<td>إصل</td>
</tr>
</tbody>
</table>

Pattern matching is used to decide whether a noun is in its plural or singular form. If it matched then the underlined letters are removed and the noun then is reduced to its singular form. If however, it was not from the regular plural form then rhyming is used to check if it is in its irregular plural form. Again, the underlined letters are removed in order to reduce the noun to its singular form. If, however, the noun did not match with any of the regular plural form and neither rhymed with the irregular ones then it is assumed to be in its singular state.
As we have seen earlier with verbs, nouns also have derivations. Verbs derived belonged to the مصادر المؤولة (al masader al mou'awwala), whereas nouns derived belong to what is called in Arabic مصادر تأويلية (masader ta'iyya) and finally a bunch of miscellaneous derivations which have no specific common format or rule. To make the task of the indexer easier, the module always checks for the first letter. If it was a م (mīm) then it is rhymed against the patterns found in Table 3.7. If it was a ت (tā) then it is rhymed against the patterns found in Table 3.8. If however the derivation was an irregular one then it is rhymed against the miscellaneous patterns, some of which are found in Table 3.9. And again, to extract the original word, the system removes the letters that are underlined.
Table 3.7 – A subset of M-Derivations

<table>
<thead>
<tr>
<th>Stem</th>
<th>Example</th>
<th>M-Derivation</th>
<th>M-Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>نبع</td>
<td>مخرج / متبع</td>
<td>مفعل / مفعل</td>
<td>فعل</td>
</tr>
<tr>
<td>خرج</td>
<td>مخرج</td>
<td>مفعل</td>
<td>فعل</td>
</tr>
<tr>
<td>قن</td>
<td>مقبل</td>
<td>مقبل</td>
<td>فعل</td>
</tr>
<tr>
<td>سرب</td>
<td>مشروب</td>
<td>متغل</td>
<td>فعل</td>
</tr>
<tr>
<td>قرب</td>
<td>مقارب</td>
<td>متغل</td>
<td>فعل</td>
</tr>
<tr>
<td>عزل</td>
<td>متغل</td>
<td>متغل</td>
<td>فعل</td>
</tr>
<tr>
<td>رقب</td>
<td>مريب / مرتقب</td>
<td>متغل / متغل</td>
<td>فعل</td>
</tr>
<tr>
<td>سود</td>
<td>مسود</td>
<td>فعل</td>
<td>فعل</td>
</tr>
<tr>
<td>غرق</td>
<td>مغرق</td>
<td>مستقبل</td>
<td>مستقبل</td>
</tr>
<tr>
<td>خرج</td>
<td>مستخرج</td>
<td>مستقبل</td>
<td>مستقبل</td>
</tr>
<tr>
<td>فتح</td>
<td>مفتح</td>
<td>فعل</td>
<td>فعل</td>
</tr>
<tr>
<td>غدر</td>
<td>مضور</td>
<td>فعل</td>
<td>فعل</td>
</tr>
</tbody>
</table>

Table 3.8 – A subset of T-Derivations

<table>
<thead>
<tr>
<th>Stem</th>
<th>Example</th>
<th>T-Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>مثال الكلمة</td>
<td>مثال</td>
<td>مصدر ثاني</td>
</tr>
<tr>
<td>خرج</td>
<td>تخرج</td>
<td>فعل</td>
</tr>
<tr>
<td>بدل</td>
<td>تغل</td>
<td>فعل</td>
</tr>
<tr>
<td>قسم</td>
<td>تقاسم</td>
<td>فعل</td>
</tr>
<tr>
<td>أصل الكلمة</td>
<td>مثال</td>
<td>المشتتات</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>رقص</td>
<td>رقص</td>
<td>فعل</td>
</tr>
<tr>
<td>شغف</td>
<td>شغف</td>
<td>فعل</td>
</tr>
<tr>
<td>طويل</td>
<td>طويل</td>
<td>فعل</td>
</tr>
<tr>
<td>كمال</td>
<td>كمال</td>
<td>فعل</td>
</tr>
<tr>
<td>صبح</td>
<td>صبحية</td>
<td>فعلية</td>
</tr>
<tr>
<td>مشغل</td>
<td>مشغل</td>
<td>إفعال</td>
</tr>
</tbody>
</table>
Chapter 4

Solution Approach

This chapter describes in detail the approach used to solve the problem at hand. The difficulties that lie behind the Arabic language due to its morphological rules as well as the methods that will be followed to overcome them will be discussed. The chapter also presents a graphical and literal description of the system’s components, its rules, and the database structure.

4.1 A Proposal for Arabic Search

The difficulties in dealing with the Arabic language lies in the ability of manipulating terms and words and coming up with derivations from the same stem, which will be discussed later on. For example, let us consider the Arabic language’s main paradigm فعل (fāla), from which with the addition of the appropriate prefix, infix, and suffix all verbal and nominal conjugations can be derived. For example, among the nominal conjugations we have the active participle فعل (fāl), the passive participle اسم الفعل المضارع (esem el mafoul), and the adjective فعل (fāla), and all the verbal conjugations: the singular masculine past tense verb فعل (fāla), the singular masculine present tense verb يفعل (yafalou) by adding the at the beginning to denote the present etc.

What was mentioned earlier above are few examples for the conjugations of the singular form of verbs only. The above mentioned examples form the simple cases of conjugating a verb
in Arabic. However, there exist other special cases and irregularities specifically in the case of plural terms in the Arabic language i.e. the irregular plural or جمع تكسر (jame' taksir). For example, to render a singular feminine noun that ends with a مه (tâ') into plural we change the مه (alif tâ') like the word سيدة (sayyida) which becomes in the plural form as سيدات (sayyidat). However, the irregularity is that the plural of its masculine form is not irregular and takes the form of لسيد (asyad).

Unlike the English language, pronouns add complexity to the stemming process of the Arabic terms. The reason behind that is that pronouns appear in the form of suffixes, and hence they are part of the grapheme itself. For example, the singular masculine pronoun هو (houwa or 'he' in English) can be attached to the singular past tense verb كتب (kâb) in the form of a suffix and thus becoming كتبه (kâbhou). The morphological analysis becomes even more complex when we realize that more than one pronoun can express the conjugation (and is built in the verbal morphology) while the other pronoun, the object pronoun, is attached to the same verb such as كتبته (takâbhou). The ض (ta') prefix and the و (waw) is for the plural conjugation of the present tense for the third person masculine plural, while the ل (hâ') is for the third singular masculine object pronoun. In addition to these prefixes, infixes, and suffixes, we find that other affixes exist such as the حروف (sîn) in مكتوبته (sataktoubahou), or ل (làm) in كل (kal) in كلب (kâlbâder) etc.

In view of all that, we propose instead of doing a syntactical analysis to the text being indexed that we go directly to the stem. Our strategy tries to reach and extract that stem as fast as possible. To do this, we bypass the process of syntactical parsing. This will result in excessive search results which might be considered by some users as irrelevant and not necessary.
However, by sorting the results in decreasing order of relevancy, the most relevant hits will appear at the top while the least relevant will be at the bottom and thus solving this problem.

As a result, we propose a flexible auto indexing system capable of accommodating all the above mentioned cases. This flexibility allows us to maintain the system without the need to have an internal access to the system code and allows us to add any other morphological rules that we may discover. In order to make this possible, a rules compiler was integrated into the system. Chapter 6 will elaborate more about this component.

4.2 Extensibility in Document Formats

As it was mentioned earlier, we propose a solution for the diversity in document formats issue which is the use of a parser class. This will minimize the degree of user intervention and will ease up the process of integrating a new file format parser into the system.

The Internet “stores” terabytes of data that are in different locations and in different formats such as Text files, PowerPoint slides, Excel sheets, HTML documents, Word documents, etc. The system, mainly the indexer, must be able to parse all of these documents and go through each and every one of these formats and be able to index their contents. To do that it must be already equipped with a mechanism that would enable it to do so. The real challenge is when a new format is to be added to it while the system is up and running. New formats come up each day and some contain critical and essential data. The user must be able to add a parser for this new format with minimal effort.
Therefore, the solution is to separate the parsing modules from the indexing module to ensure that the indexer will not be affected by the parsers. The indexer resides inside the server component, which is in turn running on a separate machine other than the machines running the parsers. This client/server architecture relieves the server, and thus the indexer, from the burden of running the different parsers and paves the way to easily integrate any new file format parser. A Parser class has been created for this purpose. This class handles all the communications taking place with the server. The user instantiates an object of that class and uses it to communicate with the server. He/she then only has to add to this new parser the code that opens the targeted document and parses it for useful content. Appendix C provides detailed description about this class.

In order to test the validity of the solutions proposed i.e. the parser class and the rules compiler, we have built a prototype consisting of a text parser client and a server running with an SQL Server as an RDBMS. In what follows, we present a detailed description for each and every component in the system. Additionally, we present the structure of the database that stores the indexing rules.

The user of the proposed solution does not have to be committed to this design. He/she can feel free to replace the DB by inverted files, B-Trees, or whatever data structure that helps in increasing the performance of the system since performance is not one of the issues that this study takes care of.
Figure 4.1: The Proposed Prototype Search System
4.3 Extensible Arabic Full-Text Indexer

Figure 4.1 describes the system using a graphical representation. In order to achieve a high level of modularity and independence, the components were created separately. The independence of the components provides a high level of flexibility in such a way that the client component can be replaced at any time while the server continues to function properly. This independence allows us to update the client at any time without affecting the performance of the whole system and thereby ensuring the extensibility in file formats. However, this does not mean that the components do not interact. The interdependence level is high. The client component interacts with the server. The objective of all the clients is the same: to parse documents. However, each one of them is parsing a different document with a different format. The output of these objects is then redirected to the server which in turn submits them to an internal module: the indexer.

The server runs forever. It waits on a certain port for streams of bytes from the clients. If at any time the client component stops, the server continues to function by simply waiting. No disruptions or malfunctioning occurs. This provides us with the ability to upgrade the client parsers at any time in order to add a new document format parser. This newly integrated feature insures that extensibility is maintained in file formats. At any time, any user can develop a new parser and simply connect to the server and start parsing, or he/she can stop a parser from functioning, maintain it, and then restart it again. Modularization is the key behind this "least effort possible" mechanism. If everything was wrapped together, then it would definitely be a very tedious, time consuming, and exhaustive job to upgrade any component. However, by
simplifying this task through extensibility, it has definitely become a very simple and easy process.

Extending the capabilities of the client component can be done while other parsers are running. User intervention in adding a new parser has been minimized to the minimum. A Parser Class has been created to be used in the process. What the user has to do is to simply instantiate an object of this class and add then use it to communicate with the server. The part that he has to add is simply the code for parsing the format he's concerned about.

4.3.1 The Database component

In what follows, a detailed description for the structure of the database and how data will be stored in it shall be presented. We have used a database to store the rules used by the indexer, such as stop list terms, suffixes and prefixes, non-derivable terms etc. Any changes done to the indexing rules are directly made to the database and thus ensuring that the indexer is always up to date at runtime. The database is also used to store the term phrases generated by the indexer, and are assigned to their corresponding documents. In what follows we describe each of the database tables as well as the corresponding entity relational diagram.
4.3.1.1 Stop List Terms

Stop list terms are noise words that are used frequently in any sentence such as 'the', 'in', 'at', 'from', 'to', etc. These terms are discarded by the system once encountered. The StopList table stores these terms. These terms are "useless" to the indexer in the sense that they don't add to the meaning of the text parsed anything and therefore they should be discarded. For example, أحرف النصب (ahraf al naseb) such as أن, لن, كي (an, lan, kay) are all stop list terms but belong to the category أحرف النصب. They are very useful in maintaining a coherent structure of the English/Arabic text yet they do not add to the meaning of it.
4.3.1.2 The Patterns

The Rhymes table stores the patterns that will be used in stemming and indexing words and verbs along with the corresponding category to which this pattern belongs i.e. is it a pattern for a noun or verb? For example, the patterns for the five verbs or الأفعال الخمسة (Al Af'al al khamsa) are used with verbs (Table 3.4 contains a list of these verbs). These verbs form a morphological representation that is fixed i.e. any verb that belongs to this list should rhyme with them whether they are in the present, past or imperative tense. Therefore, upon deciding the nature of a word, whether a verb or noun, only the patterns that apply to that word will be used. The Rhymes table shall contain all the Patterns that belong to that category i.e. يفعلون, يفعلان, يفعلان etc. (taf'alalan, yaf'alalan, taf'alaloun, yaf'alaloun). All the other categories such as الأسماء الممنوحة من الصرف or الأسماء الممنوحة من الصرف or الأسماء الممنوحة من الصرف or الزيادات العشرة (azziyadat al ashara, al asma' al mammou'a min assarf) are all entered into that table and the administrator can add/modify/delete any new categories.

4.3.1.3 The Non Derivables

This table contains the values for words for which the stemming process should not be applied. For example, الأسماء الممنوحة من الصرف, أسماء العلم, etc (asma' al alalam, al asma' al mammou'a min assarf) these terms should be stored in the database as is. Therefore, when the system detects one of these it directly stores it as is in the database and assigns its value as its stem.
4.3.1.4 The Suffixes and Prefixes

This table holds the values of all the suffixes and prefixes that could be attached to nouns and verbs. Again, the suffixes and prefixes can be attached to nouns or verbs. In both cases, they have to be isolated from the term to which they are attached.

4.3.1.5 The Stems

This table holds the results of the indexber. Whenever a term is submitted to the indexer, the indexer should start isolating all the unnecessary attached terms to this term and then try to guess the original stem of this word. Once figured out, it stores the original word along with its stem in this table. This feature has been added to the indexer in order to render the stemming process faster and more efficient instead of performing the stemming process to each and every received word.

4.3.1.6 The Documents

This table shall hold the references to each newly submitted document. The server will assign a unique ID to it and send it back to the parser. The parser then will use this ID to segregate its sent stream of bytes from others sent by the other concurrently running parsers.
4.3.1.7 The Words References

This table stores the words that have been processed along with their reference docID. Each word will have a count which will allow the system to identify how many times this word has been encountered in the document it was found in. In addition, the system will keep track of the positions that this word has been encountered at. This information will help later on in determining and calculating the degree of relevance that this word has in its contextual document. For example, to calculate the relevance of a word with respect to the document it is found in we base our relevance calculation on three factors: The number of times this word has been encountered literally in the text, the positions at which it was found, the number of times the stem of this word has been encountered in this text, and finally the positions at which this stem was found. The higher the counts of the term and the stem are the higher the relevance and vice versa. The closer the positions of the term and its stem are the higher the degree of relevance of that term will be.

4.3.1.8 The Stems References

This table is very much similar to the previous one. However, it stores the stems for each word rather than the words themselves. This table is different from the Stems table in that it is used to help in determining the degree of relevance that each word has in the system too.

The Stems table is used to increase the speed of the indexing process. In that table, we shall find words along with their stems. Therefore, whenever the indexer encounters a new word, it will come and search for it in that list to see if it already exists there. If it does, then the stemming process is directly applied to it and subsequent modifications shall be done to words references.
and stems references tables. If it doesn’t then it passes this word to the stemming module and subsequent records are added to the Words references table. The Words references contain data about all the words and to which document each and every word belongs. If the word is a new one, an entry is added to the Words references table along with the word’s stem specifying that this word has been encountered in that document at position X.

However, if a word has been already parsed and is found to exist in the Stems, its corresponding entry in the Words references table is modified as follows: The count field is increased by 1 and to the position field the new position is concatenated. At the end, the positions field will look somewhat like that 1, 3, 45, 56. And these are the corresponding positions in which this word has been encountered in the document.

4.3.2 The Server Component

The server component is the core component of the whole system. It contains two subcomponents or modules: the indexer and the relevance calculation module. The indexer’s job is to index the documents submitted to the system. The relevance calculation module’s job is to calculate how relevant each word is in a document.

At the heart of the core component of the system lies the indexing module. After receiving a word, the server passes it to the indexing module which processes it and extracts its original stem and then will store the corresponding results in the underlying tables. The next
chapter will present a detailed description of the indexer subcomponents and the techniques followed in the stemming process.
Chapter 5
Rules Compiler

5.1 Introduction

This chapter describes in detail the rules compiler. The rules compiler is a separate entity of the system. It runs alone and does not require any intervention from the system. It interacts directly with the database in such a way that the rules existing there are modified on the fly and thus instantly affecting the indexing process.

5.2 Motivation

Introducing this component into the system has minimized human intervention in the extensibility process. This component allows the user to dynamically modify the rules used in the indexing process without the need to rewrite any part of the indexer module found in the server component or recompiling it. The rules compiler compiles a set of rules that are stored in a file and then adds them to the system's database.
5.3 Description

In order for the rules compiler to 'compile' the rules into a format that the indexer understands and works with they have to comply with the following format:

The EBNF representation of the above rules is:

\[
\text{Rule} := \langle \text{Priority}\rangle \ \text{IF} \ \langle \text{Condition}\rangle \ \langle \text{Value}\rangle \ (\text{OR} \ \langle \text{Value}\rangle)? \ (\text{AND} \ \langle \text{Condition}\rangle \ \langle \text{Value}\rangle \ (\text{OR} \ \langle \text{Value}\rangle)? \ \text{THEN} \ \langle \text{Action}\rangle \ \langle \text{Value}\rangle+ \ (\text{AND} \ \langle \text{Action}\rangle \ \langle \text{Value}\rangle+)?
\]

Where,

\[
\langle \text{Priority}\rangle := ['SL'|'ND'|'PF'|'SF'|'RH'|'ST']D^6
\]

\[
\langle \text{Condition}\rangle := '\text{TERM-IS-EQUAL-TO}' | '\text{STARTS-WITH}' | '\text{ENDS-WITH}' | '\text{RHYMES-WITH}' | '\text{STEM-ENDS-WITH}|\text{STEM-IS-EQUAL-TO}'
\]

\[
\langle \text{Value}\rangle := ['\text{A-}\text{Z}'] + 6
\]

\[
\]

---

5 This is the priority argument used to organize the order in which the rules will be applied.

6 Value is the a letter or sequence of letters that the indexer compares with or search for its/their existence at a certain position.
The general form of a rule is:

\[ \text{<Priority> IF <Condition> <Value> [OR <Value> OR <Value>... ] [AND <Condition> <Value>...] THEN <Action> [<Values List>] [AND {<Action>} [<Values List>] AND ]...} \]

The compiler reads a text file containing these rules. It checks the syntax of the file. It will try to make sure that the rules are syntactically correct. It will do this while parsing them. If a rule is found to be syntactically correct then it is entered into the rules table in the DB and subsequent modifications will be made to the underlying database. Otherwise, the rule will be discarded.

The modifications done to the database follow a certain translation standard which will translate these rules from plain English into subsequent records in the underlying tables. The indexer will use these tables to perform the stemming operations.

The <Priority> argument will help the indexer identify which indexing rules to apply before other ones. The higher the priority the more likely that the rule will be applied before the ones with a less priority.

5.4 Examples

The following examples describe some examples of the rules submitted to the rules compiler and how they are then processed.
Example 5.4.1:

\text{RH6 IF RHYMES\_WITH ينطخون THEN EXTRACT\_STEM AND TERM\_IS\_A\_VERB}

Compilation goes as follows:

The RH6 argument indicates that this rule deals with \textit{rhymes} and its priority is 6 i.e. the other rules with priority 5 and less will be applied after that rule is applied. The RHYMES\_WITH indicates that an action is about to be done to the Rhymes table. The EXTRACT\_STEM action tells the compiler to add the ينطخون pattern to the Rhymes table and the TERM\_IS\_A\_VERB signifies that this pattern is used with verbs.

Example 5.4.2:

\text{RH1 IF TERM\_IS\_EQUAL\_TO ينطخ THEN RHYME\_IS\_EQUAL\_TO ينطخ}

The RH1 indicates that this rule applies to \textit{rhymes or patterns} and it has a low priority equal to 1. The TERM\_IS\_EQUAL\_TO indicates that the compiler should save the <argument> value for later use depending on the <action> specified. The RHYME\_IS\_EQUAL\_TO action indicates that the compiler should add this pattern to the Rhymes table and perform a direct stemming on the argument to extract the stem accordingly. This rule is mainly used with terms with irregular patterns.

Example 5.4.3:

\text{ND1 IF TERM\_IS\_EQUAL\_TO ثَفَانَز THEN SAVE\_AS\_IS}
The ND1 indicates that the rule deals with *Non Derivable* terms and has a low priority equal to 1. The TERM_IS_EQUAL_TO indicates that the compiler should save the <argument> for later use according to what the <action> indicates. The SAVE_AS_IS action indicates that the compiler should save this value in the "Non Derivables" table. This rule is mainly useful in the cases of proper and foreign nouns.

**Example 5.4.4:**

ST1 IF TERM_IS_EQUAL_TO نسوة THEN STEM_IS_EQUAL_TO إمرأة

The ST1 indicates that this rule deals with *StopList* terms and has a priority equal to 1. The TERM_IS_EQUAL_TO indicates that the compiler should save the <argument> for later use according to what the <action> indicates. The STEM_IS_EQUAL_TO indicates that the <argument> is a special case and hence should be treated in a different manner. In the case of this example, its denotes that the term نسوة is an irregular case for the plural (what is referred to before as جمع التكسير) and thus its stem cannot be extracted according to the regular rhyming rules but rather has been mentioned explicitly.

**Example 5.4.5:**

SL2 IF TERM_IS_EQUAL_TO ل THEN DISCARD AND NEXT_WORD_IS_A_VERB

The SL2 denotes that this rule deals with *StopList* terms and has a priority equal to 2. The TERM_IS_EQUAL_TO indicates that the system should save the <argument> for later use as the <action> section denotes. The DISCARD denotes that the <argument> is a *Stop List* term and thus has to be 'discarded'. Moreover, the NEXT_WORD_IS_A_VERB denotes that the term...
following the <argument> is a verb and thus has to be treated according to the indexing rules for verbs.

Duplicate rules will be automatically discarded by the compiler. Appendix A presents a detailed description of what each <condition> and <action> means and does. In what follows, we will present an example of the results of indexing and stemming a paragraph to clarify how things work. Let us consider the following Arabic passage:

يعتقد البعض أن الصخور أثناء صلابة وبسهب كسرها . نعم ، إن بعض الصخور كذلك لكن لا كلاها . إن الرمل والحمض والطباشير والصلصال صخور في نظر العالم الجيولوجي . ولو أن تكونت قبضة رمل على النفايات وتركتها تساب بين أسابيع ، فإنها تكون قد تكونت صخورة شبيهة بالحجر الرملي الصلب الذي يصبح للبناء . و هناك أنواع عديدة من الصخور تكربت بطرق مختلفة . و يقسمها علماء الجيولوجيا إلى ثلاث مجموعات: الصخور النارية ، الصخور الرسوبية ، الصخور المتحولة . و يجب علينا أن نستطيع هذه الأسماء لأنها الأسماء التي يستعملها الجيولوجيون دائما . و الصخور المجموعة الثالثة أو الصخور المتحولة هي الصخور التي بدأت نارية أو رسوبية ثم تغيرت وفعل الحرارة الشديدة أو الضغط أو الأثواء تغيرا كبيرا حتى صارت شيئا آخر . و على سبيل المثال نجد أن بعض الحجر الرملي تغيرت إلى حجرة أشد صلابة هي المرو أو الكوارتز ، وأن حجر الجير تحول إلى رخام ، وأن صخر سنجر تنقلت ودعي تطين السفحي قد تحول إلى إردوء أو الحجر المشتق ، وهو صخر قليل سريعة الأتكمار.

Processing this passage goes as follows:

يعتقد

This term will be processed according to the following rule:

RH4 IF RHYMES_WITH بقع THEN EXTRACT_STEM

Hence, the output will be: 

بعد

To the

PF2 IF STARTS_WITH أ THEN REMOVE_LETTERS → بعض

53
SL3 IF TERM IS EQUAL TO بغض THEN DISCARD ⇒ Nothing will be indexed

ولو أنك في لا ما، إن، نعم كنتك أن

SL3 IF TERMS IS EQUAL TO أن OR OR OR لك أن OR OR في OR OR THEN DISCARD ⇒ Nothing will be indexed

الصومال

PF IF STARTS WITH ال THEN REMOVE LETTERS ⇒ صخر
RH4 IF RHYMES_WITH قول THEN EXTRACT_STEM ⇒ صخر

الصومال

RHH IF RHYMES_WITH أفعال THEN EXTRACT_STEM ⇒ شيء

صلاة

SF1 IF ENDS WITH ؤ THEN REMOVE LETTERS ⇒ صلب

وصب

RH4 IF RHYMES_WITH فعل THEN EXTRACT_STEM ⇒ صعب

كسرها

SF2 IF ENDS WITH ئا THEN REMOVE LETTERS ⇒ كسر

كلها

SL4 IF TERM IS EQUAL TO كلها THEN DISCARD ⇒ Nothing is indexed

رم

PF2 IF STARTS WITH ال THEN REMOVE LETTERS ⇒ رمل

وصي:

PF3 IF STARTS WITH و THEN REMOVE LETTERS ⇒ حصى

وف الطباشير

PF3 IF STARTS WITH وال THEN REMOVE LETTERS ⇒ طباشير
RH5 IF RHYMES_WITH فعلين THEN EXTRACT_STEM ⇒ فعل

ي المالصل

PF3 IF STARTS_WITH وال THEN REMOVE LETTERS ⇒ صلب
ND1 IF TERM IS EQUAL TO صلب THEN SAVE AS IS ⇒ صلب

فصول

RH4 IF RHYMES_WITH قول THEN EXTRACT_STEM ⇒ صخر

نظر

RH3 IF RHYMES_WITH فعل THEN EXTRACT_STEM ⇒ نظر

العالم

PF2 IF STARTS WITH ال THEN REMOVE LETTERS ⇒ عالم

54
RH4 IF RHYMES_Will THEN EXTRACT_STEM → علم

PF2 IF STARTS_WITH و THEN REMOVE_LETTERS → جيولوجي

ND1 IF TERM_IS_EQUAL_TO THEN SAVE_AS_IS → جيولوجي

و زا:
PF1 IF STARTS_WITH و THEN REMOVE_LETTERS → او
SL2 IF TERM_IS_EQUAL_TO THEN DISCARD → Nothing is indexed

يتقلت:
SF1 IF ENDS_WITH ت THEN REMOVE_LETTERS → تقلب (moctal al akher) تقلب فعل محتل and requires some adjustments. This is left for future enhancements. The correction should be to replace the (waw) in the middle with the original letter (alef) since some derivations of the original term ن أ / ن أ require that the letter أ in the middle be replaced by أ.

...

And so on and so forth.
Chapter 6
Experimental Results

In this chapter, we will show the results of some tests that were done using a prototype for the proposed solutions. We start with a basic set of rules that will be gradually enhanced during the system's lifecycle. As a measure of speeding up things, the system stores the results for later use. This increases the performance of the system in a very significant manner as we will see later on.

After processing each word, the stem is extracted and is related directly to that word. However, before the system moves to another word, it saves this result in table "Stems." In this way, if the indexer encounters later on the same word, be it from the same text or another, it does not need to reprocess it again but rather goes directly and fetches its stem from the "Stems" table. Consequently, the longer the system runs, the more processed words will be stored. As a result, the chance of finding the stem for the newly presented words increases correspondingly. This eliminates the need to reprocess them again. Figure 6.1 shows how the performance of the system is dramatically enhanced by the use of such a mechanism.
6.1 Indexing Duration

![Graph showing indexing duration](image)

**Figure 6.1 Indexing Duration**

A random sample of documents was picked up from various sources. These documents deal with different issues and are of different sizes. When we refer to sizes we mean the number of pages or more accurately, the word count. The horizontal axis reflects the count of these words. Generally speaking, as the number of words increases, the time needed to perform the stemming and indexing increases as well. However, by introducing the mechanism of saving results, indexing time has been reduced significantly. The vertical axis shows the timeline of the duration taken to index these documents in seconds. The upper line indicates the duration taken before "using" this "saving" mechanism. The lower line shows how the time has been decreased after it was used. On average, the results indicated that the indexing speed has been enhanced by approximately 75 percent.

Figure 6.2 shows the accuracy of search hits performed while indexing. The more documents are indexed the higher the accuracy becomes.
6.2 Search Hits

![Precision Graph](image)

**Figure 6.2 Search Hit Results**

The introduction of the rules compiler helped in empowering the indexing process. At runtime, the user can index documents. At runtime, the user can run the rules compiler and add more rules. The user can delete/modify existing rules from an interface built for that purpose. As shown in Figure 6.1, it is assumed that the time taken to index documents will not be proportional to their sizes only, but rather will be affected with decreasing factors such as the already indexed documents. The more documents that are indexed the "faster" the indexing process will be. Speed will also increase. Indexing similar documents will be even faster since it will be very much likely that similar words exist in them. Another factor that can enhance the indexing performance is the addition of indexing rules such as "Non-derivable" terms since these do not require stemming but are rather taken as is. The client-server architecture relieved the server from the burden of document formats. At any time a new client parser can be modified or created and started. The abnormal termination of any client does not affect the server.
The scheme proposed can offer a very good ground to start building an Arabic full-text indexed search engine similar to Google. However, it is noted that indexing Arabic documents is always more time consuming than English texts due to the complexity of the Arabic language. Pronouns are not only detached but can also come in the form of attached pronouns. Moreover, in addition to the singular and plural states, the Arabic language has the *moukhama* state. This is the state of nouns and verbs referring to two entities. Moreover, the Arabic language has besides the regular plural state another irregular one. This feature in addition to the already mentioned ones and others make the job of indexing Arabic documents one of the most difficult tasks to be done.
Chapter 7
Rules Compilation and Indexing in Hebrew and Persian Languages

The extensibility in this study has proven to be applicable not only in terms of parsers for file formats and indexing rules but also in terms of the languages indexed. Not only the user can extend the file formats and the rules used by the indexer but also he/she can, with minimal effort, index other languages. No changes are required for any of the components of the system. The only changes that have to be done are to the rules. By simply removing the already existing ones and replacing them by a new set, the user can directly start indexing a new language. Each language has its unique specifications that require some additions or changing for the existing rules. Therefore, the additions or updates have to be performed in order to render the rules compiler fully customized for the language indexed.

7.1 The Persian Language

The first test was done using the Persian language. Its rules are similar to the one used by the Arabic language. However, the Persian language is much simpler in terms of syntactic and semantic rules. There is no masculine, feminine, or Mounhanna (the state for two entities) states. There are too many suffixes and some prefixes and the stop list for that language is quite long.

---

7 The rules used in these tests are not 100% accurate i.e. there might be other cases which contradict these rules. This is due to our limited know-how about them. However, our attempts to index these languages are done simply to prove how extensibility has propagated from adding/updating the Arabic indexing rules to totally removing them and replacing them with new ones and directly resume indexing. Therefore, with some customization to these rules it shouldn’t be a very hard task to apply this study to other languages.
Moreover, there are no patterns in Persian. Therefore, shifting to this language was simple and fast.

Here are some examples of how simple shifting to the Persian language is. Notice that the same rules were used and the only changes done were to the <Value> and <Argument> sections.

- **SL2 IF TERM_IS_EQUAL_TO دو THEN DISCARD**
  
  دو is a noisy word similar to the 'two' in English which considered as a stop list term and thus has to be discarded.

- **SL4 IF TERM_IS_EQUAL_TO هکدام THEN DISCARD**
  
  Processing this term is similar to the previous example.

- **SF2 IF ENDS_WITH ان OR ا THEN REMOVE_LETTERS**
  
  These terms are the equivalent to the 's' in English which should be removed to revert to the singular format of the term having them as suffixes.

- **ND4 IF TERM_IS_EQUAL_TO این THEN SAVE_AS_IS**
  
  This term is a proper noun and thus has to be saved as is. No processing is required for it.

- **PF1 IF STARTS_WITH ن THEN REMOVE_LETTERS**
  
  Similar to 'not' in English. However, this term is attached as a prefix and has to be removed from the term in which it has been encountered.

- **SL2 IF TERMS_IS_EQUAL_TO وی OR OR OR این THEN DISCARD**

---

8 دو (do) in Persian means two
9 کدام (kadam) in Persian means 'which'
10 ان (an) and ا (an) are used to indicate the plural. Equivalent to the "s" in English
11 این (ayn) is a proper noun
12 ن (noon) is used for denial. Similar to not in English
13 دو, وی, ان, این are equivalent to the pronouns you, he, you(plural), and we in English.
7.2 The English Language

Shifting to the English language was as easy as shifting to the Persian Language. The syntactic and grammatical rules in English are simple. This is due to the fact that the English language is an easy to learn language in terms of grammar. For example, in general, to shift from singular to plural we add an "s." The feminine and the masculine tenses are treated the same way and there is no such thing as "rhyming." To move a verb from present to the past tense we add, in general, and "ed" at the end of the word. To move to the past perfect we add an auxiliary, which in turn is treated as a stop list term, and thus is discarded by the indexer, and an "ed" to the end of the verb.

Here are some examples of how simple shifting to the English language is. Again notice that the same rules were used and the only changes done were to the <Value> and <Argument> sections.

- **SF1 IF ENDS_WITH s THEN REMOVE_LETTERS**
  This applies to the regular plural such as apples, cases, texts, etc. This rule reverts the plural term to its singular format.

- **SF2 IF ENDS_WITH ss THEN DONT_REMOVE_LETTERS**
  This applies to the special cases for terms that end with two s such as 'stress', 'press', 'confess', etc. This prevents the previously mentioned rule from removing the last s since this 's' is not the sign for the plural but rather is part of the original word.

- **ND1 IF TERM_IS_EQUAL_TO english THEN SAVE_AS_IS**
This rule is used with proper nouns such as country names, proper nouns for people, language names such as English, French, Spanish, etc. These words should not be changed in any way but rather should be saved and indexed as they are.

- **SF3 IF ENDS_WITH ies THEN REPLACE_WITH y**
  This rule takes care of the plural form for terms that end with a 'y' such as 'entry' which becomes 'entries' in the plural format. To restore the singular format for this term we replace the 'ies' with a 'y'.

- **SL1 IF TERM IS EQUAL TO from THEN DISCARD.**
  This rule takes care of stop list terms or noisy words which do not add to the meaning or the sentence and are very frequently used and thus should be discarded.

- **SL3 IF TERM IS EQUAL TO has OR have OR had THEN DISCARD.** This is similar to the previous example.

- **ST1 IF TERM IS EQUAL TO mice THEN STEM IS mouse.**
  This example takes care of the special plural cases similar to the ones in Arabic (specifically the example of قِصَارُةٌ and جُرَاءٌ) where the plural of a term is totally different from its singular format and thus reverting to that singular term does not follow and regular stemming rule but rather has to be mentioned explicitly.

### 7.3 The Hebrew Language

The Hebrew language, on the other hand, is very much similar to the Arabic language. It has everything that the Arabic language has. It has masculine, feminine, plural, mouthanna, patterns, suffixes and prefixes, stop list terms etc. Shifting to that language was as hard as using
the Arabic language. In fact, Hebrew is even more complex in terms of morphological representation. However, it is very similar to Arabic in terms of grammatical and syntactical rules.

Here are some examples of how simple shifting to the Hebrew language is. Again notice that the same rules were used and the only changes done were to the <Value> and <Argument> sections.

- ND1 IF TERM IS_EQUAL_TO יָשָׁר THEN SAVE AS IS\(^{14}\)
  יָשָׁר is a proper noun and therefore it should not be processed but rather saved as it is.

- SL3 IF TERM IS_EQUAL_TO יֵעַ THEN DISCARD \(^{15}\)
  יֵעַ is equivalent to 'you' in English and thus has been treated as a stop list term.

- SF2 IF ENDS WITH נ THEN REMOVE LETTERS AND TERM IS A NOUN\(^{16}\)
  These are suffix letters that are added to a noun to represent its feminine state. Therefore, they should be removed.

- PF1 IF STARTS WITH ט THEN REMOVE LETTERS AND TERM IS A NOUN
  This letter is equivalent to "the" in English and (al) in Arabic and thus it should be treated as a prefix.

\(^{14}\) יָשָׁר (yashara) a proper noun

\(^{15}\) יֵעַ (ate) is equivalent to 'you' in English

\(^{16}\) these letters indicate the feminine state of a noun
SL1 IF TERM_IS_EQUAL_TO ? THEN DISCARD (This term is equivalent to the "from" in English)

This is a stop list term and thus it should be discarded.

The rules given above as examples are a subset of the rules used to test the indexing of Persian, English, and Hebrew. As it was the case with Arabic, the rules used for testing are not complete i.e. They do not cover all the grammatical rules for each of these languages. However, making this test has helped us conclude that the solution proposed i.e. the rules compiler has helped us add a new dimension to this study in terms of the languages indexed. Future enhancements and the addition of new compilation rules will render the compiler and the indexer more robust, accurate, and complete.
Chapter 8
Conclusion and Future Enhancements

8.1 Conclusion

Any file format can be indexed using the proposed solutions. If its parser already exists then it is used to parse its contents and send them to the indexer. Otherwise, adding a new parser can be done with minimal effort. Once indexed then the contents of this file become available for searching. If the user feels like adding new rules to the indexer then he/she can simply submit a text file with these rules. The indexer will compile these rules and will then perform the corresponding updates and changes to the already existing ones. This should help in tuning the accuracy of stemming and thus indexing. The user can change the design of the prototype proposed to tune up performance by using B-Trees or inverted files instead of the database. The system can be tested against other languages. It has already been tested on not only Arabic but also English, Persian, and Hebrew and proved to be applicable to a great extent with minor customizations done each according to the specifications and needs of the language being indexed. Future enhancements can also include the applicability of this study on other languages too and thus rendering it more "complete."

8.2 Future Enhancements

This system is subject to further enhancements in the concept behind it and the implementation done for that purpose. In addition, future enhancements can be performed to the
already built prototype in such a way that renders the indexing process even faster and more efficient. Additional rules can be added to the already existing ones. This renders the stemming process faster and more accurate. An example for such rules is the one that handles the irregular plural forms of *al aṣfaal al moṣatalat al akher* اللائفعال المئلة الآخر. One of the problems with these verbs is that their derivations change the morphological representation of their original stem. For example, let us consider the verb رمي (rama or throw in English). The morphological representation of its present conjugate is رمي (yarmi). Unlike the regular verbs, the present tense of this verb has changed the last character, the كاف مقصورة (Ka'f Makṣūra) into a أ (a') and therefore when the indexer attempts to extract its stem by matching it against the pattern يفعل (yaf'al) the stem will be رمي instead of رمي. One way to solve this problem is the use of the following rule:

\[
\text{ST1 IF STEM_ENDS_WITH ي THEN REPLACE_WITH ي}
\]

And thus rectifying the problem. Regarding the other morphological representation for the three letters word أ رمَي these are some suggestions and combinations of rules which help us identify their stem:

رمي (rama, equivalent to *throw* in English):

\[
\text{RH3 IF RHYMES_WITH فعل THEN EXTRACT_STEM}
\]

رميا (ramaya equivalent to *throw* in English but this applies to the past double case):

\[
\text{RH4 IF RHYMES_WITH فعل THEN EXTRACT_STEM}
\]

\[
\text{ST1 IF STEM_ENDS_WITH ي THEN REPLACE_WITH ي}
\]

رموا (ramaw equivalent to *throw* in English but this applies to the past plural case):

\[
\text{RH4 IF RHYMES_WITH فوا THEN EXTRACT_STEM}
\]
Here is where this type of verbs has broken the regular rule another time. Instead of having their original stem rhyme with the three letters word /فعل (facala), they rhyme with the truncated two letters word /ف (faci) and then the problem that needs to be solved would be to find out the right character that was truncated. In case of the verb رم (rama) this letter would be the letter أ (alef maksoura). One rule that would solve this specific problem would be for example:

```
IF STEM_RHYMES_WITH فك THEN ADD_LETTER أ
```

Although this might not be the perfect solution for this type of verbs (and this is the reason why this was mentioned in the "future enhancements" section), however, this solves the problem. Similarly, this applies to the other special morphological representations such as رم which rhymes with the rhyme فك (facat), the derivation رم (ramata equivalent to threw in English but applies to the feminine past double tense) which rhymes with ف (facata), and أرم which rhymes with أفي (ifci) instead of أفق (facalat), فم (facalata), and أفق (ifcal). In other words to perform stemming for these three terms we apply the following rules:

```
R3 IF RHYMES_WITH فك THEN EXTRACT_STEM AND ADD_LETTER أ
```
```
R4 IF RHYMES_WITH ف (facata) THEN EXTRACT_STEM AND ADD_LETTER أ
```
```
R3 IF RHYMES_WITH أفي THEN EXTRACT_STEM AND ADD_LETTER أ
```
The above mentioned morphological representations are some irregular cases which require further research and analysis. The research and analysis are to be done for all the أفعال المعطة (Al asil al moctalla) to come up with complete and applicable solutions. As for the other morphological representations of the three letters word ارّمي, which rhyme with the complete morphological representations of the three letters word فعل, we treat them the regular way.

There might be other cases where the ج is part of the original stem and this contradicts the above mentioned rule and requires more investigation and maybe the addition of new rules or a combination of them to deal with it. This is why this subject was left for future enhancements. Moreover, since the indexer was built using Java, the JDBC to ODBC Bridge was used to perform the connection with the SQL Server DB. This driver is known to be slow and thus affects the performance. Furthermore, the static lists such as the "StopList" and "Nonderivables" can be further populated in such a ways that will enhance the performance and accuracy of the system.
References


أشوا في قواعد اللغة العربية. أحمد فارس دار الأمان بيروت. 1995

المنجد في اللغة والإعلام. دار المشرق. 1998
(Al Mounjed Fi Allougha Waleclam. Dar Al Mashrek. 1998)

أوئية الفعل في ثقافة ابن الحبيب. دراسات لسانية ولغوية. د. عصام فور الدين. دار الفكر العربي بيروت. 1997

الكافي في اللغة الفارسية وادابها. الدكتور عبد الله الخادم. 1997
(Al Kafi Fi Al Lougha Al Farisiyya Wa Adabouha. Dr. Abdallah Al Khalidi. 1997)

منحة اللغة الفارسية. الجزء الأول. موجز في قواعد اللغة. الدكتور عبد الله الخادم. 1981
(Mouftah Al Lougha Al Farisiyya. Part 1. Mouijaz Fi Kawaced Al Lougha. Dr. Abdallah Al Khalidy)

القربيون لغة وتاريخ. الدكتورة حنوة الخضير. 1995
(Al cibriyyon Lougha Wa Tarikh. Dr. Haniya Al Khatib. 1995)
Appendix A
Definitions for the Rules Compiler
Conditions and Actions

The general form of a rule compiler is:

<Priority> IF <Condition> <Value> [OR <Value> OR <Value> ... ] [AND <Condition>
<Value> ... ] THEN <Action> [<Values List>] [AND {<Action>} [<Values List>] AND ] ...

<Conditions> section:

1. TERM_IS_EQUAL_TO: this condition usually signals a term. What is to be done with this term depends on what action exits in the <action> section.
2. STARTS_WITH: This condition indicates the presence of a prefix.
3. ENDS_WITH: This condition indicates the presence of a suffix.
4. RHYMES_WITH: This condition indicates the presence of a pattern that the terms parsed in documents should be rhymed against. The <value> argument for this condition is usually a pattern that should be added to the Rhymes list.
5. STEM_ENDS_WITH: This condition is used to control the removal of suffixes. It is mainly used with the REPLACE_WITH action.
6. STEM_IS_EQUAL_TO: This condition is used to check the value of a certain stem after the lemmatization process takes place and then deals with it according to what the <action> section denotes.
<Actions> Section

1. **DISCARD**: This action tells the indexer to discard the term found in the <Value> argument of the condition section. This mainly causes the argument to be added to the StopList list.

2. **NEXT_WORD_IS_A_NOUN**: This action indicates that the next term is an improper noun.

3. **NEXT_WORD_IS_A_VERB**: This action indicates that the next term is a verb.

4. **REMOVE_LETTERS**: This action will cause the suffixes and prefixes found in the <Value> argument to be removed from the words they encountered in.

5. **SAVE_AS_IS**: This action is used mainly with proper nouns, names, and foreign words. The term is added to the NonDerivable list.

6. **RHYME_IS_EQUAL_TO**: This action indicates the presence of a pattern which should be added to the Rhymes table.

7. **TERM_IS_A_NOUN**: This action indicates that the term is a noun. It is usually used with prefixes and suffixes.

8. **TERM_IS_A_VERB**: This action indicates that the term is a verb. It is usually used with prefixes and suffixes.

9. **EXTRACT_STEM**: This action instructs the indexer to extract the stem of the term found in the <Value> argument of the condition section.

10. **REPLACE_WITH**: This action instructs the indexer to replace the characters of the term in the <Value> argument with whatever values are provided in its <Value> argument.
Examples:

- \text{SL2 IF TERM IS_EQUAL_TO 'با' THEN DISCARD AND NEXT WORD IS A NOUN}
- \text{PF3 IF STARTS WITH أل OR قل THEN TERM IS A NOUN AND REMOVE LETTERS}
- \text{PF1 IF STARTS WITH و THEN REMOVE LETTERS}
- \text{SF1 IF ENDS WITH ت THEN TERM IS A VERB AND REMOVE LETTERS}
- \text{SF1 IF ENDS WITH ك THEN TERM IS A NOUN}
- \text{SL2 IF TERM IS_EQUAL_TO 'قد' THEN DISCARD AND NEXT WORD IS A VERB}
- \text{ND1 IF TERM IS_EQUAL_TO 'تتثبت' THEN SAVE AS IS}
- \text{ND1 IF TERM IS_EQUAL_TO 'لدن' THEN SAVE AS IS}
- \text{ND1 IF TERM IS_EQUAL_TO 'سرير' THEN SAVE AS IS}
- \text{RH6 IF RHYMES WITH 'يتعلن' THEN EXTRACT STEM AND TERM IS A VERB}
- \text{RH6 IF RHYMES WITH 'قطعون' THEN EXTRACT STEM AND TERM IS A VERB}
- \text{RH5 IF RHYMES WITH 'متعلن' THEN EXTRACT STEM AND TERM IS A NOUN}
- \text{RH1 IF TERM IS_EQUAL_TO 'لاضطراب' THEN RHYME IS_EQUAL_TO 'فت ول'}
- \text{RH1 IF TERM IS_EQUAL_TO 'يتطاوع' THEN RHYME IS_EQUAL_TO 'فت ول'}
Appendix B
Transliteration Conventions

The transliteration used here is adapted from Roman (1990). It includes no special character for the sake of portability. 'Emphatic' (pharyngealized) consonants as well as the voiceless pharyngeal h are in boldface. Underlining is used to distinguish constrictive consonants from their occlusive counterpart, or from a 'neighboring' phoneme (this not a phonetic transcription system!). 'Long' vowels bear a circumflex accent. Here is a short presentation:

- Short vowels: a, u, i.
- Long vowels: 'alif = ä ; wāw = ū ; yā = ĭ.
- Consonants (in Alphabetic order)

<table>
<thead>
<tr>
<th>Table B.1: Transliteration Conventions</th>
</tr>
</thead>
<tbody>
<tr>
<td>hamza = '</td>
</tr>
<tr>
<td>hā' = h</td>
</tr>
<tr>
<td>zāy = z</td>
</tr>
<tr>
<td>tā'= t</td>
</tr>
<tr>
<td>qā'= q</td>
</tr>
<tr>
<td>hā' = h</td>
</tr>
</tbody>
</table>

- Morphograms: tā'marbūta = +a&.

---

17 The above transliterations have been taken exactly as they were mentioned in Disky's paper [11].
The indexer resides inside the server component, which is in turn running on a separate machine other than the machines running the parsers. This client/server architecture relieves the server, and thus the indexer, from the burden of running the different parsers and paves the way to easily integrate any new file format parser. A Parser class has been created for this purpose. This class handles all the communications taking place with the server. The user instantiates an object of that class and uses it to communicate with the server. He/she then only has to add to this new parser the code that opens the targeted document and parses it for useful content. Figure C.1 shows the UML representation of that class. This class contains the following attributes:
• **Socket**: This is the socket object which the parser object will use to communicate with the server. Each parser object will have a separate socket.

• **is**: The input stream reader attribute. The parser object will use this object to receive any data from the server.

• **out**: The output stream writer attribute. This attribute is used to send anything to the server.

• **IP**: This attribute represents the IP address of the server.

• **Port**: This is the specific port that the instantiated parser object will use to communicate with the server. Each parser object will have its own port.

• **Args**: This attribute is mainly the full path or url of the document being indexed.

The already existing methods will minimize the code that the user will have to provide. All what he/she has to do is to only add the code to open the document being indexed and read its contents. He/she then uses the already existing methods to communicate with the server without worrying about how to initiate the connection with the server or where the server resides or to which port it has to write. By calling the function `init()` the socket attribute will be ready and the communication with the server will be established. The user then uses the `readline()` method to read anything from the server and the `send()` to send anything to the server. Finally, the user calls the `finalize()` method to terminate the connection with the server.

Proposing the use of such a class will help the developer of any parser a lot since it will minimize significantly the amount of code that he/she has to write. It will relieve him/her from
the burden of worrying about where the server resides or to which port to write. Finally when the parser has done parsing the user terminates the connection with the server and thus frees a resource and making it available to other parsers.
Appendix D
Rules List

In what follows is the list of all the rules used by our prototype in order to prove the validity of the proposed solutions.

IF TERM_IS_EQUAL_TO آنذاك THEN DISCARD
IF TERM_IS_EQUAL_TO آن THEN DISCARD
IF TERM_IS_EQUAL_TO أبل THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أخرى THEN DISCARD
IF TERM_IS_EQUAL_TO أحيان THEN DISCARD
IF TERM_IS_EQUAL_TO أحوالا THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أشد THEN DISCARD
IF TERM_IS_EQUAL_TO بصب THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أث المن THEN DISCARD
IF TERM_IS_EQUAL_TO أكبر THEN DISCARD
IF TERM_IS_EQUAL_TO أكون THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أو THEN DISCARD
IF TERM_IS_EQUAL_TO أسا THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أسا THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أما THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أمكن THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أن THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO إلا THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أن THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنتم THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنتم THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنتم THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنتم THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنتم THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنتم THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أنك THEN DISCARD
IF TERM_IS_EQUAL_TO أني THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM_IS_EQUAL_TO أور THEN DISCARD
IF TERM_IS_EQUAL_TO أو الشك THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أي THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أيا THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أيان THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أيها THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أيضا THEN DISCARD
IF TERM_IS_EQUAL_TO أي THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO أيها THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO إذ THEN DISCARD AND NEXT_WORD_IS_A_VERB
IF TERM IS_EQUAL TO إذا THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL TO إن THEN DISCARD
IF TERM IS_EQUAL TO إلا THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL TO إن THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL TO إلى THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO ليك THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO إن THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO الدليل THEN DISCARD
IF TERM IS_EQUAL_TO الأحياة THEN DISCARD
IF TERM IS_EQUAL_TO الأصغر THEN DISCARD
IF TERM IS_EQUAL_TO الأكبر THEN DISCARD
IF TERM IS_EQUAL_TO لتلي THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO الذي THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO الذين THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO لكلة THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO اللتان THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO اللواتي THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO إلى THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO إن THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO بل THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO بلس THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO يتم THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO بالإمكان THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO يقال THEN DISCARD
IF TERM IS_EQUAL_TO ين THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO يد THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO يدوى THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO السرعة THEN DISCARD
IF TERM IS_EQUAL_TO بينما THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO بدعة THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO بعد THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO بعدما THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO بعدما THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO بعدما THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO بعدما THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO فيلم THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO يليك THEN DISCARD
IF TERM IS_EQUAL_TO يكل THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO يكما THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO يدن THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO يكي THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO يبل THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO يبال THEN DISCARD AND NEXT_WORD IS_A_NOUN
IF TERM IS_EQUAL_TO يبلغ THEN DISCARD AND NEXT_WORD IS_A_VERB
IF TERM IS_EQUAL_TO يلم ثم THEN DISCARD AND NEXT_WORD IS_A_VERB
IF $\text{TERM IS_EQUAL_TO} \text{ إذا} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ إن} \text{ THEN DISCARD}$
IF $\text{TERM IS_EQUAL_TO} \text{ فإن} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ إن الزمان} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ إلى} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ لذلك} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ بل} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ في} \text{ THEN DISCARD}$
IF $\text{TERM IS_EQUAL_TO} \text{ الثامن} \text{ THEN DISCARD}$
IF $\text{TERM IS_EQUAL_TO} \text{ الثامن} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ الذي} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ الذي} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ الذي} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ الذي} \text{ THEN DISCARD AND NEXT WORD IS A VERB}$
IF $\text{TERM IS_EQUAL_TO} \text{ إلى} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ إلى} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ على} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$
IF $\text{TERM IS_EQUAL_TO} \text{ يكمن} \text{ THEN DISCARD AND NEXT WORD IS A NOUN}$

80
IF TERM IS_EQUAL_TO ِبِلِيّ THEN DISCARD
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِنّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِهّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِهِّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِهِّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِهِّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِهِّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A VERB
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
IF TERM IS_EQUAL_TO ِبِيّ THEN DISCARD AND NEXT_WORD IS A NOUN
81
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM_IS_EQUAL_TO THEN DISCARD
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A VERB
IF TERM IS EQUAL TO THEN DISCARD
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN
IF TERM IS EQUAL TO THEN DISCARD AND NEXT WORD IS A NOUN

IF TERM_IS_EQUAL_TO index THEN DISCARD
IF TERM_IS_EQUAL_TO ناين THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO كون THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO كن THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO كنون THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO يمكن THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO يمكننا THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO يمكنكم THEN DISCARD AND NEXT_WORD_IS_A_NOUN
IF TERM_IS_EQUAL_TO يمكنكم THEN DISCARD AND NEXT_WORD_IS_A_NOUN

IF TERM_IS_EQUAL_TO الدعم THEN SAVE AS IS
IF TERM_IS_EQUAL_TO الجذور THEN SAVE AS IS
IF TERM_IS_EQUAL_TO الحادي THEN SAVE AS IS
IF TERM_IS_EQUAL_TO الحرير THEN SAVE AS IS
IF TERM_IS_EQUAL_TO الرفوف THEN SAVE AS IS
IF TERM_IS_EQUAL_TO القفص THEN SAVE AS IS
IF TERM_IS_EQUAL_TO القفص THEN SAVE AS IS
IF TERM_IS_EQUAL_TO الله THEN SAVE AS IS
IF TERM_IS_EQUAL_TO المتحدة THEN SAVE AS IS
IF TERM_IS_EQUAL_TO الولايات THEN SAVE AS IS
IF TERM_IS_EQUAL_TO يار THEN SAVE AS IS
IF TERM_IS_EQUAL_TO بشر THEN SAVE AS IS
IF TERM_IS_EQUAL_TO بوست THEN SAVE AS IS
IF TERM_IS_EQUAL_TO تفسار THEN SAVE AS IS
IF TERM_IS_EQUAL_TO تفسير THEN SAVE AS IS
IF TERM_IS_EQUAL_TO تكلزم THEN SAVE AS IS
IF TERM_IS_EQUAL_TO توج THEN SAVE AS IS
IF TERM_IS_EQUAL_TO جوربان THEN SAVE AS IS
IF TERM_IS_EQUAL_TO خال THEN SAVE AS IS
IF TERM_IS_EQUAL_TO دمشق THEN SAVE AS IS
IF TERM_IS_EQUAL_TO دودج THEN SAVE AS IS
IF TERM_IS_EQUAL_TO رافيق THEN SAVE AS IS
IF TERM_IS_EQUAL_TO روبرت THEN SAVE AS IS
IF TERM_IS_EQUAL_TO سعود THEN SAVE AS IS
IF TERM_IS_EQUAL_TO سمير THEN SAVE AS IS
IF TERM_IS_EQUAL_TO سوريا THEN SAVE AS IS
IF TERM_IS_EQUAL_TO سمير THEN SAVE AS IS
IF TERM_IS_EQUAL_TO كيمال THEN SAVE AS IS
IF TERM_IS_EQUAL_TO غدا THEN SAVE AS IS
IF TERM_IS_EQUAL_TO فرجانا THEN SAVE AS IS
IF TERM_IS_EQUAL_TO فلاشير THEN SAVE AS IS
IF TERM_IS_EQUAL_TO فورد THEN SAVE AS IS
IF TERM_IS_EQUAL_TO فورورد THEN SAVE AS IS
IF TERM_IS_EQUAL_TO فورورد THEN SAVE AS IS
IF TERM_IS_EQUAL_TO كون THEN SAVE AS IS
IF TERM_IS_EQUAL_TO كرون THEN SAVE AS IS
IF TERM_IS_EQUAL_TO لبنان THEN SAVE AS IS
IF TERM_IS_EQUAL_TO لندن THEN SAVE AS IS
IF TERM_IS_EQUAL_TO واشنطن THEN SAVE AS IS

IF STARTS_WITH ل THEN REMOVE_LETTERS AND TERM IS A VERB
IF STARTS_WITH ل THEN REMOVE_LETTERS AND TERM IS A_VERB
IF STARTS_WITH ل THEN REMOVE_LETTERS AND TERM IS A_NOUN
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb" 
إذا كان النهاية من الكلمة "term is a noun" 
إذا كان النهاية من الكلمة "term is a verb"
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A VERB
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A VERB
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN
IF RHYMES_WITH THEN EXTRACT_STEM AND TERM IS A NOUN

IF TERM IS EQUAL TO THEN RHYME IS EQUAL TO
IF TERM IS EQUAL TO THEN RHYME IS EQUAL TO