Republic of Yemen

University of Science and Technology

Faculty of IT and Computer

رؤيـة

Arabic Search Engine

Students' Names:

Khawlah Mohammed Shaiban

Feryal Abdullah Al-khalaqy

Somia Homadi Al-Khadhami

Supervisor's Name:

Dr. Asma'a Al-sharjabi

This project has been built to complete the graduation requirements of the Department of Computer Science and Information Technology for the year - 2018

قال تعالى:

﴿ وَقُلْ رَّبِ زِدْنِي عِلْمَا ﴾

[سورة طه]

صدق الله العظيم

DECLARATION

We hereby declare that all information contained in this document is result of our own work except cited and referenced materials. And have been presented in accordance with academic and ethical rules.

Names:

Khawlah Mohammed Shaiban Signature:……

Feryal Abdullah Al-khalaqy Signature:……

Somia Homadi Al-Khadhami Signature:……

Supervisor:

Dr. Asma'a Al-sharjabi Signature:……

ACKNOWLEDGMENT

First, all praise to Allah, the most Gracious and most Merciful, who gave us the ability and patience throughout our studies for completing the degree program.

Completion of this project would not be possible without the effort of our advisor Dr. Asma'a Alshargabi, words cannot describe our gratitude for her guidance and support with this project.

We are grateful to all of those with whom we have had the pleasure to work during this project. Finally, nobody has been more important to us in the pursuit of this project than the member of our families. We would like to thank our parents and siblings, whose love and guidance are with us in whatever we pursue. They are the ultimate role models. Most importantly, we wish to thank our doctors, teachers and University.

DEDICATION

إلى أبي الغالي من كلله الله بالهيبة والوقار ، إلى قدوتي من علمني العطاء والإصرار و أحمل اسمه بكل افتخار. إلى الحبيبة أمي ، من جنتي تحت قدميها ، ملاكي في الحياة ، إلى بسمة الحياة من كان دعائها سر نجاحي ، من دعمتني و شجعتني ووثقت بقدراتي. إلى أخواتي شموعي المتقدة التي تنير ظلمة حياتي ، و تطلعنّ لنجاحي بنظرات الأمل. إلى أخوتي سندي ورفقاء دربي في هذه الحياة ، في نهاية مشروعي أريد أن أشكركم على مواقفكم النبيلة. إلى أستاذاتي و أساتذتي الأفاضل من أخذت منهم العلم والحكمة ، وأخيراً إلى كل من دعمني و شجعني وكان مصدراً لإلهامي ونجاحي ، وإلى الوطن العربي.

ABSTRACT

Information retrieval refers to the retrieval of textual documents such as Web documents. Due to wide research in the Information Retrieval field, there are many retrieval techniques that have been developed for Arabic language. Because Arabic language is different and has a difficult structure than other languages. The main objectives of this research is enhancing Arabic information retrieval by Support the search using the basic morphological forms for words, definitely stem, lemma. Also, enhance ranking and recognize Name of Entity.

To achieve these objectives, we create a website that implements preprocessing stage that is necessary for information retrieval. One of these steps is normalizing, by remove all the diacritics and return the word for stem or lemma. The second preprocessing step is removing the stop words.

We improved the searching process by searching using the basic morphology in Solr search server and Farasa linguistic tool with the website that we created to achieve the optimal results.

The proposed toolkits were Solr Search server that uses light stemming for Arabic Language, Farasa linguistic tool that used to extract lemma of the word and the programming languages were Java language, PHP language, Html Language and Java Script Language. Also, we used PHP/Java bridge and Solarium PHP/Solr library for the integration.

At result for that, the website that we created showing the Arabic results quickly using the stem of the words and the user can use another way for searching using lemma or Name of entity.

Keywords: *Natural Language Processing, Information Retrieval, Search Engine, light Stemming, lemma, Arabic Language, Arabic Search, Morphology.*

Table of Contents

QURANIC VERSE I

DECLARATION II

ACKNOWLEDGMENT III

DEDICATION IV

ABSTRACT V

LIST OF FIGURES XI

LIST OF TABLES V

ℜ GLOSSARY & ABBREVIATION VI

CHAPTER ONE 1

INTRODUCTION 1

1. 1 INTRODUCTION 2

1. 2 PROBLEM STATEMENT 2

1. 3 PROJECT OBJECTIVES 2

1. 4 PROJECT SCOPE 3

1. 5 OUT SCOPE 3

1. 6 PROJECT METHODOLOGY 3

1. 7 TOOLS 4

1. 8 PROJECT PLAN 4

CHAPTER TWO 6

LITERATURE REVIEW 6

PART ONE 7

THEORETICAL BACKGROUND 7

2. 1 NATURAL LANGUAGE 8

2. 2 NATURAL LANGUAGE PROCESSING 8

2.2.1 MOTIVATIONS OF NATURAL LANGUAGE PROCESSING 9

2.2.2 NATURAL LANGUAGE PROCESSING ANALYZING TECHNIQUES 10

2.2.2.1 RULE BASED APPROACH 10

- TREEBANKS 10

2.2.2.2 STATISTICAL BASED APPROACH 11

2.2.3 NATURAL LANGUAGE PROCESSING TASKS 11

2.2.4 NATURAL LANGUAGE PROCESSING LEVELS 16

- PHONETIC OR PHONOLOGICAL LEVEL OF ANALYSIS: 16
- LEXICAL LEVEL OF ANALYSIS: 16
- MORPHOLOGICAL LEVEL OF ANALYSIS: 17
- SYNTACTIC LEVEL OF ANALYSIS: 17

- **SEMANTIC LEVEL OF ANALYSIS:** **18**
- **THE DISCOURSE LEVEL OF ANALYSIS:** **18**
- **THE PRAGMATIC LEVEL OF ANALYSIS:** **18**

2.2.5 CHALLENGES OF NATURAL LANGUAGE PROCESSING **19**

- **TYPES OF AMBIGUITY** 19
  - ***Syntactic Ambiguity*** *19*
  - ***Semantic Ambiguity*** *19*
  - ***Referential Ambiguity*** *19*
  - ***Local Ambiguity*** *19*

2.2.6 NATURAL LANGUAGE PROCESSING APPLICATIONS **19**

2.2.6.1 SPELL AND GRAMMAR CHECKING 19
2.2.6.2 WORD PREDICTION 20
2.2.6.3 INFORMATION RETRIEVAL 20
2.2.6.4 TEXT CATEGORIZATION 21
2.2.6.5 SUMMARIZATION 21
2.2.6.6 QUESTION ANSWERING 22
2.2.6.7 INFORMATION EXTRACTION 22
2.2.6.8 MACHINE TRANSLATION 23
2.2.6.9 SENTIMENT ANALYSIS 23
2.2.6.10 OPTICAL CHARACTER RECOGNITION 23
2.2.6.11 SPEECH RECOGNITION 24
2.2.6.12 SPEECH SYNTHESIS 24
2.2.6.13 SPOKEN DIALOG SYSTEMS 25

2.3 THE ARABIC LANGUAGE **25**

2.3.1 ARABIC'S GROWTH ON THE INTERNET 26
2.3.2 ARABIC'S POPULARITY AS A SECOND-FASTEST GROWING LANGUAGE 26
2.3.3 ARABIC LANGUAGE FEATURES 26
2.3.4 ARABIC ORTHOGRAPHY 27
2.3.5 ARABIC MORPHOLOGY 27
2.3.6 SIGNIFICANCE OF ANLP FOR THE ARABIC-SPEAKING POPULATION 28

2.4 ARABIC NATURAL LANGUAGE PROCESSING(ANLP) **28**

2.4.1 ARABIC NATURAL LANGUAGE PROCESSING TECHNIQUES 29
2.4.1.1 Rule-Based Approach *29*

- **ARABIC TREEBANKS** 31

2.4.1.2 Statistical-Based Approach *32*
2.4.1.3 Hybrid-Based Approach *34*

2.4.2 ARABIC NATURAL LANGUAGE PROCESSING TASKS 35
2.4.2.1 TOKENIZATION 35

2.4.2.2 PART-OF-SPEECH TAGGING 36
2.4.2.3 BASE PHRASE CHUNKER 36
2.4.2.4 PARSING 36
2.4.2.5 PROPER NAME TRANSLITERATION 36
2.4.2.6 SPELLING CORRECTION 37
2.4.2.7 SPEECH RECOGNITION AND SYNTHESIS 37
2.4.2.8 DETOKENIZATION 38
2.4.3 ARABIC NATURAL LANGUAGE LEVELS 38
A) PHONETIC OR PHONOLOGICAL LEVEL 38
B) LEXICAL LEVEL OF ANALYSIS 39
C) MORPHOLOGICAL LEVEL OF ANALYSIS 39
D) SYNTACTIC LEVEL OF ANALYSIS 40
E) SEMANTIC LEVEL OF ANALYSIS 40
F) DISCOURSE LEVEL OF ANALYSIS 41
G) PRAGMATIC LEVEL OF ANALYSIS 41
2.4.4 WHY ARABIC IS A CHALLENGE TO COMPUTATIONAL LINGUISTS 41
2.4.4.1 Arabic Diglossia 41
2.4.4.2 The Arabic Script 42
2.4.4.3 Diacritics 42
2.4.4.4 Ambiguity of ANLP 43
2.4.5 APPLICATIONS AND TOOLS OF ANLP 44
2.5 SEARCH ENGINES 46
2.5.1 DEFINITION 46
2.5.2 COMPONENTS OF SEARCH ENGINE 46
2.5.3 ARCHITECTURE OF SEARCH ENGINE 47
2.5.4 HOW SEARCH ENGINES WORK 47
2.5.5 FEATURES OF SEARCH ENGINE 48
2.5.6 SEARCH ENGINE INDEXING PROCESS 48
2.5.7 QUERY PROCESSING IN SEARCH ENGINES 48
2.5.8 SEARCH ENGINES THAT SUPPORT ARABIC 48
2.5.9 ARABIC SEARCH ENGINES 51
PART TWO 53
RELATED WORKS 53
2. 1 FARASA 54
2.1.1 DEFINITION 54
2.1.2 PROCEDURE 54
2.1.3 TRAINING AND TESTING 55
2.1.4 WHY USE FARASA 56

2.2 APACHE SOLR SEARCH SERVER 56

2.2.1 DEFINITION 56

2.2.2 BRIEF HISTORY 56

2.2.3 FEATURES OF APACHE SOLR 57

2.2.4 LUCENE IN SEARCH APPLICATIONS 58

2.2.5 SOLR ARCHITECTURE - BUILDING BLOCKS 58

2.2.6 CONFIGURATION FILES 59

2.3 THE WIKIPEDIA 60

2.3.1 DEFINITION 60

2.3.2 BRIEF HISTORY 60

2.3.3 WHY USING WIKIPEDIA 60

2.4 SEARCH ENGINE INDEXING 60

2.4.1 INTRODUCTION 60

2.4.2 INDEXING PURPOSE 61

2.4.3 INDEX DESIGN FACTORS 61

2.4.4 CHALLENGES IN PARALLELISM 62

2.4.5 INDEX DATA STRUCTURES 62

2.4.6 INVERTED INDEX 62

2.4.7 APPLICATIONS OF INVERTED INDEX 63

CHAPTER THREE 64

ANALYSIS 64

3.1 FEASIBILITY STUDY 65

3.1.1 TECHNICAL FEASIBILITY 65

3.1.2 LEGAL FEASIBILITY 65

3.2 DATA GATHERING 66

3.2.1 The Internet 66

3.2.2 The Previous Documents 66

3.2.3 The Questionnaire 66

3.3 THE REQUIREMENT SPECIFICATION 66

3.3.1 USER REQUIREMENTS 66

3.3.1.1 Functional Requirements 66

3.3.1.2 Non-Functional Requirements 67

3.3.2 SYSTEM REQUIREMENTS 68

3.3.2.1 Functional Requirements 68

3.3.2.2 Non-Functional Requirements 70

3.4 MODELING 72

3.4.1 UML USE CASE DIAGRAM 72

3.4.2 USE CASE DESCRIPTION 73
3.4.3 CLASS DIAGRAM (DOMAIN MODEL) 80
3.4.4 ACTIVITY DIAGRAM 81
CHAPTER FOUR 82
DESIGN 82
4. 1 ARCHITECTURE DESIGN 83
4. 2 CLASS DIAGRAM (DESIGN MODEL) 84
4. 3 SEQUENCE DIAGRAMS 85
4. 4 COMMUNICATION DIAGRAMS 89
CHAPTER FIVE 92
IMPLEMENTATION 92
5. 1 THE SITE MAP 93
5. 2 USER INTERFACE IMPLEMENTATION 93
5. 3 DEPLOYMENT DIAGRAM 97
5. 4 THE RESULTS 98
CHAPTER SIX 99
CONCLUSION & RECOMMENDATIONS 99
6. 1 CONCLUSION 100
6. 2 FUTURE WORK 100
REFERENCES 101
APPENDIX A (THE QUESTIONNAIRE SAMPLE) 109
THE QUESTIONNAIRE SAMPLE 109
A) THE QUESTIONNAIRE TITLE: 109
B) BRIEF DESCRIPTION: 109
C) QUESTIONNAIRE'S QUESTIONS: 109
D) QUESTIONNAIRE'S RESULTS 111
APPENDIX B (COMPONENT IMPLEMENTATION) 118
COMPONENT IMPLEMENTATION 118
1) FARASA PACKAGES 118
I) FARASA Segmenter *118*
II) FARASA Name of Entity Package *124*
2) SOLR SEARCH SERVER 125
II) managed-schema *126*
3) PHP/JAVA BRIDGE 129
4) SOLARIUM PHP/SOLR LIBRARY 130

List of Figures

Figure 1.1: Project Plan Part 1 4

Figure 1.2: Project Plan Part 2 5

Figure 2.1: Roadmap for NLP research in IS 12

Figure 2.2: General Organization for NLP Tasks 13

Figure 2.3: Detailed Organization for NLP Tasks 15

Figure 2.4: Google Spell & Grammar checking 20

Figure 2.5: Google Word Prediction 20

Figure 2.6: Google Information Retrieval 20

Figure 2.7: uClassify Text Categorization..... 21

Figure 2.8: SMMRY Summarization application 21

Figure 2.9: START Question Answering application..... 22

Figure 2.10: WIKIPEDIA Information Extraction application..... 22

Figure 2.11: Information Extraction application..... 23

Figure 2.12: Iancet Information Extraction application..... 23

Figure 2.13: Sentiment analysis application..... 23

Figure 2.14: Mobile OCR Optical Character Recognition application..... 24

Figure 2.15: Siri Speech Recognition application 24

Figure 2.16: Speech Synthesis application 24

Figure 2.17: Siri & IBM Watson Developer Cloud spoken dialog systems 25

Figure 2.18: Search Engine Architecture 47

Figure 2.19: Google Search Engine Architecture..... 50

Figure 2.20: Solr Architecture – Building Blocks 58

Figure 3.1: Use Case Diagram 72

Figure 3.2: Class Diagram (Domain Model) 80

Figure 3.3: Activity Diagram ……. **81**

Figure 4.1: Architecture Diagram ……..…… **83**

Figure 4.2: Class Diagram (Design Model) ……..…… **84**

Figure 4.3: Sequence Diagram for overall search process …..……..…… **85**

Figure 4.4: Input Query Sequence Diagram …..……...…… **85**

Figure 4.5: Process Query Sequence Diagram …..……...…… **86**

Figure 4.6: Search Results Sequence Diagram …..……...…… **86**

Figure 4.7: Select Results Sequence Diagram …..…..……...…… **87**

Figure 4.8: Show Results Sequence Diagram …..……...…….. **87**

Figure 4.9: Customize SE Sequence Diagram …..……...…… **88**

Figure 4.10: Input Query Communication Diagram ……...…….…….. **89**

Figure 4.11: Process Query Communication Diagram ……...…….…….. **89**

Figure 4.12: Search Results Query Communication Diagram ……...…….…….. **90**

Figure 4.13: Select Results Communication Diagram …..……...……...…….…….. **90**

Figure 4.14: Show Results Communication Diagram ……..……...…….…….. **90**

Figure 4.15: Customize SE Communication Diagram ……...…….…….. **91**

Figure 5.1: The Website Site Map ……..……..…..…….…….**93**

Figure 5.2: The Website Home Page …..……..…..…….…….**94**

Figure 5.3: The Website Menu ……..……...…..……...….**94**

Figure 5.4: The Website Setting - The setting submenu ……..…...……**95**

Figure 5.5: The Website Setting - Change font color option ……..…..……**95**

Figure 5.6: The Website Setting - Change theme color option ……..…..…..……**96**

Figure 5.7: The Website Setting - Change background color option ……...…..……...**96**

Figure 5.8: The Website Result Page ..……...……..…..……...**97**

Figure 5.9: The Deployment Diagram ……...……..…..……...**97**

Figure 5.10: The Website Result Page for explain the results……..…..…..……..…..…..…98

Figure A-1: Questionnaire's result of Age ……....……111

Figure A-2: Questionnaire's result of Gender ...……....……111

Figure A-3: Questionnaire's result of Dexterity of Arabic ...……112

Figure A-4: Questionnaire's result of Dexterity of English ..……112

Figure A-5: Questionnaire's result of Preferred Search Language……113

Figure A-6: Questionnaire's result of SEs that support Arabic ....……113

Figure A-7: Questionnaire's result of Motivations for Search in Arabic ...……...114

Figure A-8: Questionnaire's result of obstruction for search in Arabic ……114

Figure A-9: Questionnaire's result of obstruction for search in English …..……..114

Figure A-10: Questionnaire's result of main purpose of search ……..……..……..…… 115

Figure A-11: Questionnaire's result of the most used Search Engine ..……115

Figure A-12: Questionnaire's result of time wasted in search using Arabic Language ……116

Figure A-13: Questionnaire's result of Adding features to SE ……..……...……116

List of Tables

Table 2.1: Comparison of Error rate between Farasa, Madamira & Qatara **56**

Table 3.1: Use Case Description table order **83**

Table 3.2: Input Query Use Case Description **84**

Table 3.3: Process Query Use Case Description **85**

Table 3.4: Search Results Use Case Description **86**

Table 3.5: Show Results Use Case Description **87**

Table 3.6: Select Results Use Case Description **88**

Table 3.7: Customize the Search Engine Use Case Description **89**

ℜ Glossary & Abbreviation

1) **Search Engine:** A type of software program or script available through the Internet that seeks out and indexes documents from the World Wide Web and USENET groups based on specific criteria and searches these documents and files using keywords and returns the results of any files containing those keywords EXAMPLES: Google, Excite, Lycos, AltaVista, Infoseek, Yahoo.
2) **Web site:** A group of related pages, images, and files on a Web server.
3) **Graph link:** An electronic pathway that may be displayed in the form of highlighted text, graphics or a button that connects one web page with another web page address. Think of a hyperlink as a request to visit another place. A simple click on the link will take you there.
4) **Browser (Web browser):** software program (either text-based or graphical) that allows a person to access and browse (surf) pages on the Internet in an easy-to-use way. EXAMPLES: Firefox, Safari, Internet Explorer, Chrome, Opera.
5) **Surf:** To look at or search for web pages, usually when you are browsing from one page to another quickly by following links.
6) **Web:** Abbreviation for World Wide Web.
7) **QCRI:** Abbreviation for Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar, QCRI is an advanced research center focused on modern ICT areas.
8) **Morphology:** the study of the internal structure of words and their semantic building blocks.
9) **Corpus:** is a collection of pieces of language text in electronic form, selected according to external criteria to represent, as far as possible, a language or language variety as a source of data for linguistic research.
10) **Farasa:** ("insight" in Arabic), a fast and accurate Arabic SVM-based segmenter that uses a variety of features and lexicons to rank possible segmentations of a word.
11) **The root:** is the basic form (an abstract "super-lemma") of word from which many derivations can be obtained by attaching certain affixes or suffixes so we produce many nouns and verbs and adjectives from the same root (Group all words that share a semantic field).
12) **Modern Standard Arabic (MSA):** is the language of modern writing, prepared speeches, and the language of the news.
13) **Homographic:** they have the same orthographic form, though the pronunciation is different.

14) **Stem**: is the least marked form of a word, that is the uninflected word without suffixes or prefixes and is the actual appearance of the root.
15) **Pattern:** is a template of vowels, or a combination of consonants and vowels.
16) **Morphology**: is the study of the internal structure of words and the systematic covariation between that structure and the meaning of the word.
17) **Diglossia:** "a situation in which two varieties of the same language live side by side, each performing a different function".
18) **Classical Arabic:** "the endpoint of a development within the complex of varieties of Old Arabic".
19) **Lexeme:** is a word in an abstract sense. "Lexemes can be thought of as families of words that differ only in their grammatical endings or grammatical forms". Lexemes are also-called lemmas.
20) **NL:** Natural language refers to any human written or spoken languages that has evolved and learned naturally for human communication at early childhood.
21) **IS:** Information Systems.
22) **Wiki:** is a type of website whose contents can be edited from the web browser, and which keeps a version history for each editable page.
23) **Index Size:** How much computer storage is required to support the index.
24) **Maintain Index:** How the index is maintained over time.
25) **Lookup speed:** How quickly a word can be found in the Inverted index.

CHAPTER ONE

INTRODUCTION

1. 1 INTRODUCTION

The web creates new challenges for information retrieval. The amount of information on the web is growing fast, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained or with search engines [1]. That are indexing almost a thousand times as much data and between them providing reliable responses to around a billion queries a day in a different of languages [2].

Search engines that depend on keyword matching usually return too many low quality matches, especially for Arabic language. Arabic language is different and has a difficult structure than other languages, that's because it is a very rich language with complex morphology. [3] The morphology of Arabic creates special challenges to computational natural language processing systems. Many SEs doesn't fully support Arabic morphology; there are many weakness and problems in returning too many low quality results. Starting from this point, we have built a morphological search engine using existing components (Farasa linguistic tool & Solr search engine) which addresses many of the problems of existing systems and search using complex query. Farasa and Solr are available in Internet and fully supports Arabic morphology to provide much higher quality search results.

1. 2 PROBLEM STATEMENT

Search engines (SEs) don't consider Arabic basic morphology even in famous Arabic sites (like Aljazeera.net) or famous search engine (like Google.com). They don't fully support Arabic characteristics, such as search by stem or lemma, root and derivatives, etc. For example, typing same sentence with same key word and different stop words such as (حروف الجر - prepositions) the result almost be different. Also there isn't an Arabic website that provides search services using stemming and lemmazation. There are some search engines that try to solve this problem, such as Google, Yamli, Eiktub, Yoolki, Bing and so on. But still have some weakness with get low quality results. These weaknesses are because of the complex morphology that Arabic Language has.

1. 3 PROJECT OBJECTIVES

This project aims to build a website that provides search service using Arabic language. The project objectives are as follows:

1) **Enhance Accuracy of Arabic Search:** Support the search using the basic morphological forms for words, definitely stem, lemma.
2) **Enhance ranking and recognize Name of Entity.**
3) **Provide a search service using complex queries.**

1. 4 PROJECT SCOPE

1. Develop and configure Farasa tools and Solr SS (Search Server) to identify the processes of search that should be provided.
2. Build a search engine that use Farasa packages and solr search server.
3. The search engine will use only Farasa corpuses (Aljazeera corpus, Wikipedia gazetteer, AraComLex and Buckwalter stems).

1. 5 OUT SCOPE

1. This project will provide the search in Arabic language only.
2. If the user writes the Arabic word with English characteristics, then the search engine will not expect what is that word.
3. The SE will not provide searching by dialect (accent).

1. 6 PROJECT METHODOLOGY

The methodology that will be used is Reuse-oriented Software Engineering with waterfall: these models are appropriate for this project because it will use existing components and should build in high quality to ensure the efficiency, reliability, and maintainability of the project.

Steps for project methodology are as the following:

1. Requirement specification: in this step, build a site to search in Arabic Wikipedia (similar to corpus.byu.edu, from Brigham Young University, which has texts in English, Spanish, and Portuguese).

2. Component Analysis: in this step, Call **QCRI's Farasa** tools to extract for each word: its root, stem, etc. And Use Solr search engine to index words and their morphological info.

3. System design with reuse: in this step, Build interface!

4. Development and Integration: in this step, Integrate Solr, Farasa and Search for complex queries (ex: What are the adjectives that follow a certain noun?) which are important for linguistic study and language learning.

5. System validation: in this step, **Test** the system.

1. 7 TOOLS

The tools that will be used are:

- ✓ Apache Solr 6.6 or 7.1 for index the files.
- ✓ Farasa Linguistic tool.
- ✓ Tomcat server 7.
- ✓ Rapid PHP 2015.
- ✓ Wikipedia dump for Arabic Articles.
- ✓ Solarium php/solr library.

1. 8 PROJECT PLAN

project activities that followed in this project are shown in figure (1.1, 1.2).

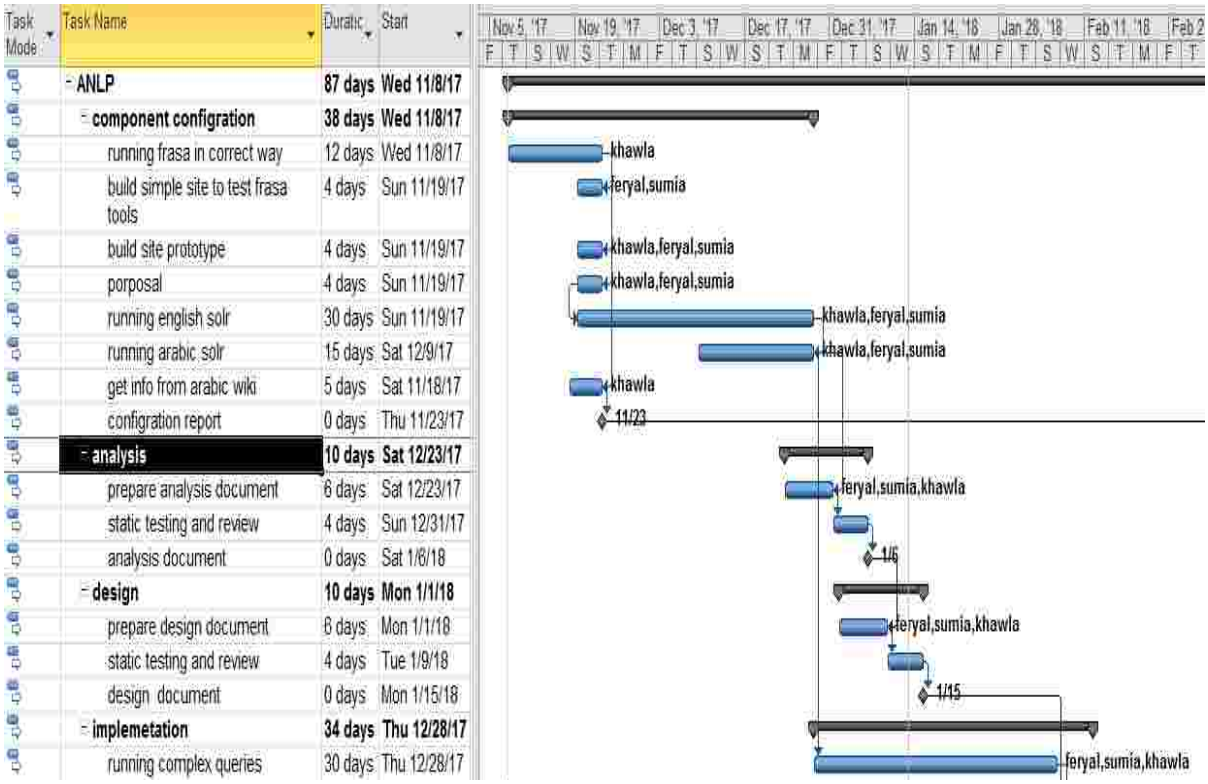

Figure 1.1: Project Plan Part 1

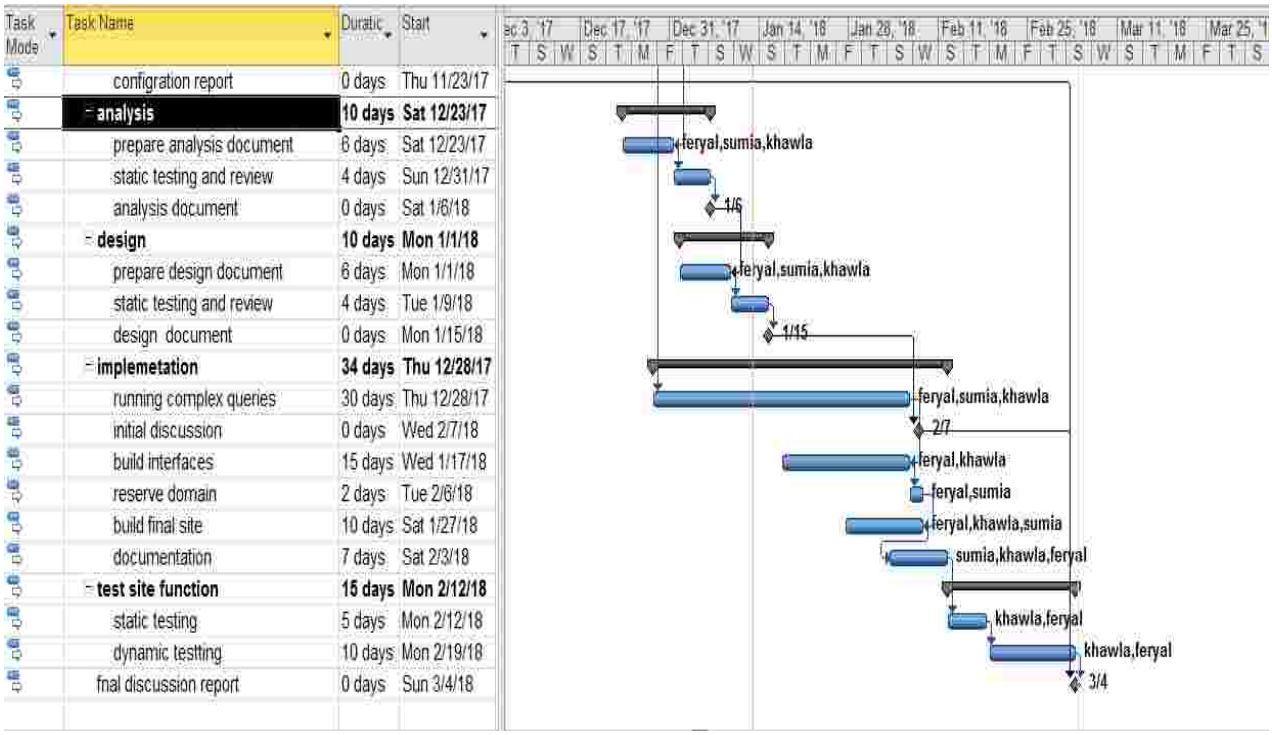

Figure 1.2: Project Plan Part 2

CHAPTER TWO

LITERATURE REVIEW

PART ONE

THEORETICAL BACKGROUND

2. 1 Natural Language

A natural language (NL) or ordinary language is any language that has evolved naturally in humans through use and repetition without aware planning or premeditation. Natural languages can take different forms, such as writing or speaking. They are distinguished from constructed and formal languages such as those used to program computers or to study logic. [4]

The interaction between computers and human contains two branches of activities: Natural language understanding and Natural language generation. Natural language understanding is the process of transferring natural language collected Natural language generation from human to a machine understandable format (computational representation). It attempts to understand the meaning behind written text produce data which encapsulates this meaning. [5] In contrast, Natural language generation focuses on computer systems that can produce understandable texts in human language (based on the rules of the target language and the task at hand [6]). They share similar theoretical foundations and are used together in many real world applications, but the internal process of these two areas are different. Both of them are concerned with computational models of language, which requires essential linguistic understanding of all aspects of language, like words and parsing, parts of speech and morphology, phrases and grammars, lexical and sentence semantics, etc. [7]

2. 2 Natural Language Processing

Natural Language Processing (NLP) - aka Computational Linguistics - is an interdisciplinary field of Computer Science (CS), Artificial Intelligence (AI), and It is related to the field of computer- human interaction (HCI) **[5]** that is a blanket term used to describe a machine's ability to ingest what is said to it, break it down, understand its meaning, determine appropriate action, and respond back in language the user will understand. The field of natural language processing is deep and diverse. It studies of mathematical and computational modeling of various aspects of language and the development of a wide range of systems. These includes the spoken language systems that integrate speech and natural language. Natural language processing has a role in computer science because many aspects of the field deal with linguistic features of computation. And it is useful in the tutoring systems, duplicate detection, computer supported instruction and database interface fields as it provides a pathway for increased interactivity and productivity **[6]** [7].

Early computational approaches to language research focused on automating the analysis of the linguistic structure of language and developing basic technologies such as machine translation, speech recognition, and speech synthesis. Today's researchers refine and make use of such tools and sub-system integrated with an application concerns the refinement and application of NLP

techniques to solve real-world problems by applying existing NLP-related algorithms and tasks, creating spoken dialogue systems and speech-to-speech translation engines, mining social media for information about health or finance, and identifying sentiment and emotion toward products and services [8]. The research output may include several design artifacts implementing one or more NLP task and/or algorithms. For example, Valencia-Garcia designed a system to translate surgeon's natural language into robot-executive commands by integrating speech recognition, information extraction, semantic annotation, and inference tasks.

2.2.1 Motivations of Natural Language Processing

Factors enabled NLP:

2.2.1.1 A huge increase in computing power.

2.2.1.2 The explosion ,availability and vastness of very large amounts of linguistic or textual (not annotated) data, through electronic communication systems, social media, and whole contents available on World Wide Web , where the Exabyte (1018 bytes) of text, doubling every year or two like Web pages, emails, IMs, SMSs, tweets, docs, PDFs, …etc. combined with the increasing necessity to extract meaning from them and need for quick access to specific and comprehensive information has driven the advancement and commercial adoption of NLP in recent years **[8]** [9].

2.2.1.3 The development of highly successful machine learning (ML) methods e.g. deep learning.

2.2.1.4 A much richer understanding of the structure of human language and its deployment in social contexts.

2.2.1.5 The society whose access to web information is obstructed simply by their inability to use the key-board and operating system [10].

2.2.1.6 Growing role of natural language in human-computer interaction, and need for natural language based I/O for new devices, especially in robotics [11].

2.2.1.7 For search engines NLP motivations which are [12]:

a) Intelligently responding to the query.

b) Predicting next word for auto completion.

c) Ability to do selling corrections.

d) Segmenting words that may be joined without space.

e) Ranking the search results.

f) For e-mail understanding contents of an e-mail through NLP and determine if it is spam or not.

Nowadays, Natural Language Processing (NLP) is widely integrated with web and mobile applications, enabling natural interactions between human and computers.

2.2.2 Natural Language Processing analyzing Techniques

In NLP area, researchers have proposed several techniques for analyzing the NL. These techniques are basically categorized into two leading approaches:

2.2.2.1 Rule Based Approach

Rule approach relies on hand-constructed rules which are used to structure linguistic, acquired from linguistic experts and encapsulates human knowledge to apply these rules to the NL text. There is the possibility to automatically learn these rules through the analysis of annotated corpora using machine learning methods. It is important to mention here that a corpus (plural" corpora ") describes a set of documents that have been annotated by NLP experts with the correct values that need to be learned.

There are some advantages to the rule-based approach. First, rules can be refined for accuracy by experts in order to detect spelling or grammar mistakes. Second, is easy to in-corporate domain knowledge into the linguistic knowledge acquired for one natural language processing system may be reused to build knowledge required for a similar task in another system. However, the cost of this approach is high, since it requires great effort from human experts to analyze the large volume of data from textual documents [13].

- **Treebanks**

  Collections of manually checked syntactic analyses of sentences, or treebanks, are an important resource for building statistical parsers and evaluating parsers, in general. Rich treebank annotations have also been used for a variety of applications such as tokenization, diacritization, part-of-speech (POS) tagging, morphological disambiguation, base phrase chunking, and semantic role labeling. Under time restrictions, the creation of a treebank faces a tradeoff between linguistic richness and treebank size. This is especially the case for morpho-syntactically complex languages such as Arabic or Czech. Linguistically rich representations provide many (all) linguistic features that may be useful for a variety of applications. This comes at the cost of slower annotation as a result of longer guidelines and more intense annotator training. As a result, the richer the annotation, the slower the annotation process and the smaller the size of the treebank. Consequently, there is less data to train tools [19].

2.2.2.2 Statistical Based Approach

Rule based approach has problem, indeed as was noticed by Edward Sapir "All grammars leak". This is because people are always bending the rules to meet their needs. That's why researchers have proposed another approach. Rather than dividing sentences into grammatical and parts, ask (What are the common patterns that occur in language?) The major approach which used to identify these patterns is counting things, also known as statistics, statistical models of language are successfully used for many natural language processing tasks.

Statistical-based approach requires large text corpora to develop statistical models in order to automatically learn the linguistic features identified in the textual data. This approach uses mathematical techniques, including stochastic, probabilistic and statistical methods to solve some of the problems that arise due to long sentences from the NL text.

These methods are data-driven and rely on quantitative methods to automatically dis-cover relations between words from the sentences. The use of statistical approach has proven to be advantageous for lower levels NLP tasks of text analysis process but the main disadvantage of this approach is that it requires a large amount of data in order to achieve statistically significant results [13].

NLP research has evolved from empirical-based (rule-based) approaches (i.e. based on the physical symbol system hypothesis) to statistic-based, where quantitative approaches, such as machine learning algorithms, are adopted to facilitate automated language processing. Examples of machine learning algorithms that have been applied in NLP research include genetic algorithm, decision tree, support vector machine (SVM), hidden Markov model (HMM), entropy model, etc.

2.2.3 Natural Language Processing tasks

Though NLP tasks are obviously very closely interweaved but they are used frequently, for convenience. Some of the task (Basic task) such as automatic summarization, co-reference analysis etc. act as subtask that are not usually considered a NLP goal in and of itself they are the means for accomplishing and solving another Basic task or a particular, larger and more complex task. Therefore, you have Information Retrieval (IR) systems that utilize NLP, as well as Machine Translation (MT), Question-Answering, etc. And some of these tasks have direct real world applications such as Machine translation, Named entity recognition, Optical character recognition etc. **[12]** [13]

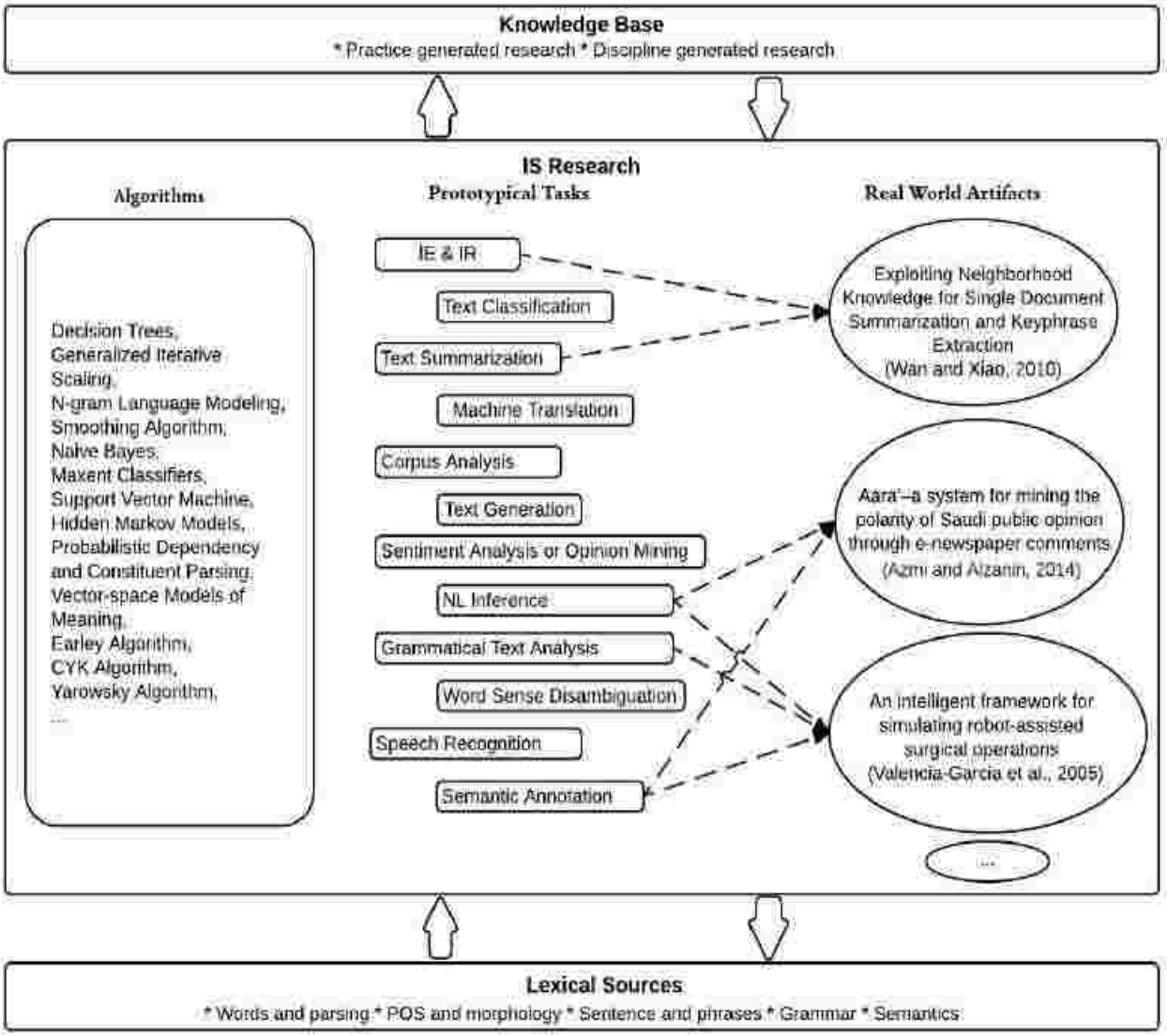

Figure 2.1: Roadmap for NLP research in IS [8]

2.2.3.1 Basic tasks of Natural Language Processing

a) **Tokenization** the task of segmenting running text into words, and normalization, the task of putting words/tokens in a standard format.

b) **Lemmatization** the task of determining that two words have the same root, despite their surface differences. For example, the words sang, sung, and sings **are forms of** the verb sing.

c) **Stemming** refers to a simpler version of lemmatization in which we mainly just strip suffixes from the end of the word .

d) **Named Entity Recognition** it describes a stream of text, determine which items in the text relates to proper names: a person, a location, an organization. The term is commonly extended to include things that aren't entities including dates, times, and other kinds of temporal expressions, and even numerical expressions like prices.

e) **Part of Speech Tagging** it describes a sentence, determines the part of speech for each word.

f) **Co-Reference Resolution** refers to a sentence or larger set of text that determines which word refer to the same object

g) **Discourse Analysis** refers to the task of identifying the discourse structure of connected text

h) **Morphological Segmentation** on which refers to separate word into individual morphemes and identify the class of the morphemes

i) **Optical Character Recognition** it gives an image representing printed text, which help in determining the corresponding or related text. [15]

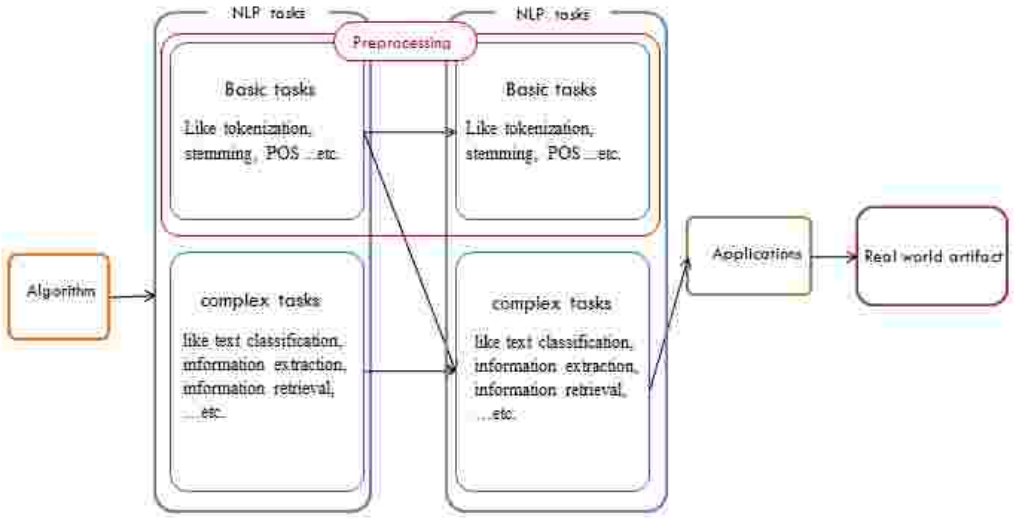

Figure 2.2: General Organization for NLP Tasks

2.2.3.2 Complex NLP Tasks

Once NL is processed into machine-readable formats, various prototypical and complex NLP tasks can be carried as a following: **[8]**

a) **Text summarization (TS) produces** an understandable summary of a set of text and provides summaries or detailed information of text of a known type.

b) **Machine Translation (MT)** which refers to automatic translation of text from one human language to another.

c) **Question-Answering** determine human-language question answer.it is related to the Web. This is a generalization of simple Web search, where instead of just typing keywords, a user might ask complete questions. Some of these questions such as definition questions, or simple factoid questions like dates and locations, can already be answered by search engines. But answering more complicated questions might require extracting information that is embedded in other text on a

Web page, doing inference (drawing conclusions based on known facts), or synthesizing and summarizing information from multiple sources or Web pages. [14]

d) **Text classification or categorization (TC)** is the task of classifying an entire text by assigning it a label drawn from some set of labels.

e) **Information extraction (IE)** is the task of extracting structured information from unstructured information embedded in texts. [15] The first step in most IE tasks is to find the proper names or named entities mentioned in a text. [16]

f) **Information retrieval (IR)** is the task of finding the information resources document from several documents in some collection that best matches a query.

g) **Text generation** is the task of Converting information from some (typically nonlinguistic) internal Representation like computer databases or semantic intents into readable and understandable human language. [17]

h) **Sentiment analysis or opinion mining** which refers to extraction of sentiment, the positive or negative orientation that a writer expresses toward some object. It is especially useful for identifying trends of public opinion in the social media, for the purpose of marketing.

i) **Semantic annotation** is the process of tying ontological definitions to natural text by providing class information to textual instances Described as a mediator platform between concepts and their worded representations, Semantic Annotation as metadata can automate the identification of concepts and their relationships in documents. Semantic annotation enriches documents, enabling access on the basis of a conceptual structure. This aids information retrieval from heterogeneous data enabling users to search across resources for entities and relations instead of words. [18]

j) **Word sense disambiguation (WSD)** Many words have more than one meaning; we have to select the meaning which determining the correct sense of a word in context. typically given a list of words and associated word senses, e.g. from a dictionary or from an online resource such as WordNet.

k) **Speech recognition** The task of speech recognition is to convert speech into a sequence of words by a computer program. As the most natural communication modality for humans, the ultimate dream of speech recognition is to enable people to communicate more naturally and effectively. One example of a useful such task is a conversational agent. that use automatic speech recognition and natural language understanding tasks.

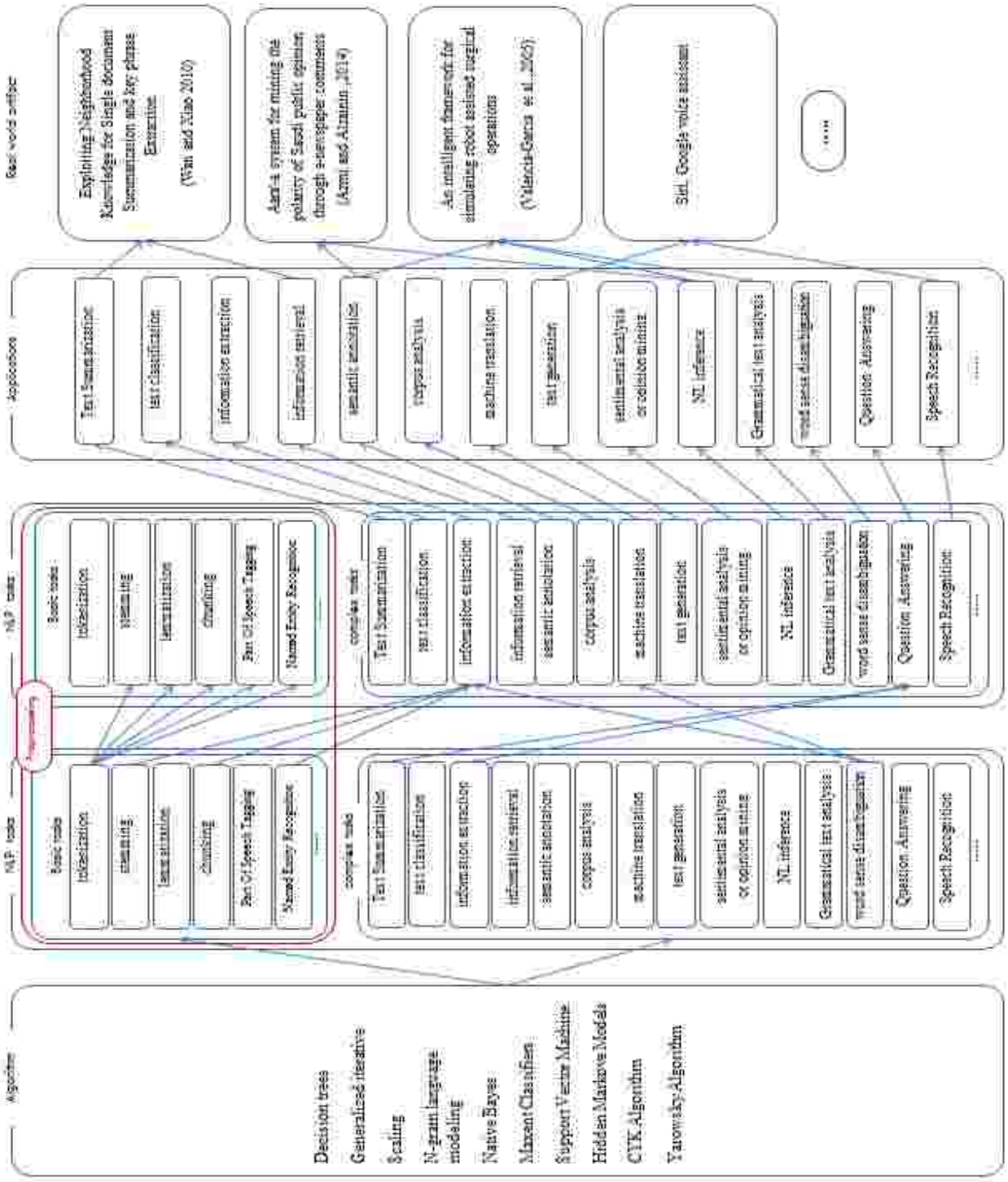

Figure 2.3: Detailed Organization for NLP Tasks

2.2.4 Natural Language Processing levels

The most explanatory method for presenting what actually happens within a Natural Language Processing system is by means of the 'levels of language' approach. This is also referred to as the synchronic model of language and is distinguished from the earlier sequential model, which hypothesizes that the levels of human language processing follow one another in a sequential manner. Psycholinguistic research suggests that language processing is much more dynamic, as the levels can interact in a variety of orders. Introspection reveals that we frequently use information we gain from what is typically thought of as a higher level of processing to assist in a lower level of analysis. For example, the pragmatic knowledge that the document you word that has several possible senses is encountered, and the word will be interpreted as having the biology sense. the more capable an NLP system is; the more levels of language it will utilize [7]. Linguistic analysis is closely tied to NLP 7 levels of linguistic analysis:

2.2.4.1 Phonetic or Phonological level: how words are pronounced.

2.2.4.2 Lexical level: word level analysis including lexical meaning and Part-Of-Speech (POS)analysis.

2.2.4.3 Morphological level: prefixes, suffixes and roots analysis.

2.2.4.4 Syntactic level: grammatical analysis of words in a sentence.

2.2.4.5 Semantic level: determining the possible meanings of sentences.

2.2.4.6 Discourse level: interpreting structure and meaning for texts larger than a sentence.

2.2.4.7 Pragmatic level: understanding the purpose of a language.

More details on levels of linguistic analysis are presented as the following:

- **Phonetic or Phonological level of Analysis:**

This level deals with the interpretation of speech sounds (pronounce) within words. There are three types of rules used in phonological analysis:

a. Phonetic rules: It is used for sound within words.

b. Phonemic rules: It is used for variations of pronunciation when words are spoken together.

c. Prosodic rules: It is used to check for fluctuation in stress and intonation across a sentence.

In an NLP system that accepts spoken input, the sound waves are analyzed and encoded into a digitized signal for interpretation by various rules or by comparison to the particular language model being utilized [7].

- **Lexical level of Analysis:**

Lexical level also called token generation, is the process of dividing the whole chunk of text into paragraphs, sentences. and separating non-words such as punctuation are from the words **[**Error! Reference source not found.**]**. It is composed of the following processing steps: tokenization, sentence splitting and POS tagging. Tokenization is the first step in lexical analysis and it is used to identify words and numbers in sentences. Sentence splitting identifies sentence boundaries of a given text. POS-tagging enables NLP systems to interpret the meaning of individual words. According to this step, each word from the NL text corresponds to a particular part of speech based on the context in which it occurs. The most probable POS tag is assigned to a word taking into consideration the relationship with adjacent and related words in the phrase, sentence or paragraph Hence, POS tagging identifies words as nouns, verbs, adjectives, etc. For example, a NNI tag signifies a singular noun, while VBB indicates the base form of a verb **[7]** [13].

- **Morphological level of Analysis:**

Morphological analysis deals with the componential nature of words and how that affects their grammatical status. It describes a set of relations between words' surface forms and lexical forms. A word's surface form is its graphical or spoken form, and the lexical form is an analysis of the word into its lemma (aka its dictionary form) and its grammatical description. A word from the NL text is composed of morphemes. It is important to mention here that a morpheme is the smallest piece of word that carries a meaning. A NLP system can understand the meaning of a word analyzing it into its constituent morphemes (lemma). At this level of analysis, there are used some stemming algorithms to remove the morphological affixes (such as prefixes and suffixes) of each word from the NL text, in order to achieve the root form of the word. This task is more precisely called inflectional morphology. The information gathered at the morphological stage prepares the data for the syntactical stage which looks more directly at the target language's grammatical structure **[7]** [13].

- **Syntactic level of Analysis:**

The syntactic level focuses on analyzing the words in a sentence using a grammar in order to reveal the structural dependency relation-ships between words. The grammar provides syntax rules of the target language about possible organization of words in sentences. The output of this level of linguistic processing is a parse tree that rep-resents the syntactic structure of a given sentence. The purpose of syntax analysis is to check that a string of words is well-arranged and to break it up into a structure that shows the relationships between the different words. A syntactic

analyzer performs this using a dictionary of word definitions (the lexicon) and a set of syntax rules (the grammar) **[7]** [13].

- **Semantic level of Analysis:**

The semantic level of processing determines possible meanings of sentences examining the meaning of constituent words. This level includes semantic disambiguation of polysemous (polysemantic) words identifying the appropriate meaning of a word looking at the rest of the sentence. Semantics recognizes that most of the words have more than one meaning, based on their dictionary and context meaning. The semantic analysis focuses on the interactions among word-level meanings in the sentence in an analogous way to morphological disambiguation of words. This level permits only one sense of the multiple senses of a word to be selected and included in the semantic representation of the sentence. For the semantic analysis, different dictionaries and knowledge bases are used. The most known example is the lexical database WordNet3 which is used to disambiguate the lexical structures of the NL text and to find semantically similar terms [13].

- **The Discourse Level of Analysis:**

While syntax and semantic level deal with sentence-length units, the discourse level works with units of text longer than a sentence trying to understand the role of a piece of information that exists in a particular document. Because the meaning of an individual sentence may depend on the sentences preceding it and may influence the meanings of the sentences that follow it [11]. This level of linguistic analysis examines the structure of a given NL text, making connections between component sentences in order to understand and represent meaning of the text as a whole [13].

- **The Pragmatic Level of Analysis:**

Pragmatic analysis is concerned with how the external world knowledge impacts the meaning of the NL text. This level of analysis depends on a body of knowledge that comes from outside the contents of the document in order to gain understanding about purpose and goal of the given text. Pragmatic level of analysis eliminates ambiguities in the text in order to find its actual meaning and the speaker's intention [13]. This is accomplished by identifying ambiguities encountered by the system and resolving them using one or more types of disambiguation techniques.

2.2.5 Challenges of Natural Language Processing

Ambiguity is explained as “the problem that an utterance in a human language can have more than one possible meaning.

- **Types of Ambiguity:**
  - **Syntactic Ambiguity** is present when more than one parse of a sentence exists. “He lifted the branch with the red leaf.” The verb phrase may contain “with the red leaf” as part of the imbedded noun phrase describing the branch or “with the red leaf” may be interpreted as a prepositional phrase describing the action instead of the branch, implying that he used the red leaf to lift the branch.
  - **Semantic Ambiguity** is existent when more than one possible meaning exists for a sentence as in “He lifted the branch with the red leaf.” It may mean that the person in question used a red leaf to lift the branch or that he lifted a branch that had a red leaf on it.
  - **Referential Ambiguity** is the result of referring to something without explicitly naming it by using words like “it”, ‘he” and “they.” These words require the target to be looked up and may be impossible to resolve such as in the sentence: “The interface sent the peripheral device data which caused it to break”, it could mean the peripheral device, the data, or the interface.
  - **Local Ambiguity** occurs when a part of a sentence is unclear but is resolved when the sentence as a whole is examined. The sentence: “this hall is colder than the room,” exemplifies local ambiguity as the phrase: “is colder than” is indefinite until “the room” is defined [7].

2.2.6 Natural Language Processing Applications

In order to gain a better sense of the overall utility of NLP, it is important to look at some of the ways that it is being implemented and used. Accordingly, in this section show some of NLP Applications and brief description as following:

2.2.6.1 Spell and Grammar Checking: these applications Checking spelling and grammar. Then Suggesting alternatives for the errors. E.g. Google as shown in figure 2.4.

Figure 2.4: Google Spell and Grammar Checking

2.2.6.2 Word Prediction: these applications Predicting the next word that is highly probable to be typed by the user. E.g. Google as shown in figure 2.5.

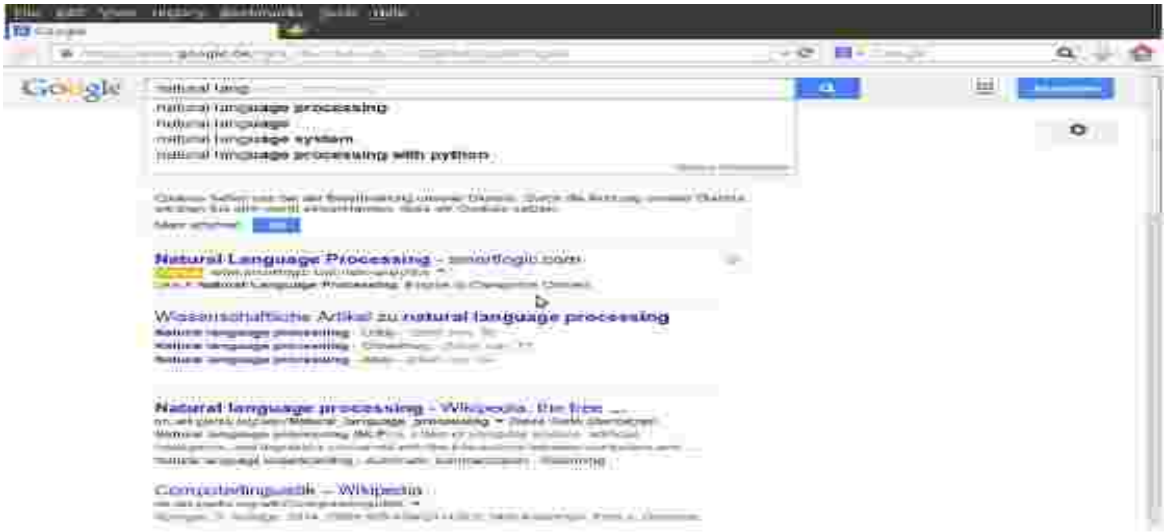

Figure 2.5: Google Word Prediction

2.2.6.3 Information Retrieval: these applications Finding relevant information to the user's query. E.g. Google research engine as shown in figure 2.6.

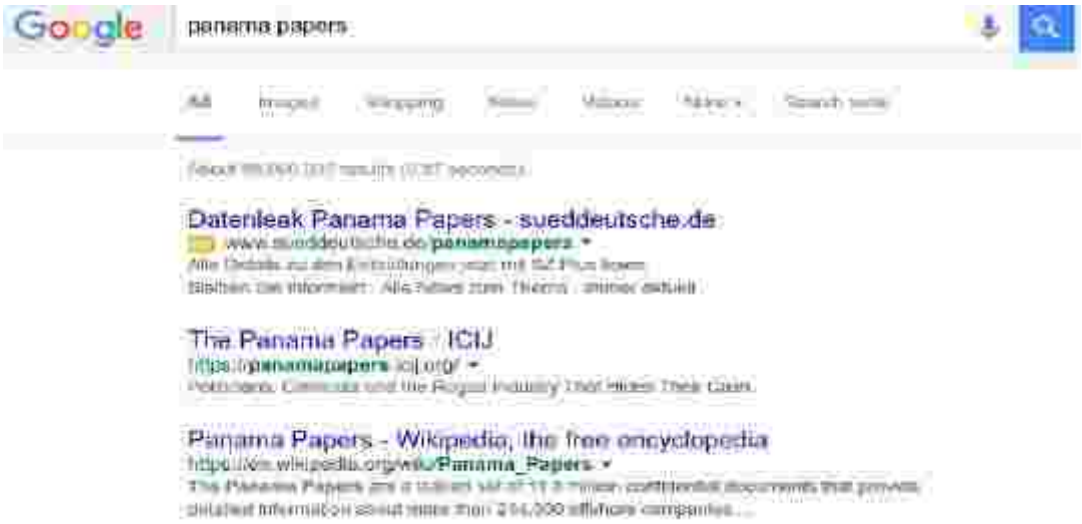

Figure 2.6: Google Information Retrieval

2.2.6.4 Text Categorization: these applications Assigning one (or more) pre-defined category to a text. this saves much time by doing the work that is to be done by staff or human indexers. E.g. email spam filters and uClassify as shown in figure 2.7.

The uClassify application is available at:

http://www.uclassify.com/browse/mvazquez/News-Classifier

Figure 2.7: uClassify Text Categorization.

2.2.6.5 Summarization: these applications Generating a short summary from one or more documents, sometimes based on a given query. E.g. SMMRY as shown in figure 2.8. This application is available at: http://smmry.com/

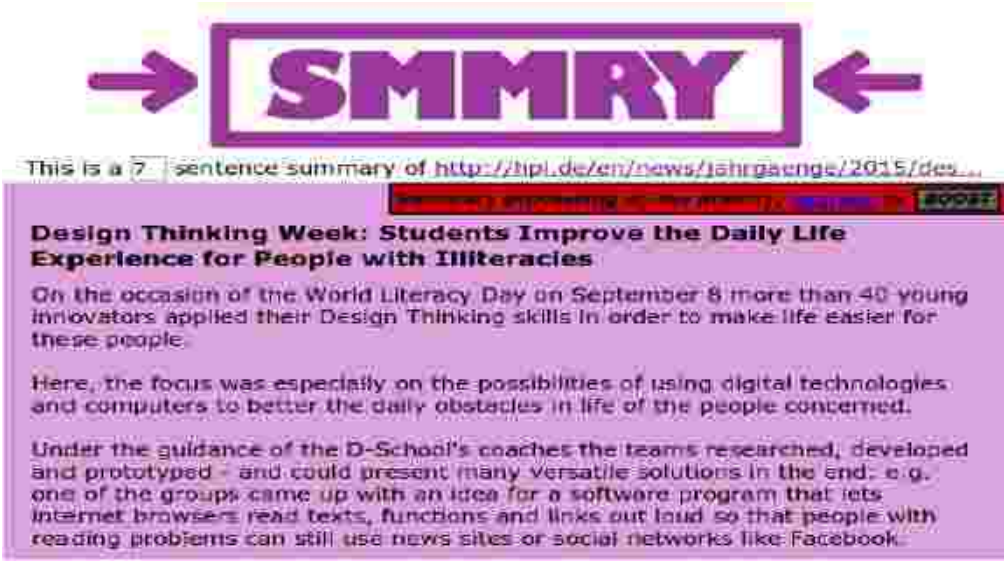

Figure 2.8: SMMRY Summarization application

2.2.6.6 Question answering: these applications Answering questions with a short answer. E.g. START as shown in figure 2.9.

This application is available at: http://start.csail.mit.edu/index.php

Figure 2.9: START Question Answering Application

2.2.6.7 Information Extraction: these applications Extracting important concepts from texts and assigning them to slot in a certain template. E.g. WIKIPEDIA as shown in figure 2.10.

Figure 2.10: WIKIPEDIA Information Extraction application

They Includes named-entity recognition (figure 2.11) e.g. lancet as shown in figure 2.12.

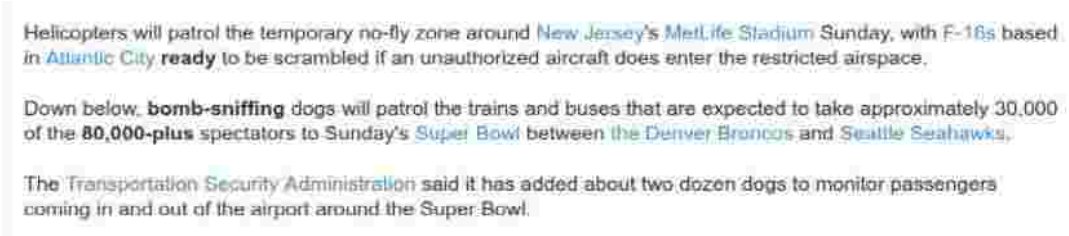

Figure 2.11: Information Extraction application

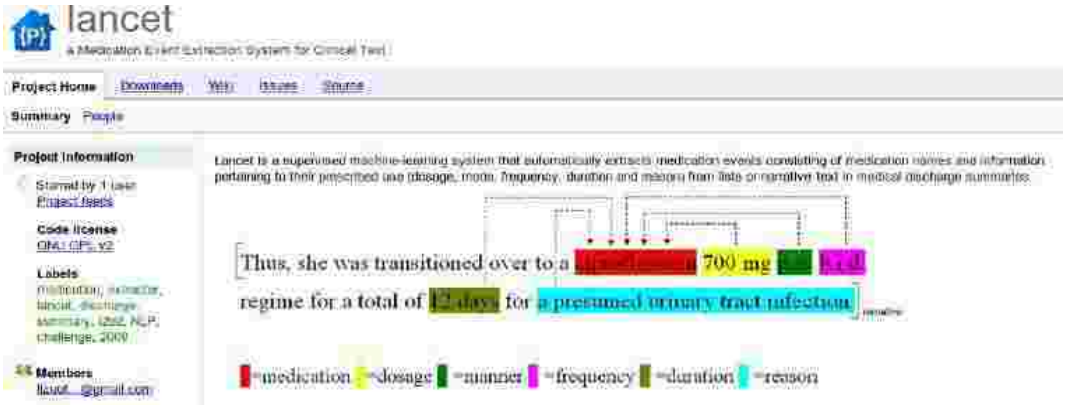

Figure 2.12: lancet information Extraction application.

2.2.6.8 Machine Translation: these applications translating a text from one language to another. E.g. Google translation.

2.2.6.9 Sentiment Analysis: these application identifying sentiments and opinions stated in a text as shown in figure 2.13.

Figure 2.13: Sentiment Analysis application.

2.2.6.10 Optical Character Recognition: these applications recognizing printed or handwritten texts and converting them to computer-readable texts. A lot of this application is available as mobile applications e.g. Google translation and Mobile OCR as shown in figure 2.14.

Figure 2.14: Mobile OCR Optical Character Recognition application.

2.2.6.11 Speech recognition: these applications Recognizing a spoken language and transforming it into a text. E.g. Siri as shown in figure 2.15.

Figure 2.15: Siri Speech recognition application.

2.2.6.12 Speech synthesis: these applications Producing a spoken language from a text. E.g. Siri as shown in figure 2.15.

Figure 2.16: Speech synthesis application.

2.2.6.13 Spoken dialog systems: these applications Running a dialog between the user and the system. E.g. Siri and IBM Waston Developer Cloud as shown in figure 2.17.

Figure 2.17: Siri and IBM Watson Developer Cloud Spoken dialog systems.

2.3 The Arabic Language

The Arabic language is both challenging and interesting. It is interesting due to its history [20] and challenging because of its complex linguistic structure **[21].** Culturally, the Arabic language is closely associated with Islam and with a highly esteemed body of literature [22]. Arabic is a Semitic language spoken by more than 330 million people as a native language, in an area extending from the Arabian/Persian Gulf in the East to the Atlantic Ocean in the West. Semitic languages are characterized by i) a lexicon built mainly from triliteral and quadriliteral roots ii) a writing system from right to left and iii) an alphabet of Abjad nature where only consonants, not vowels, are represented among the basic graphemes [23]. Moreover, it is the language in which 1.4 billion Muslims around the world. It belongs to the Semitic languages branching family of the Asian-African Languages Group. Arabic is a highly structured, derivational and very rich language where complex morphology plays a very important role [22], [23], **[24], [25], [26],** [27]. Arabic refers to three linguistic entities: **1.** Arabic spoken varieties, the natural languages of the Arab people **2.** Modern Standard Arabic (MSA) **3.** Classical Arabic (CA). Classical Arabic represents the language spoken by the Arabs more than fourteen centuries ago, while Modern Standard Arabic is an evolving diversity of Arabic with constant borrowings and innovations proving that Arabic reinvents itself to meet the changing needs of its speakers [28]. The morphology of Arabic poses special challenges to computational natural language processing systems. The exceptional degree of ambiguity in the writing system, the rich morphology, and the

highly complex word formation process of roots and patterns all contribute to making computational approaches to Arabic very challenging.

2.3.1 Arabic's Growth on the Internet

Arabic is the fifth most spoken language in the world with over 200 million speakers from northwest Africa to southwest Asia and the fastest growing one on the Internet. The advent of Arabic script support on the Internet has made it the fourth most used language online and the most growing one with an estimated growth rate of 5296.6% of Arab users between 2000 and 2013. Arabic is also the 6th most used language on Twitter in the year of 2011 and the 9th most used one on Facebook in 2012. Internet penetration has reached 39.6% of the population of Arab speaking countries, and 135.6 million Arab-speaking people are estimated to be present online. Subsequently, as every growth brings about change, content in the Arabic language has known an equivalent increase and should be met with means to process it and analyze it. These trends have encouraged scholars and researchers to look at Arabic Natural Language Processing (ANLP) and to build tools capable of keeping up with the amount of data produced every day and with its atypical formats. Another reason might be the poor quality of search engines for Arabic, as even those 1% websites are hard to navigate through and stumble upon given the poor indexing systems for Arabic. This difference has encouraged scholars and researchers to look at Computational Linguistics for Arabic, the same way it has prompted Arab entrepreneurs to create Arabic content and means of processing it to put an end to "Arabic-speakers' second-class experience of the internet". As previously mentioned, many giants of the tech world have already integrated Arabic (Google, Firefox, Bing) into their platforms [35].

2.3.2 Arabic's Popularity as a Second-fastest growing Language

Another motivation for ANLP is the fact that it's one of the most popular second languages in the world as well as "the fastest growing at US colleges and universities" according to an enrolment survey conducted by the Modern Language Association (MLA). The National Security Language Initiative (NSLI) introduced programs to learn Arabic in 2006 and described it as a "critical language". This interest in Arabic has created a need for more textbooks, resources, teachers, and for better ways of learning the language, which in turn has created a need for more research and scholarship, namely in computational linguistics for machine translation, natural language processing, Information retrieval and extraction etc. [28]

2.3.3 Arabic Language Features

The Arabic language is an inflectional language and not an analytic language. The derivation in Arabic is based on morphological patterns and the verb plays a greater inflectional role than in

other languages. Furthermore, Arabic words are built-up from roots representing lexical and semantic connecting elements. Arabic offers the possibility of combining particles and affixed pronouns to words. In other words, Arabic allows a great deal of freedom in the ordering of words in a sentence. Thus, the syntax of the sentence can vary according to transformational mechanisms such extra position and fronting, or according to syntactic replacement such as an agent noun in place of a verb. The Arabic language is distinguished by its high context sensitivity in several directions. On the writing level, the shape of the letter depends on the letter that precedes it and the one that follows it. On the syntactic level, the different synthetic coherence relations such as case-ending, matching, connecting, associating and pronominalizing represent various examples of syntactic sensitivity. The context sensitivity feature is not only limited to letters, words, and sentences. Arabic sentences are embedded and normally connected by copulatives, exceptive and adversative particles. For this reason, it is more difficult to identify the end of an Arabic sentence than is the case in other languages. Also, the "Shadda" in Arabic language represents a higher accent on the character (in other languages Shadda is represented by doubling the character when writing). Then we can have two words: one with "Shadda" and another the same as the first one but without "Shadda"; these two words can have different signification **[35]**.

2.3.4 Arabic Orthography

Arabic is a Semitic language written in a consonantal alphabet (or "abjad") with 29 basic graphemes, and it is read and written from right to left. Arabic words are formed from abstract forms named roots. The most characteristically Semitic feature of Arabic orthography is its rich morphology [30]which is based largely on a concatenative "root-and-pattern" [31]. The roots generally consisting of three or four consonants and give the basic lexical meaning of the word [32], and the pattern (noun-form or verb-form) supply specific grammatical information such as number, tense, person, gender etc. Words create identical forms (homographs) which may be read in different ways and have different meanings. For example, the unpointed word (ktb) كتب has a number of reading options: - كُتِبَ "kutiba" (had been written); كَتَبَ "kataba" (wrote); "kutub" كُتُبْ (books) [30].

2.3.5 Arabic Morphology

Arabic has a rich and complex morphology. This is due to its numerous linguistic features, such as gender, number, mood and case, and the existence of two types of morphemes: Templatic and affixation (concatenative). Templatic morphemes come in three types that are equally needed to create a word stem: roots, patterns and vocalisms. The root morpheme is a sequence of typically three consonants (termed radicals) that signifies some abstract meaning shared by all its derivations. The pattern morpheme is an abstract template in which roots and vocalisms are inserted. The vocalism morpheme specifies which short vowels to use with a pattern. An Arabic

word is constructed by first creating a word stem from templatic morphemes, then adding affixational morphemes. The process of combining morphemes involves a number of morphemic, phonological and orthographic rules that modify the form of the created word so it is not a simple interleaving or concatenation of its morphemic components [34].

2.3.6 Significance of ANLP for the Arabic-Speaking Population

ANLP applications developed in the Arab World have different objectives and usually employ both rule-based and machine-learning approaches. The following are some of the objectives of ANLP for the Arab World:

2.3.6.1 Transfer of knowledge and technology to the Arab World: Most recent publications in science and technology are published in the English language and are not accessible to Arab readers with little or no competence in English. To use human translators to translate such a huge amount of data to Arabic is very costly and time consuming. So Arabic NLP could help reduce the time and cost of translating, summarizing, and retrieving information in Arabic for Arab speakers.

2.3.6.2 Modernize and fertilize the Arabic language: Translating new concepts and terminology into Arabic involves coinage, Arabization, and making use of lexical gaps in the Arabic language. This will positively affect the revitalization of the Arabic language and enable it to fulfill the essential needs for its speakers.

2.3.6.3 Improve and modernize Arabic linguistics: Arabic NLP needs a more formal and precise grammar of Arabic than the traditional grammar so widely employed today. Innovation is needed as well to preserve the valuable heritage of traditional Arab grammarians.

2.3.6.4 Make information retrieval, extraction, summarization, and translation available to the Arab user: The hope is to bridge the gap between peoples of the Arab world and their peers in more technically advanced countries. By making information available to Arabic speakers in their native language, Arabic NLP tools empower the present generation of educated Arabs. Thus Arabic NLP tools are indispensable in the struggle of Arabic speakers to attain parity with the rest of the world which is, in turn a matter of national security to the Arab World [46].

2.4 Arabic Natural Language Processing(ANLP)

Arabic morphological analysis and generation are important to many NLP applications such as machine translation [37] and information retrieval. As a natural language, Arabic is unique in terms of its history, diglossic nature, internal structure, attached link with Islam, and the Arabic culture and identity. Any Arabic NLP system that does not take the specific features of the Arabic

language into account is certain to be inadequate **[38], [39]**. The challenge the Arabic language poses to researchers is not limited to the social aspects of the language, but also extends to its inherent linguistic structure [40].

2.4.1 Arabic Natural Language Processing Techniques

In ANLP area, researchers have proposed several techniques for analyzing the Arabic Language. These techniques are basically categorized into three approaches:

2.4.1.1 Rule-Based Approach

Rule-based approach is a traditional method that is concerned, mainly, with European languages.

I) **In** [47], authors proposed system automates the grammar analysis of Arabic language sentences utilizing the rule-based frameworks. The proposed system consists of two consequent phases: the lexical (morphological) analysis and the syntactic analysis. The first phase, the lexical analysis, has two tasks: the first task where the input stream (words) are broken into morphological items(morphemes). These morphemes go in two ways: the first way to form a single free form word (unbounded), while the other to form a complexed form inflected word(bounded). The second task is assigning a suitable symbol to each lexeme(word). The second phase is the syntax analysis. It has two tasks. The first task is determining all Arabic rules and, then, writing the equivalent Context Free Grammar(CFG). The second task is choosing and building the recursive parser which has a top-down technique which is built from a group of mutually-recursive procedures that involve implementing the grammar rules for each case.

In summary, the syntax analysis receives all the tokens and finds the best grammar for the given sequence of the tokens by using CFG. The system considers only verbal sentences with different analysis forms. In addition, the proposed framework is restricting to right sentences, lexically and grammatically, with verbs in the active voice only.

II) **In** [48], authors proposed in this work presented an efficient top-down chart parser that parses simple Arabic sentences. The CFG has been used to represent the Arabic grammar depending, solely, on Arabic grammar rules to determine the sentence structure. The grammar rules encode the syntactic and the semantic constrains that help resolve the ambiguity of parsing Arabic sentences. The proposed parsing technique provides a promising impact on

many language applications such as question answering and machine translation. That is because the source sentences are analyzed according to the grammar rules that go with the sentence's meaning. Consequently, the syntactic and semantic ambiguity is reduced. Using the proposed top-down chart parser has advantages over the other existing approaches as follows:

- It analyses both Arabic nominal and verbal sentences regardless their length.
- It uses efficient parsing techniques, the top-down chart parser, which showed the effectiveness of the system for analyzing both verbal and nominal sentences.

In summary, this study presents a parsing system using the top-down chart parser technique. The system consists of three steps: word classification, Arabic grammar identification using CFG, and parsing. The system was tested on 70 sentences with different sizes, from 2-6 words, achieving accuracy of 94.3%.

III) **In [49], [50],** authors used a parsing-based technique to resolve the disambiguity in Arabic texts. He constructed an Arabic parser using Xerox linguistics environment to write grammar rules and symbolics that follow the LFG formalisms. Attia verified his approach on short sentences arbitrarily that were selected from a corpus of news articles. The accuracy obtained was 92%.

IV) **In** [51], authors developed a new parser aiming to analyze and extract the attributes of Arabic words. The methodology was, mainly, based on studying and analyzing the Arabic grammar rules conforming to gender and number. Then, it formulizes the rules using the CFG- the context free grammar. After that, the system uses transition networks for representing the rules, and then constructing a lexicon of words that construct the sentence structure, implementing the recursive transition network parser and evaluating the system using real Arabic sentences.

A top-down algorithm technique with a recursive transition network was used in the parser development. The efficiency of the developed parser was put to evaluation using a sample of 90 sentences for testing. The results showed that 85.6% of the sentences were parsed successfully, 2.2% of sentences were parsed unsuccessfully and 14.4% of the sentences were not parsed for various reasons such as lexical problem (4.4%), incorrect sentences (2.2%), or not recognizable by linguists according to Arabic grammar rules (5.6%). In

conclusion, the parser was an efficient and a satisfactory system due to its remarkable performance.

V) **In** [52], authors proposed built a parser to test whether the syntax of an Arabic sentence is grammatically right or not by building new effective Context-Free Grammar. A Top-Down technique was used in this model for parsing schemes constructing a parse from the initial symbol {S}. The method of this system that it chooses a production rule and tries to match the input sentence's words with the chosen rule. A set of experiments were made on a dataset that holds 150 Arabic sentences. The system reached an average accuracy of 95%.

VI) **In** [53], authors proposed in this paper an Arabic bottom-up chart parser based on rule-based approach. This method was devised in order to analyze the Modern Standard Arabic (MSA) sentences and judge the syntax which leads to reduce their ambiguity. The process consists of performing a morphological analysis based on the Augmented Transition Network (ATN) technique which is used to signify the context-sensitive information regarding the relation of the stem and the inflectional extractions. The augmented transition network (ATN) contains a total number of 170 rules that are partitioned into 22 groups, each group has a grammatical category (such as: the subject, the object, defined, conjunction forms, etc...) were used. Also, additional linguistic features, such as lexical and semantic features, have been used to disambiguate the sentence.

The treebank is an example for rule based approach.

- **Arabic Treebanks**

Treebanks are "collections of manually checked syntactic analyses of sentences" and they're resources used to build statistical parsers which are 'trained on them'. Treebank annotations are used for **tokenization** (splitting words and sentences into tokens), **morphological disambiguation**, and many other applications of ANLP, three Treebanks are worth describing and pointing to: The Penn Arabic Treebank (PATB), the Prague Arabic Dependency Treebank (PADT), and the Columbia Arabic Treebank (CATiB). A lot of effort went into the making and annotating of these Treebanks as they enable "important research in general NLP applications". The Penn Arabic Treebank project which started in 2001 at the University of Pennsylvania and at the Linguistic Data Consortium (LDC) was a pioneer in the field. It is "annotated for morphological

information" and it models the English version of the Penn Treebank. It was the first major effort for Arabic, and it remains the most required one as CATiB and PADT both used it and converted it to their "own representation" and then annotated the data. The Columbia Treebank, was an attempt at speeding up production and "avoiding annotation of redundant linguistic information", as both PATB and PADT were found to contain way too much information that doesn't always get processed and only slows down production. Habash describes the efforts as an attempt to ease "training annotators who no longer need to have degrees in linguistics". Statistical parsers are thus trained on these Treebanks depending on the grammar they follow and on the eventual use of the parse trees. [49]

2.4.1.2 Statistical-Based Approach

I) In [54], authors proposed a system called "MADA+TOKAN" which is one of the greatest well-known Arabic NLP systems. This framework implements morphological disambiguation, POS tagging, diacritization, lexicalization, lemmatization, stemming, etc...

The MADA+TOKAN system is made of two major parts: firstly, the MADA, which does a morphological analysis and disambiguation. The second part is TOKAN, which is a general tokenizer for the MADA-disambiguated text. The both parts collaborate together to find a solution to the different Arabic NLP problems. The system, as a whole, follows a statistical-based approach. It inspects a list of all probable analysis for each word, and then chooses the analysis that match the existing context best. This is done, in addition, by support vector machine models that has 19 different weighted morphological features. The MADA+TOKAN's chosen analysis includes diacritics, lexemic, glossary, and morphological information. These all disambiguation tasks are made in one step.

II) In [55], authors devised a system called AMIRA which is a framework that was designed for Arabic tokenization, POS tagging, Base Phrase Chunking, and Named Entities Recognition. AMIRA is made of a clitic tokenizer (TOK), part of speech tagger (POS) and base phrase chunker (BPC)-shallow syntactic parser. The technology of AMIRA is built on supervised learning technique using Support Vector Machine(SVM) algorithm with implicit dependence on

the knowledge of deep morphology. Therefore, in contrast to other systems such as MADA, AMIRA depends on surface data to learn generalizations.

III) **In [56],** authors used the Stanford natural language processing group to develop Arabic NLP tools. This group includes a word segmenter, a part-of-speech tagger and a probabilistic parser. The dataset used in this system is the Penn Arabic Treebank. The Arabic Stanford word segmenter, Stanford tagger, and Stanford tagger are all written in java code, and based on machine learning technique using a Conditional Random Field model, Maximum-Entropy probabilistic context free grammar (PCFG) depending on the hand-parsed sentences, consecutively.

IV) **In [57],** authors presented a supervised learning technique using a support vector machine algorithm(SVM) for Arabic Base Phrase Chunking(BPC). The system achieved an F-score of 96.33% over 10 base phrase chunk types. Diab versified the feature sets according to two factors: the usage of explicit morphological features, and the usage of different part of speech (POS) tag sets. 75 POS tags were used that represented information about definiteness, gender and number features. In details, ERTS comprises 75 tags. For the current system, only 57 tags are initiated. The author developed a POS tagger based on this new set. Also the author adopted the YAMCHA sequence model based on the TinySVM classifier. The tagger trained for ERTS tag set uses lexical features of +/-4 character ngrams from the beginning and end of a word in focus. The context for YAMCHA is defined as +/-2 words around the focus word. The words before the focus word are considered with their ERTS tags. The kernel is a polynomial degree 2 kernel. The author adopts the one-vs.-all approach for classification, where the tagged examples for one class are considered positive training examples and instances for other classes are considered negative examples.

V) **In [58],** authors built the Columbia Arabic Treebank (CATiB), which is a database of syntactic analysis of Arabic sentences. CATiB is distinguished for its speed despite some constraints regarding linguistic richness. Two basic ideas encourage using the CATiB approach: 1) no annotation of redundant information and 2) using illustrations and terminology inspired by traditional Arabic syntax. The grammar analysis is done by applying a guileless parsing approach.

VI) **In [59],** authors built the Quranic Arabic Dependency Treebank (QADT), which is an annotated grammar resource containing of 77,430 words from the Quran. This project offers a language training model, Hidden Markov model part-of-speech taggers, based on traditional Arabic grammar affiliated by a Linguistic research in the Quran that uses the annotated corpus, an automatic categorization of Quranic chapters, and prosodic analysis of the text.

VII) **In [60],** authors presented an approach for parsing Arabic sentences based on supervised machine learning using Support Vector Machine (SVMs). This system selects syntactic labels of the sentence. This proposed method has two stages: 1) the learning stage and 2) the prediction stage. The first stage is based on a training corpus, extraction features, and a set of rules that are obtained from the corpus of learning. The second stage implements the results of learning obtained from first stage to accomplish parsing. Promising results for this method were achieved with an f-score of 99%.

VIII) **In [61],** authors presented a methodology of using models with access to additional information of exact syntactic analysis and rules to offer an enhanced estimation of case and state. The expected case and state values are, then, used to re-tag the Arabic morphological tagger MADAMIRA output by choosing the best match its graded morphological analysis. They edge their retagging to nominals. Since what they are learning to expect is how to correct MADAMIRA's baseline choice (as opposite to a generative model of case-state), they also re-apply the model on its output to repair mainly spreader agreement errors in a way similar to agreement classifier.

IX) **In [62],** authors proposed in this paper the writers extend the Morphological Analysis and Disambiguation of Arabic (MADA) system. They reused the tool and training set features, that had been used by others, to improve the results and make it easier for comparison. The improvements of results in all categories in numbers: As for WER (word error rate) the Zitouni et al. mistake was reduced by 17.2%, while the DER (diacritic error rate) error diminution was only 10.9%.

2.4.1.3 Hybrid-Based Approach

Hybrid-based approach is a combination of both the rule-based and the statistical annotated corpora approaches. This method is used for the following reasons: firstly, the shortage of language resources, such as parallel or bilingual big corpora, and secondly

that the Arabic language, although it is rich and has an availability of corpora, suffers data scarcity. The above reasons encouraged several researchers to follow the rule-based approach combined with statistical annotated corpora (and that is called hybrid-based approach) for developing tools and systems in Arabic natural language processing.

- **In [64],** authors proposed a hybrid system composed of the learning-based and rule-based approaches for Arabic grammar analysis. This system showed an adequate accuracy and easy to implement. However, the system requires deep knowledge of Arabic despite the use of learning portions availability. Many experiments were made on a dataset that holds 600 Arabic sentences. The system reached an average accuracy of 90.44%.

When a sentence is inputted to the proposed framework, the system assigns each token an appropriate tag, case, and a sign. Then, the system determines for every token its POS tag, Base Phrase chunk and its morphological features (such as token definiteness). The rule-based system is responsible for determining the tag, case, and the sign of each word in the sentence. From the other side, the grammar analyzer input and features could be characterized as follows:

- "Input': A complete sentence of Arabic words.
- "Context": The whole sentence.
- "Features": To extract the grammatical role of the words in the sentence.

A stemmer, POS tagger, BP chunker, and a morphological analyzer are used to extract extra morphological features of the words in the sentence. The Arabic grammar analyzer module uses stemmer to separate proclitics and enclitics of the word. Then, the POS tagger assigns an adequate POS tag to each token. Then, the base phrase chunker groups words belonging to the same phrases. Additional morphological information is extracted for each word using the morphological analyzer. Finally, it applies the Arabic grammar rules to assign a tag, case and sign for each word.

As an evaluation of this framework, the developer of the system generated 600 sentences. The 600 sentences consisted of 3452 tokens. The overall accuracy of the tokens, that have correct tag, case and sign, was 90.44% which is a good precision for this complex task. [61]

2.4.2 Arabic Natural Language Processing Tasks

2.4.2.1 Tokenization

It (also sometimes called segmentation) refers to the division of a word into clusters of consecutive morphemes, one of which typically corresponds to the word stem, usually including inflectional morphemes [16].

2.4.2.2 **Part-of-speech tagging** (POS-tagging)

Is the process of automatically assigning the proper grammatical tag for each word in the text according to its context in the sentence. POS is implemented by assigning each token a lexical category. POS-tagging is usually the first step in linguistic analysis. Also, it is a very important intermediate step to build many natural language processing applications [45].

2.4.2.3 **Base phrase chunker**

Also known as a shallow syntactic parser, is the process of grouping related words into phrases based on their context and their dictionary-based role. Phrases, not individual words, are the base of most advance NLP process, such as machine translation, spell checking and correcting, speech recognition, information retrieval, information extraction, corpus analysis, syntactic parsing and text-to-speech synthesis systems [19].

2.4.2.4 **Parsing**

Is the process of mapping the sentence (string of words) to its parse tree. To do that, an efficient Context-Free Grammar (CFG), which defines the language rule is used, CFG in natural languages represents a formal system which describes a language by specifying how any legal text can be derived from a distinguished symbol called the sentence symbol. Furthermore, a robust syntactical analysis system to check whether the parser input sentence may generate by a given CFG is also very important step, which requires an efficient Part-Of-Speech (POS) tagging system to assign the syntactic category (noun, verb, and particle) to each word in the input sentence. The main component of the CFG is the set of production rules. For example, VP V NP, represents one of the CFG production rules that may be used to describes the context of a verbal sentence. Furthermore, CFG is represented by a recursive nesting of phrases that efficiently describes the context of all languages, which is analyzed using CFG. Arabic language as many other natural languages has nominal (NP) and verbal sentences (VP). It well known that nominal sentences begin with noun while verbal begin with verb. Parsing Arabic sentences considered a requirement to many NLP applications like information retrieval and machine translation and others [48].

2.4.2.5 **Proper Name Transliteration**

Is a specific sub-problem of machine translation focusing on mapping/ approximating the phonetic value of proper names from one language to another and typically across scripts. In the context of mapping from Arabic to/from English, we face several challenges:

- Arabic optional diacritics.

- Arabic consonants with no exact match in the Roman script.
- Dialectal variants on the pronunciation of Arabic names.
- English consonants foreign to Arabic.
- Names in English from other European languages also written in Roman script bringing their own particular orthographic and phonological challenges.

2.4.2.6 Spelling Correction

Spelling correction is often thought of as a preprocessing step that addresses the presence of spelling errors. Spelling errors can cause NLP models to be less effective and can add an often irrecoverable error margin from the first step in a system. Spelling errors can be hard to identify if the misspelled form happens to be a valid word that is contextually incorrect morphologically or semantically. the most common spelling errors in Arabic involve Hamzated Alifs and Alif-Maqsura/Ya confusion. These errors affect 11% of all words (or 4.5 errors per sentence) in the Penn Arabic Treebank (PATB). Other forms of errors including misplaced dots, joint/split words near disconnected letters, or misplaced letters occur less frequently – 0.3% of all words (at least once in around 12% of all sentences). This is still non-trivial since a single spelling error can wreak havoc on the processing of the whole sentence.

2.4.2.7 Speech Recognition and Synthesis

Speech Recognition (aka Automatic Speech Recognition – ASR; Speech-to-Text – STT) is the task of mapping an acoustic speech signal to its corresponding string of words. Inversely, Speech Synthesis (aka Text-to-Speech –TTS) is the task of producing an acoustic signal from an input word string. Much research has been done in both areas. Most of the techniques used for ASR and TTS are language independent once an appropriate level of representation is defined for the language of interest. For Arabic, the big challenge is bridging the gap between Arabic phonology and its orthography, since typically the more complex and loss the orthography, the more difficult a language is for ASR or TTS. As such, a central task for work on ASR and TTS for Arabic involves producing the phonemic or phonetic form of the Arabic text. It's been noted that diacritization alone does not predict actual pronunciation in MSA. Different researchers have described different sets of pronunciation/ phonotization rules based on MSA phonology which extend a diacritized word to a set of possible pronunciations. Some of these rules attempt to accommodate pronunciation variants to handle common failures to

produce "proper" pronunciation according to Arabic syntax and phonology even by MSA-trained speakers.

2.4.2.8 Detokenization

In certain contexts, when Arabic is the output language, it is desirable to produce proper Arabic that is orthographically correct; i.e., tokenized and orthographically normalized words should be detokenized and enriched (orthographically corrected). As an example, the output of English-to- Arabic MT systems is reasonably expected to be proper Arabic regardless of the preprocessing used to optimize the MT performance. Anything less is comparable to producing all lower cased English or uncliticized and undiacritized French. Detokenization may not be a simple task because there are several morphological adjustments that should be applied in the process. Obviously, the more complex the tokenization, the harder is detokenization.

2.4.3 Arabic Natural Language Levels

As mentioned before, the most descriptive method for presenting what actually happens within a Natural Language Processing system is by means of the 'levels of language' approach. Linguistic analysis is closely tied to NLP 7 levels of linguistic analysis:

2.4.3.1 Phonetic or Phonological level: how words are pronounced.

2.4.3.2 Lexical level: word level analysis including lexical meaning and Part-Of- Speech (POS)analysis.

2.4.3.3 Morphological level: prefixes, suffixes and roots analysis.

2.4.3.4 Syntactic level: grammatical analysis of words in a sentence.

2.4.3.5 Semantic level: determining the possible meanings of sentences.

2.4.3.6 Discourse level: interpreting structure and meaning for texts larger than a sentence.

2.4.3.7 Pragmatic level: understanding the purpose of a language.

More details on levels of linguistic analysis are presented as the following:

a) Phonetic or Phonological level

Phonology is the study of how sounds, or phones, are organized in natural languages. A central concept in phonology is the *phoneme*, the smallest contrastive unit in the sound system of a language. A phoneme can correspond to multiple *phones*, or basic sounds, that are distributed according to predictable rules, called phono tactics. The predictable phones associated with a phoneme are called its *allophones*. For example, while Arabic does not have a phoneme /p/, often causing the characteristically Arabic-accented p-b confusion in English speech, the phone [p] appears as an allophone of the phoneme /b/ in limited contexts, such as preceding a voiceless phone: the word دبس *dibs* 'molasses' is phonemically represented as /dibs/, but phonetically as

[dips]. MSA vowel phonemes are limited in number compared to English or French; however, there are many allophones to each of them depending on the consonantal context. For instance, contrast the pronunciation of the vowel /¯a/ in بأس /b¯as/ 'he kissed' and باص /b¯aS/ 'bus', which can be approximated by the English words 'bass [the fish]' and 'boss', respectively. This phenomenon is called emphasis spread. Another interesting phenomenon in MSA vowel pronunciation is the optionality of dropping the final vowel marking syntactic case in words at the end of utterances (as in the end of a sentence or in citation). This is called *waqf* (وقف) '[lit. stopping/pause]' pronunciation. There are numerous additional phonological variations that are limited to specific morphological contexts, i.e., they are constrained morpho-phonemically as opposed to phonologically. Some of these phenomena are explicitly expressed in the orthography and some are not.

b) Lexical Level of Analysis

MSA expressed differences from CA on the lexical, morphological and syntactic levels [2, 3]. These differences were summarized by [63] in the following points:

- The MSA lexicon is richer than CA lexicon, since it incorporates new words borrowed from other languages.
- In general, MSA conforms to the morphology and syntax rules of CA, but in MSA there is a greater tendency for simplification and modern writers use only a subset of the full range of structures, inflections and derivations available in CA.
- The classical word order of Object -Verb -Subject (OVS) is rarely found in MSA.
- In MSA, there is a tendency to avoid passive verb forms where the active readings are also possible.
- The relatively marginal Subject- Verb- Object (SVO) word order in CA is gaining more weight in MSA.

c) Morphological level of Analysis

Is at the level of the word and its endings. A morpheme is considered the smallest unit in grammar analysis, and it may or may not be a word. In other words, a word is formed of one or more morphemes. Arabic morphological analysis and generation have been a focus of research in natural language processing for a long time due to the complexity of Arabic morphology. There are certain desiderata that are generally expected from a morphological analysis/generation system for any language. These include (1) coverage of the language of interest in terms of both lexical coverage (large scale) and coverage of morphological and orthographic phenomena (robustness); (2) the surface forms are mapped to/from a deep level of representation that abstracts as much as possible over language-specific morphological and orthographic features; (3) full reversibility of

the system so it can be used as an analyzer or a generator; (4) usability in a wide range of natural language processing applications such as MT or IR; and finally, (5) availability for the research community. These issues are essential in the design of any Arabic morphological analysis and generation system.

d) Syntactic level of Analysis

Deals with language at the level of the sentence, it validates grammaticality and determines which combinations are possible in the given language. Syntax and morphology form grammar. Arabic has two types of sentences: verbal sentences (V-Sent) and nominal sentences (N-Sent). N-Sents are also called copular/equational sentences.

- **Verbal Sentences**

  The prototypical structure of a V-Sent is Verb-Subject-Object(s). This is expressed in different forms. The most basic form of the V-Sent consists of just a verb with a conjugated (pro-dropped) pronominal subject.1 The verb expresses the person, gender and number of the subject.

- **Nominal Sentences**

The prototypical Nominal Sentence (N-Sent) has the form of Subject-Predicate/Topic-Complement (مبتدأ و خبر mubtadaÂ wa + xabar). This is sometimes referred to as a copular construction or equational sentence. Nominal Sentence Variants: In the simplest N-Sent, the subject is typically a definite noun, proper noun or pronoun in the Nom case and the predicate is an indefinite Nom noun, proper noun or adjective that agrees with the subject in number and gender.

e) Semantic level of Analysis

Is a "hot topic" and it deals with meaning which is very hard to characterize. Arabic words represent and distinguish different aspects of meaning idiosyncratically. For example, the Arabic word **قلم** *qalam* is used for both 'pen' and 'pencil,' yet the word صلاة *SlA.h* is used for 'prayer' only in the worship (*pray to*) sense not the request (*pray for*) sense. Fantastic cliches of Arabic having a large number of words for *camel* (among others) are true. However, most Arabic speakers won't know more than a couple, particularly جمل *jamal* 'male camel', ناقة *nAqa.h* 'female camel' and إبل *Aĭibil* 'camels (collective plural)'. Other words for *camel* are part of the jargon of camel breeders and specialists, e.g., حوار *HuwAr* 'a baby camel still at its mother's side', لبون *labuwn* 'lactating camel' or خلوج *xaluwj* 'a female camel whose baby died'. This situation is comparable to the numerous words for 'horse' in the English jargon of horse breeders, e.g., *foal* 'a

baby horse still at its mother's side' or *gelding* 'a castrated male horse'. In that respect, Arabic is no different again from other languages.

f) Discourse level of Analysis

Recognizing sentence boundaries in a running text is a more difficult task in languages such as Arabic than it is in languages like English due to the absence of strict punctuation rules. In fact, it is common in Arabic discourse to write an entire paragraph without a single period except at the end of that paragraph. Sentences are often conjoined via the Arabic coordinators (و) *wa* and (ف) *fa* and Arabic discourse is characterized by excessive use of coordination, subordination and logical connectives. Capitalization and punctuation not only facilitate recognizing sentence boundaries, but also play an important role in the task of named entity recognition (NER), which has become an essential component of many NLP applications.

g) Pragmatic level of Analysis

Is the top level of linguistic analysis as it deals with the use of the language in specific situations. [64]

2.4.4 Why Arabic is a Challenge to Computational Linguists

Arabic natural language processing started peaking the interest of computational linguists in the late 70s and early eighties. Linguists soon realized that Arabic is very rich morphologically (at the level of the word) and syntactically compared to English and that further studies were needed. The challenges can be as following:

2.4.4.1 Arabic Diglossia

As we've mentioned in the introduction, Arabic has three linguistic entities:

1. Arabic spoken varieties, the natural languages of the Arab people **2.** MSA **3.** Classical Arabic or its numerous dialects. This difference results in what linguists call 'diglossia', a situation which two languages or dialects exist within the same language community. Arabic exhibits a true diglossic situation where at least three varieties of the same language are used within a speech community [38] and in defined situations. Classical Arabic is the language of religion and is used by Arabic speakers in their daily prayers while Modern Standard Arabic (MSA), a more recent variety of Classical Arabic, is used by educated people in more formal settings such as in the media, classroom, and business. With family, friends, and in the community, people speak their own regional dialect which differs greatly from region to region. How then to define diglossia features? We can define the features of diglossia according to Ferguson [1959], [39]: "a relatively stable language situation in which, in addition to the primary dialects of the language, (which may include a standard or regional standards), there is a very divergent, highly codified (often more

grammatically complex) superposed variety, the vehicle of a large and respected body of written literature, either of an earlier period or in another speech community, which is learned largely by formal education and is used for most written and formal spoken purposes but is not used by any sector of the community for ordinary conversation."

2.4.4.2 The Arabic Script

One of the key linguistic properties of the Arabic language that poses a challenge to the computational linguists of Arabic is the Arabic script itself. Although Arabic is a phonetic language in the sense that there is one-to-one mapping between the letters in the language and the sounds they are associated with, Arabic is far from being an easy language to read due to the lack of dedicated letters to represent short vowels, changes in the form of the letter depending on its place in the word, and the absence of capitalization and minimal punctuation.
However, the main problem with the Arabic script is "Diacritics" [43].

Normalization of the Arabic Script

Another challenge facing researchers and developers of Arabic computational linguistics is the dilemma of normalization. The problem arises because of the inconsistency in the use of diacritic marks and certain letters in contemporary Arabic texts. Some Arabic letters share the same shape and are only differentiated by adding certain marks such as a dot, a hamza or a madda placed above or below the letter. To manage this problem, the common practice in Arabic NLP systems is to normalize the input text [41]. For example, in order to handle the different variations in Arabic script, Larkey and Connell [2001], [44] replace the initial alif with a hamza above or below with simply an alif.

2.4.4.3 Diacritics

Diacritics are symbols added to Arabic words to indicate a Vowel or Shadda. With the absence of short vowels, two types of linguistic information are lost. The first is most of the case markers that define the grammatical function of Arabic nouns and adjectives. For example, a *Damma,* which is a high back rounded vowel at the end of a common noun or adjective marks the nominative case whereas a *fatHa*, which is a low front vowel in the final position of a common noun, marks the accusative case and a *kasra* which is a high front vowel marks the genitive case. The absence of case markers and thus the grammatical function of a word, creates multiple ambiguities due to the relatively free word order in Arabic and because Arabic is a pro-drop language. The second type of information that is lost due to the nature of the Arabic script is the lexical and part of speech information. Thus, in the absence of internal voweling it is sometimes impossible to determine the part of speech (POS) without contextual clues. For example, without

contextual clues a word like (من) *mn* could be a preposition meaning "from," a wh-phrase meaning "who" or a verb meaning "granted." The issue with diacritics is that they're optional, and that most of MSA doesn't have them which creates ambiguity. The ambiguity affects word segmentation given the rich morphology of Arabic where one word is actually composed of many morphemes which have to be tokenized for full grammatical analysis. This problem is due to the fact that Arabic allows attaching prepositions and affixes to nouns or verbs which result in syntactic ambiguity.

2.4.4.4 Ambiguity of ANLP

The many levels of ambiguity pose a significant challenge to researchers developing NLP systems for Arabic [27]. The reason is that ambiguity exists on many levels. Although ambiguity is caused primarily by the absence of short vowels, at SYSTRAN researchers have found ambiguity in Arabic to be present at every level through:

i. **Homographs:** A word belonging to more than one part of speech such as قدم *qdm* which could be a verb of Form II meaning "to introduce" or a verb of Form 1 meaning "to arrive from" or a noun meaning "foot." Some homograph ambiguity can be resolved by contextual rules. For example, an Arabic word that could be either a noun or a verb can be disambiguated by the following rule which says that such a word will be disambiguated to a noun when preceded by a preposition.

ii. **Internal word structure ambiguity:** That is, when a complex Arabic word could be segmented in different ways. For example, "ولي" wly could be segmented into "و +ل +ي" corresponding to coordinate-prep-pronoun meaning "and for me," or may not be segmented at all meaning "a pious person favored by God". An Arabic machine translation system or anArabic named entity extraction system should select the correct analysis.

iii. **Syntactic ambiguity:** Arabic is a relatively free word order language. While the primary word order in Classical Arabic and Modern Standard Arabic is verb-subject-object (VSO), they also allow subject-verb-object (SVO) and object-verb-subject (OVS). Arabic has a very rich and complex agreement system. A noun and its modifiers have to agree in number, gender, case, and definiteness. In SVO structures, a verb must agree with its subject in gender, number, and person. However, in VSO sentences the verb is always in the singular even when its subject is dual or plural. The feature definiteness plays an important role in principal formation. As in the case of a prepositional attachment as in " قابلت مدير البنك الجديد"/*qabaltu mudiir al-bank al-jadiid*/ which could mean "I met with the new bank manager" or "I met with the manager of the new bank" depending on the

internal analysis of the noun phrase. The feature definiteness plays an important role in principal formation. For example, a noun construct usually begins with an indefinite noun followed by either a definite or indefinite noun as in "مدير البنك" "the manager of the bank." The first term of the Arabic construct "مدير" "manager" is indefinite whereas the second noun "البنك" "the bank" is definite.

iv. **Semantic ambiguity:** Sentences and phrases may be interpreted in different ways. For example, "يحب علي أحمد أكثر من إبراهيم"/*yhb 'ly ahmd aktr mn abrahy*m/ "Ali likes Ahmed more than Ibrahim." Does this mean that Ali likes Ahmed more than Ali likes Ibrahim, or do Ali and Ibrahim like Ahmed, but Ali likes Ahmed more than Ibrahim likes Ahmed?

v. **Constituent boundary ambiguity:** For example, "مدير البنك الجديد" *mdyr albnk algydyd* could mean "the new manager of the bank" or "the manager of the new bank" depending on the boundary of the adjective phrase within this noun construct.

vi. **Anaphoric ambiguity:** As in علي قال أنه نجح /*qala Ali annahu najah*/Ali said that he succeeded. This sentence is ambiguous both in English and Arabic. Chomsky's [45] Binding principles account for sentences like this.

In addition to these levels of ambiguity the process of normalization plus features of Arabic such as the pro-drop structure language which means that the pronoun is dropped and it has a passive voice, complex word structure, lack of capitalization, and minimal punctuation contribute to ambiguity, but it is the absence of short vowels that contributes most significantly to ambiguity.

2.4.5 Applications and tools of ANLP

Over the last few years, Arabic natural language processing (ANLP) has gained increasing importance, and several state-of-the-art systems have been developed for a wide range of applications. These applications must deal with several complex problems related to the nature and structure of the Arabic language. For example, Arabic is written from right to left. Like Chinese and Japanese there is no capitalization in Arabic. In addition, Arabic letters change shape according to their position in the word. Modern Standard Arabic does not have orthographic representation of short letters which requires a high degree of homograph resolution and word sense disambiguation. Like Italian, Spanish, Chinese, and Japanese, Arabic is a pro-drop language, that is, it allows subject pronouns to drop [47] subject to recoverability of deletion. These applications include:

2.4.5.1 Machine translation

i. Google Translate: bidirectional translation for over 50 languages including Arabic. Application available at: http://translate.google.com/

ii. Microsoft's Bing Translator: bidirectional translation for over 30 languages including Arabic. Application available at: http://www.microsofttranslator.com/

iii. Sakhr's Tarjim (Arabic-English and English-Arabic). http://translate.sakhr.com/

iv. Almisbar Arabic-English translation. Application available at: http://www.almisbar.com/index.html

v. Statistical MT public resources: Giza alignment, Pharaoh and Moses decoders, etc. Application available at: http://statmt.org/

2.4.5.2 Information retrieval

Application available at: http://www.yamli.com/

2.4.5.3 Information extraction.

2.4.5.4 Speech synthesis and recognition.

2.4.5.5 Text to speech.

2.4.5.6 Tutoring systems.

2.4.5.7 Arabic named entity recognition

Application available at: http://qatsdemo.cloudapp.net/Farasa/

2.4.5.8 Morphological analyzers.

2.4.5.9 Sentiment analysis.

2.4.5.10 Lexicography

aConCorde: A concordance generation program for Arabic. Application available at: http://www.andy-roberts.net/software/aConCorde .

Most ANLP systems developed in the Western world focus on tools to enable non-Arabic speakers make sense of Arabic texts. are very useful to intelligence and security agencies. Because the need for such tools was critical, they were developed using machine learning approaches. Machine learning does not usually require deep linguistic knowledge and fast and inexpensive. These tools are characterized by their diversity in terms of development languages used, inputs/outputs manipulated, internal and external representations of results, etc. Developers of such tools had to deal with difficult issues. One problem is when Arabic texts include many translated and transliterated named entities whose spelling in general tends to be inconsistent in Arabic texts [37]. For example, a named entity such as the city of Washington could be spelled 'واشنجطن' ، 'واشنطن'، ''واشنطغن' ، 'وشنطن'. Another problem is the lack of a sizable corpus of Arabic-named entities which would have helped both in rule-based and statistical named entity

recognition systems. Efforts are being made to remedy this. A third limitation is that NLP tools developed for Western languages are not easily adaptable to Arabic due to the specific features of the Arabic language. Recognizing that developing tools for Arabic is vital for the progress in ANLP [48].

The Arabic digital world knows a continuous growth in terms of content (texts, videos, images etc.), this content has known a large growth. Thereby, the processing of this content requires the development of appropriate tools dedicated to ANLP. By using these different tools, the researcher in ANLP needs sometimes to call several ones at the same time in the same project, whether morphology, syntax or even semantic. In addition, most researchers do not take into account the software engineering principles while developing their tools, which affects the efficiency, productivity, flexibility, robustness, etc. This is due to the diversity in terms of inputs and outputs of the various tools. To remedy this, one solution consists of adjusting all inputs and outputs to meet the needs of this pipeline only, and this may not be reused for another pipeline in another context.

2.5 SEARCH ENGINES

2.5.1 Definition

Search Engine refers to a huge database of internet resources such as web pages, newsgroups, programs, images etc. It helps to locate information on World Wide Web. User can search for any information by passing query in form of keywords or phrase. It then searches for relevant information in its database and return to the user. [69] Search engines allow people to access online information on the Web. As such, search engines provide great benefits to individuals, organizations, and society.

2.5.2 Components of Search Engine

There are three basic components of a search engine as listed below:

- Web Crawler: It is also known as spider or bots. It is a software component that navigates the web to gather information.
- Database: All the information on the web is stored in database. It consists of huge web resources.
- Search Interfaces: This component is an interface between user and the database. It helps the user to search through the database. [71]

2.5.3 Architecture of Search Engine

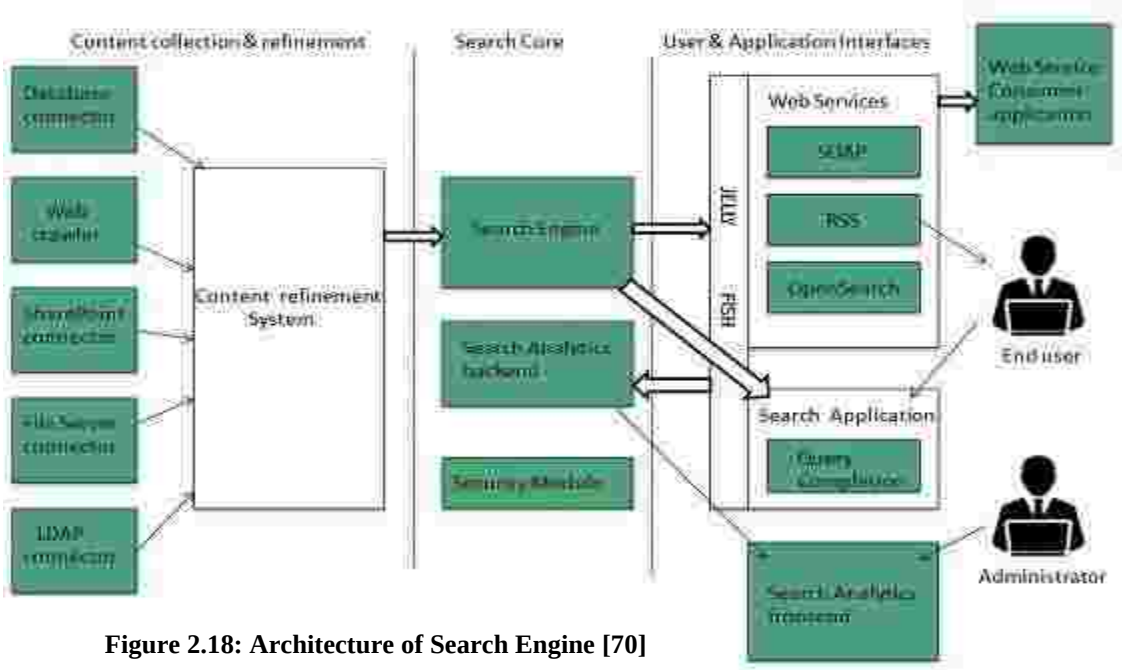

Figure 2.18: Architecture of Search Engine [70]

The search engine architecture comprises of the three basic layers listed below:

2.5.3.1 Content collection and refinement.

2.5.3.2 Search core.

2.5.3.3 User and application interfaces.

2.5.4 How Search Engines work

Web crawler, database and the search interface are the major component of a search engine that actually makes search engine to work. Search engines make use of Boolean expression AND, OR, NOT to restrict and widen the results of a search. Following are the steps that are performed by the search engine:

2.5.4.1 The search engine looks for the keyword in the index for predefined database instead of going directly to the web to search for the keyword.

2.5.4.2 It then uses web crawler to search for the information in the database.

2.5.4.3 Once web crawler finds the pages, the search engine then shows the related web pages as a result. These retrieved web pages generally include title of page, size of text portion, first several sentences etc.

2.5.4.4 User can click on any of the search results to open it.

These search criteria may vary from one search engine to the other. The retrieved information is ranked according to various factors such as frequency of keywords, relevancy of information, links etc.

2.5.5 Features of Search Engine

- A true search engine is an automated software program that moves around the Web collecting Web Pages to include in its catalog or database.
- It searches when a user requests information from a search engine; not the entire Web.
- Each search engine has its own catalog or database of collected Web Pages, so you will get different results/hits by using different search engines. **[70]**

2.5.6 Search Engine Indexing Process

Indexing process comprises of the following three tasks:

2.5.6.1 Text acquisition: It identifies and stores documents for indexing.

2.5.6.2 Text transformation: It transforms document into index terms or features.

2.5.6.3 Index creation: It takes index terms created by text transformations and create data structures to support fast searching.

2.5.7 Query Processing in search engines

Query process comprises of the following three tasks:

2.5.7.1 User interaction: It supports creation and refinement of user query and displays the results.

2.5.7.2 Ranking: It uses query and indexes to create ranked list of documents.

2.5.7.3 Evaluation: It monitors and measures the effectiveness and efficiency. It is done offline.

2.5.8 Search Engines that support Arabic

Arabic content is often not well indexed on the web, and suffers from poor rankings on Google and other standard search engines originally developed in the U.S. But businesspersons in the Arab world have been addressing the problem for several years now, creating specialized search engines to better serve the Arabic web user, such as Yamli, Eiktub and Yoolki. **[72]** [77] In Arabic-speaking cultures these are the major Search Engines:

i. Google

Google Search, commonly referred to as Google Web Search or simply Google, is a web search engine developed by Google. It is the most-used search engine on the World Wide Web, handling more than three billion searches each day. As of February 2016, it is the most used search engine in the US with 64.0% market share.

The order of search results returned by Google is based, in part, on a priority rank system called "PageRank". Google Search also provides many different options for customized search, using symbols to include, exclude, specify or require certain search behavior, and offers specialized interactive experiences, such as flight status and word definitions, and more. The main purpose of Google Search is to hunt for text in publicly accessible documents offered by web servers, as opposed to other data, such as images or data contained in databases. [73]

Google makes use of three important processes in order to fetch, filter and rank webpages, thus making sure no stone is left unturned in augmenting the relevancy of a web search. The three vital procedures that are part of a google search are:

- **Crawling:** Crawling is nothing but fetching of web pages depending on a search. The founders of google developed a software called spiders which crawl web pages based on keywords in the search and once the web pages are fetched the spiders then crawl the links on those pages. This process continues until all related pages and the links on them are dumped into an index database. The google spiders while crawling first search for keywords in the URL of a page and then checks for them in the title of a page. Once the webpages are crawled based on keywords, the spiders then look for synonyms of those keywords in the fetched webpages.
- **Indexing:** All the crawled documents in the index database are indexed using Google's unique technique for indexing. The process of indexing pages is simple; all the parsed documents are assigned significant document ids and instead of placing the picked documents in word order, the words are placed in document order which is elucidated in example 1.
- **Page Ranking:** The ranking of pages in google is based on an algorithm proposed by Larry Page and Sergey Brin, which is represented below

  $PR(A) = (1-d) + d\ (PR(T1)/C(T1) + ...+ PR(Tn)/C(Tn))$

  Where, PR(A) - Page Rank of page A

  PR(Ti) - Page Rank of pages Ti which link to page A

C(Ti) - number of outbound links on page Ti

d, θ - damping factor which can be set between 0 and 1, usually 0.85

- **The Architecture of Google**

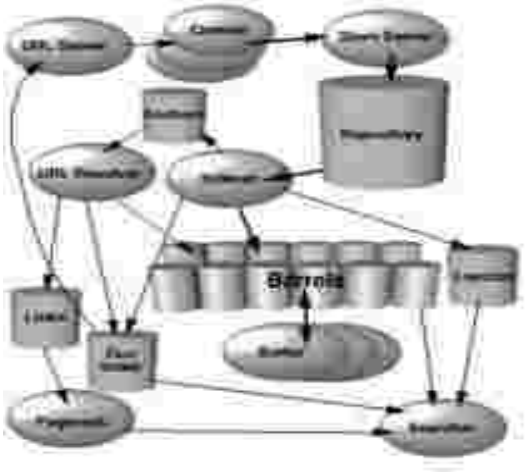

Figure 2.19: Architecture of Google Search Engine [73]

As seen in figure 2.18, the architecture of Google is simple. It consists of a URL server which sends all the URLs to the crawlers. The crawlers then crawl all the available URLs depending on keywords of the search, once the process of crawling is done; the fetched pages irrespective of relevancy are dumped into a store server. These pages are then compressed and saved in a repository. The indexer is then used to parse the pages. With the help of a sorter, the parsed pages are assigned doc ids and are sorted in various barrels accordingly. The doc ids of various pages are stored in the doc index file shown above.

All the non-textual information and links to the webpages are diverted into an anchors file. The URL resolver depicted in the above architecture then parses all the links and information on those links along with the data placed in barrels to pick the pages with relevant information. These parsed pages are then sent into a links file where actual page ranking is done by implementing the page rank algorithm. Once the page ranks are computed, the corresponding values and ranked pages are sent to the page rank file. The parsed and ranked pages ultimately reach the user or a searcher. [70]

ii. Bing

Bing is a web search engine owned and operated by Microsoft. The service has its origins in Microsoft's previous search engines: MSN Search, Windows Live Search and later Live Search. Bing provides a variety of search services, including web, video, image and map search products. It is developed using ASP.NET. As of November 2015, Bing is the second largest desktop search

engine in the US, with a query volume of 20.9%, behind Google on 63.9%. Yahoo! Search, which Bing largely powers, has 12.5%.

- **Performance Issues**

Bing has been criticized for being slower to index websites than Google. It has also been criticized for not indexing some websites at all. **[74]**

iii. Yahoo

Yahoo! Search is a web search engine owned by Yahoo. As of February 2015 it is the third largest search engine in the US by the query volume at 12.8%, after its competitors Google at 64.5% and Bing at 19.8%. Yahoo Search indexed and cached the common HTML page formats, as well as several of the more popular file-types, such as PDF, Excel spreadsheets, PowerPoint, Word documents, RSS/XML and plain text files. For some of these supported file-types, Yahoo Search provided cached links on their search results allowing for viewing of these file-types in standard HTML. Using the Advanced Search interface or Preferences settings, Yahoo Search allowed the customization of search results and enabling of certain settings such as: Safe Search, Language Selection, Number of results, Domain restrictions, etc. For a Basic and starter guide to Yahoo Search, they also provided a Search Basics tutorial. In the first week of May 2008, Yahoo launched a new search mash up called Yahoo Glue, which is in beta testing.

Yahoo Search provided the ability to search across numerous vertical properties outside just the Web at large. These included Images, Videos, Local, Shopping, Yahoo! Answers, Audio, directory, Jobs, News, Mobile, Travel and various other services as listed on their About Yahoo Search page. **[75]**

2.5.9 Arabic Search Engines

Below are three transliteration platforms allowing users to type Arabic without an Arabic keyboard converting words typed with Latin characters to their closest Arabic equivalent, in addition to a search enabler powered by Google:

2.5.9.1 Yamli: which means "he dictates", was launched in 2007, and offers two services. The first one is Arabic transliteration, and an Arabic search engine focused on providing more relevant search results for an Arabic query by expanding it to its most frequently used Latin equivalent.

2.5.9.2 Eiktub: which means "write", was launched in 2008, and also offers a transliteration service to type Arabic, and search in Arabic, through a Latin keyboard. It may not be in wide use anymore, but it also offers a pad that can be downloaded and used offline.

2.5.9.3 Yoolki: which means "to give or write a speech", was launched in 2007, and offers a transliteration and Arabic search service using a Latin keyboard. [76[

All of these Arabic search engines were created a good 6+ years ago to fill a gap in the search engine marketplace. At this point in time the main search engines (i.e. Google and Yahoo) as we know them today actually were not able to work with Arabic characters and therefore struggled to give Arabic-speaking cultures good search results. As time has moved on the investment in Arabic language search engines, and other major non-western characters languages such as Chinese, by the big international search engine has progressed hugely. This has left a 50/50 split in the use of Arabic search engines.

Indeed, if a strictly Arabic language search engine wanted to penetrate the marketplace now they would need to offer something that Google cannot offer. Considering Google's domination of the marketplace, and its continuous heavy investment in improving its search engine, it seems unlikely that a strictly Arabic language search engine would be able to set itself beyond Google's offerings. So, on the one hand, we have the increased use of Google Arabic search engine due to its continued use as a marketing tool and also due to its popularity amongst users. But on the other hand, we have the continued use of the strictly Arabic search engines such as Yamli which has gained and managed to keep its users from 6+ years ago. [78]

Part Two

Related Works

Now this part laid down the techniques, that can finally use in this project. As mentioned in the introduction chapter, this project aims at building a website that provides Arabic searching using different basic morphological shapes (Stem, Lemma and Name of entity), on the Wikipedia documents to enhance the search process. The optimal scenario is the following: User input the query, then the system will process the query using the techniques that will be discussed later. Then search about the query in the Wikipedia documents then provide the results for the user in different morphological shapes. Now we will talk about how to achieve that goal and scenario using computational linguistic tool & Search server. In this part, two techniques that use to build the project are discussed which they're: Farasa and Solr search server.

2. 1 Farasa

2.1.1 Definition

Farasa (which means "insight" in Arabic), is a fast and accurate text processing toolkit for Arabic text. [79] Segmenter takes a word and splits prefixes and suffixes:
Noun: wbktAbnA -> w+b+ ktAb+nA (and in our book).
Verb: fsynfqwnhA -> f+s+ynfq+wn+hA (so they will spend it).

Proper segmentation is critical for machine translation (MT) and Arabic Information Retrieval (IR). [79] Farasa consists of the segmentation/tokenization module, POS tagger, Arabic text Diacritizer, and Dependency Parser.

2.1.2 Procedure

Farasa segmentation/tokenization module is based on SVM-rank using linear kernels that uses a variety of features and lexicons to rank possible segmentations of a word. The features include: likelihoods of stems, prefixes, suffixes, their combinations; presence in lexicons containing valid stems or named entities; and underlying stem templates. [79]

- **Posed segmentation as a ranking problem (SVM^{Rank})**

  - Given word (without context), find all possible segmentation:
    - **Valid prefixes:** *f, w, l, b, k, Al, s*
    - **Valid suffixes:** *A, p, t, k, n, w, y, At, An, wn, wA, yn, kmA, km, kn, h, hA, hmA, hm, hn, nA, tmA, tm, tn*

- **Rank each possible segmentation based on the following features:**

  - Probability (*prefix| leading character sequence*)

- P (*suffix*| *trailing character sequence*)
- Language Model P(*stem*): LM trained on 12 years of Aljazeera newswire articles (94 million words).
- LM P (*stem+1st suffix*)
- P (*prefix*| *suffix*) & P (*suffix*| *prefix*)
- P (*stem template*) - template acquired from QATARA
- Is stem in Lexicon:
    - **Aljazeera corpus**
    - **Wikipedia gazetteer**
    - **AraComLex**
    - **Buckwalter stems**
- |$stem_{length}$ - average($stem_{length}$)|

2.1.3 Training and Testing

2.1.3.1 Training

- Arabic Penn Treebank (ATB) - parts 1, 2, and 3.
- 629k tokens (66k unique).

2.1.3.2 Testing

- 70 WikiNews articles (from 2013 and 2014).
    - Cover politics, economics, health, science and technologies, sports, art, and culture (10 per theme).
    - Contain 18,300 words.

2.1.3.3 Experimental setups

- $Farasa_{base}$: scores every word.
- $Farasa_{lookup}$: uses segmentations from training and scores unseen words.
- Compared to: MADAMIRA and QATARA.

Table 2.1: Comparison of Error rate between Farasa, Madamira and Qatara

System	Error Rate
MADAMIRA	1.24%
QATARA	1.77%
$Farasa_{base}$	1.24%
$Farasa_{lookup}$	**1.06%**

2.1.4 Why use Farasa

2.1.4.1 Speed

Speed test on 7.4 million words on i5 laptop:

$Farasa_{base}$: 129 sec

Qatara: 18 min

MADAMIRA: 2.5 hours

$Farasa_{lookup}$ can process one billion words in less than 5 hours

2.1.4.2 Integration

- 100% Java.
- Packaged into 1 jar file with no external dependencies.
- Lucene and Moses integration available.

2.1.4.3 Availability

Available one the internet: 100% free. Available at:
http://qatsdemo.cloudapp.net/farasa/register.html [80]

2.2 Apache Solr Search Server

2.2.1 Definition

Solr is an open-source search platform which is used to build **search applications**. It's a scalable, ready to deploy, search/storage engine optimized to search large volumes of text-centric data. It was built on top of **Lucene** (full text search engine). Solr is enterprise-ready, fast and highly scalable. The applications built using Solr are sophisticated and deliver high performance.

2.2.2 Brief History

Yonik Seely who created Solr in 2004 in order to add search capabilities to the company website of CNET Networks. In Jan 2006, it was made an open-source project under Apache Software Foundation. Solr can be used along with Hadoop. As Hadoop handles a large amount of

data, Solr helps us in finding the required information from such a large source. Not only search, Solr can also be used for storage purpose. Like other NoSQL databases, it is a **non-relational data storage** and **processing technology**.

2.2.3 Features of Apache Solr

Solr is a wraparound Lucene's Java API. Therefore, using Solr, you can leverage all the features of Lucene. Some of most prominent features of Solr:

2.2.3.1 Restful APIs: To communicate with Solr, it is not mandatory to have Java programming skills. Instead you can use restful services to communicate with it. We enter documents in Solr in file formats like XML, JSON and .CSV and get results in the same file formats.

2.2.3.2 Full text search: Solr provides all the capabilities needed for a full text search such as tokens, phrases, spell check, wildcard, and auto-complete.

2.2.3.3 Enterprise ready: According to the need of the organization, Solr can be deployed in any kind of systems (big or small) such as standalone, distributed, cloud, etc.

2.2.3.4 Flexible and Extensible − By extending the Java classes and configuring accordingly, we can customize the components of Solr easily.

2.2.3.5 NoSQL database − Solr can also be used as big data scale NOSQL database where we can distribute the search tasks along a cluster.

2.2.3.6 Admin Interface − Solr provides an easy-to-use, user friendly, feature powered, user interface, using which we can perform all the possible tasks such as manage logs, add, delete, update and search documents.

2.2.3.7 Highly Scalable − While using Solr with Hadoop, we can scale its capacity by adding replicas.

2.2.3.8 Text-Centric and Sorted by Relevance − Solr is mostly used to search text documents and the results are delivered according to the relevance with the user's query in order.

Unlike Lucene, you don't need to have Java programming skills while working with Apache Solr. It provides a wonderful ready-to-deploy service to build a search box featuring autocomplete, which Lucene doesn't provide. Using Solr, we can scale, distribute, and manage index, for large scale (Big Data) applications.

2.2.4 Lucene in Search Applications

Lucene is simple yet powerful Java-based search library. It can be used in any application to add search capability. Lucene is a scalable and high-performance library used to index and search virtually any kind of text. Lucene library provides the core operations which are required by any search application, such as **Indexing** and **Searching**. Lucene works as the heart of any search application and provides the vital operations pertaining to indexing and searching.

2.2.5 Solr Architecture - Building Blocks

The following illustration shows a block diagram of the architecture of Apache Solr.

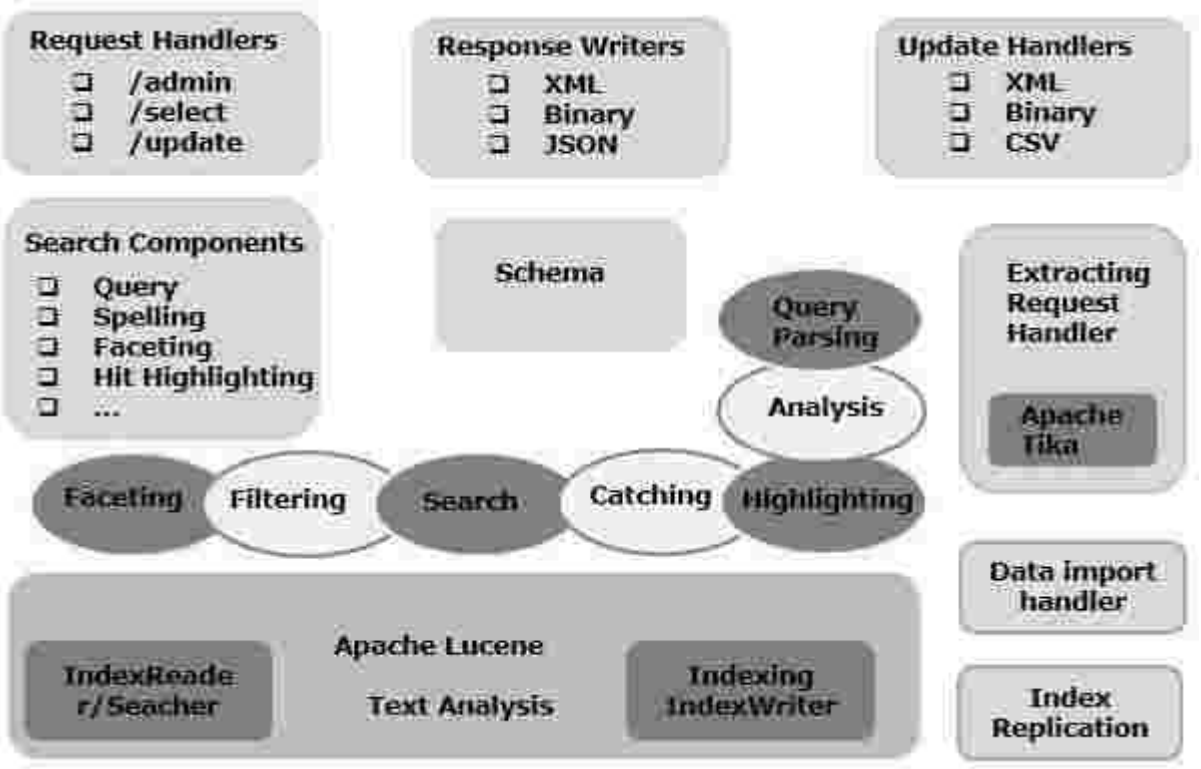

Figure 2.20: Solr Architecture – Building Blocks [81]

Following are the major building blocks (components) of Apache Solr:

2.2.5.1 Request Handler − The requests we send to Apache Solr are processed by these request handlers. The requests might be query requests or index update requests. Based on our requirement, we need to select the request handler. To pass a request to Solr, we will generally map the handler to a certain URI end-point and the specified request will be served by it.

2.2.5.2 Search Component − A search component is a type (feature) of search provided in Apache Solr. It might be spell checking, query, faceting, hit highlighting, etc.

These search components are registered as **search handlers**. Multiple components can be registered to a search handler.

2.2.5.3 Query Parser − The Apache Solr query parser parses the queries that we pass to Solr and verifies the queries for syntactical errors. After parsing the queries, it translates them to a format which Lucene understands.

2.2.5.4 Response Writer − A response writer in Apache Solr is the component which generates the formatted output for the user queries. Solr supports response formats such as XML, JSON, CSV, etc. We have different response writers for each type of response.

2.2.5.5 Analyzer/tokenizer − Lucene recognizes data in the form of tokens. Apache Solr analyzes the content, divides it into tokens, and passes these tokens to Lucene. An analyzer in Apache Solr examines the text of fields and generates a token stream. A tokenizer breaks the token stream prepared by the analyzer into tokens.

2.2.5.6 Update Request Processor − Whenever we send an update request to Apache Solr, the request is run through a set of plugins (signature, logging, indexing), collectively known as **update request processor**. This processor is responsible for modifications such as dropping a field, adding a field, etc.

2.2.6 Configuration Files

The main configuration files in Apache Solr are as follows:

2.2.6.1 Solr.xml − It is the file in the $SOLR_HOME directory that contains Solr Cloud related information. To load the cores, Solr refers to this file, which helps in identifying them.

2.2.6.2 Solrconfig.xml − This file contains the definitions and core-specific configurations related to request handling and response formatting, along with indexing, configuring, managing memory and making commits.

2.2.6.3 Schema.xml − This file contains the whole schema along with the fields and field types.

2.2.6.4 Core.properties − This file contains the configurations specific to the core. It is referred for **core discovery**, as it contains the name of the core and path of the data directory. It can be used in any directory, which will then be treated as the **core directory**. [81]

2.3 The Wikipedia

2.3.1 Definition

Wikipedia is a multilingual, web-based, free-content encyclopedia that is based on a model of openly editable content. It is the largest and most-popular general reference work on the Internet, and is named as one of the most popular websites. It is owned and supported by the Wikimedia Foundation, a non-profit organization which operates on whatever money it receives from its annual fund drives. [84]

2.3.2 Brief History

Wikipedia was launched on January 15, 2001 by Jimmy Wales and Larry Sanger. as a single English-language edition at www.wikipedia.com, and announced by Sanger on the Nupedia mailing list. Sanger coined its name, a portmanteau of wiki and encyclopedia. There was only the English-language version initially, but similar versions in other languages quickly developed which differ in content and in editing practices. With 5,655,690 articles, the English Wikipedia is the largest of the more than 290 Wikipedia encyclopedias. The English Wikipedia passed the mark of two million articles on September 9, 2007, making it the largest encyclopedia ever assembled. Overall, Wikipedia comprises more than 40 million articles in 301 different languages and had 18 billion page views and nearly 500 million unique visitors each month as of February 2014. By the end of December 2016, Wikipedia was ranked fifth in the most popular websites globally. As of March 2017, Wikipedia has about 40,000 Featured Articles and Good Articles that cover vital topics. **[84]**

2.3.3 Why Using Wikipedia

As mentioned in the definition, Wikipedia is a huge and free encyclopedia. So it includes millions of documents. Because of that, the Wikipedia is suitable for build and test the search engine on. Also the Wikimedia organization offers a complete copy of Wikipedia's database (documents) in a lot of languages. It's available for free downloading in format of XML or SQL from: https://dumps.wikimedia.org

2.4 Search Engine Indexing

2.4.1 Introduction

Search engine indexing collects, parses, and stores data to facilitate fast and accurate information retrieval. Index design incorporates interdisciplinary concepts from linguistics, cognitive psychology, mathematics, informatics, and computer science. An alternate name

for the process in the context of search engines designed to find web pages on the Internet is web indexing. Popular engines focus on the full-text indexing of online, natural language documents. Meta search engines reuse the indices of other services and do not store a local index, whereas cache-based search engines permanently store the index along with the corpus. Unlike full-text indices, partial-text services restrict the depth indexed to reduce index size. Larger services typically perform indexing at a predetermined time interval due to the required time and processing costs, while agent-based search engines index in real time.

2.4.2 Indexing Purpose

The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power. For example, while an index of 10,000 documents can be queried within milliseconds, a sequential scan of every word in 10,000 large documents could take hours. The additional computer storage required to store the index, as well as the considerable increase in the time required for an update to take place, are traded off for the time saved during information retrieval.

2.4.3 Index design factors

Major factors in designing a search engine's architecture include:

1) Merge factors: How data enters the index, or how words or subject features are added to the index during text corpus traversal, and whether multiple indexers can work asy nchronously. The indexer must first check whether it is updating old content or addin g new content.

2) Storage techniques: How to store the index data, that is, whether information should be data compressed or filtered.

3) Lookup speed: The speed of finding an entry in a data structure, compared with how quickly it can be updated or removed, is a central focus of computer science. Mainte nance How the index is maintained over time.

4) Fault tolerance How important it is for the service to be reliable. Issues include dealing with index corruption, determining whether bad data can be treated in isolation, dealing with bad hardware, partitioning, and schemes such as hash-based or composite partitioning, as well as replication.

2.4.4 Challenges in parallelism

A major challenge in the design of search engines is the management of serial computing processes. There are many opportunities for race conditions and coherent faults. For example, a new document is added to the corpus and the index must be updated, but the index simultaneously needs to continue responding to search queries. This is a collision between two competing tasks. Consider that authors are producers of information, and a web crawler is the consumer of this information, grabbing the text and storing it in a cache (or corpus). The forward index is the consumer of the information produced by the corpus, and the inverted index is the consumer of information produced by the forward index. This is commonly referred to as a producer-consumer model. The indexer is the producer of searchable information and users are the consumers that need to search. The challenge is magnified when working with distributed storage and distributed processing. In an effort to scale with larger amounts of indexed information, the search engine's architecture may involve distributed computing, where the search engine consists of several machines operating in unison. This increases the possibilities for incoherency and makes it more difficult to maintain a fully synchronized, distributed, parallel architecture.

2.4.5 Index data structures

Search engine architectures vary in the way indexing is performed and in methods of index storage to meet the various design factors. Suffix tree Figuratively structured like a tree, supports linear time lookup. Built by storing the suffixes of words. The suffix tree is a type of trie. Tries support extendable hashing, which is important for search engine indexing. A major drawback is that storing a word in the tree may require space beyond that required to store the word itself. An alternate representation is a suffix array, which is considered to require less virtual memory and supports data compression. Inverted index Stores a list of occurrences of each atomic search criterion, typically in the form of a hash table or binary tree.

2.4.6 Inverted Index

In computer science, an inverted index (also referred to as postings file or inverted file) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents (named in contrast to a forward index, which maps from documents to content). The purpose of an inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database. The inverted file may be the database file itself, rather than its

index. It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines**.**

There are two main variants of inverted indexes: A record-level inverted index (or inverted file index or just inverted file) contains a list of references to documents for each word. A word-level inverted index (or full inverted index or inverted list) additionally contains the positions of each word within a document. The latter form offers more functionality (like phrase searches), but needs more processing power and space to be created.

2.4.7 Applications of Inverted Index

The inverted index data structure is a central component of a typical search engine indexing algorithm. A goal of a search engine implementation is to optimize the speed of the query: find the documents where word X occurs. Once a forward index is developed, which stores lists of words per document, it is next inverted to develop an inverted index. Querying the forward index would require sequential iteration through each document and to each word to verify a matching document. The time, memory, and processing resources to perform such a query are not always technically realistic. Instead of listing the words per document in the forward index, the inverted index data structure is developed which lists the documents per word. With the inverted index created, the query can now be resolved by jumping to the word ID (via random access) in the inverted index.

Chapter Three

Analysis

This chapter describes the functions and the requirements of the system. And it illustrates the structure and the feasibility of the system by the models created within it such as class diagram and use case diagram. This chapter is organized as following: section 1 identifies the feasibility study, Section 2 identifies the data gathering, Section 3 identifies the system requirements and Section 4 describe the modeling.

3.1 Feasibility Study

The first step of analysis that used to estimate the user needs may by satisfied using current software and hardware technologies. The purpose of feasibility study is to define the potential solutions to the problem.

3.1.1 Technical Feasibility

The technical feasibility is divided into steps. In this project there are very basic requirements in order to finish it.

3.1.1.1 Familiarity with the application: The application is a new idea but the design and the concepts are based on some of the applications which are currently popular in the Web. The examples are Google, dogpile, Duck Duck GO, Yoolki and Yorwa.

3.1.1.2 Familiarity with the technology: The application will have some new technological requirements which may create some technical risk according to integration between these components. This application will be using easy and popular technology such as the usage of the programming languages are PHP, java script and CSS. In addition to use new technologies such as Apache Solr 7 search engine, Solarium and Farasa Linguistic tool.

3.1.1.3 Project Size: The project size of this application is estimated around 7 months. This is a demanding and hard application to build, but doesn't have high risk factor. The project can be finished in the proposed time limit.

3.1.1.4 Compatibility: As this application is web based and no working will be done using old technology thus the compatibility of the proposed project is highly reliable.

3.1.2 Legal Feasibility

As the web would be based in the Arabic world (though the servers where the software will be executed and utilized by the consumer may be located elsewhere) there are number of laws which will have to be followed to. There is no database for this system and no functionality of user's

loggings or ids, hence the Data Protection Act 1998, is not applied to it. Additionally, as services will be used for free through the websites.

3.2 Data Gathering

There are different ways to get the data, such as research, papers, previous studies, the Internet and Questionnaire.

3.2.1 The Internet

There is a lot of technical problems that everybody needs to get them solved, so the best way is the Internet and its different websites. So during the huge amount of information we learned what the Natural Language Processing and how it processes the Arabic Languages. We discover new things about the wonder of the Arabic Language and its concepts. Also we learned how to use different techniques, such as Apache Solr, Farasa, Solarium, PHP/JAVA bridge and so on. Also we understand the structure of web and search operation.

3.2.2 The Previous Documents

The previous document that available on the Internet is main source that we used to get the information and requirements about the application that we decide to build. They helped us in the literature review and understand the main concepts of the techniques that we used.

3.2.3 The Questionnaire

The main purpose of using the questionnaire in our project is, attempt to understand some of the problems that people phase when they search in Arabic Language and what are the features that they want to add them to the search engine (Customization). The Questionnaire Sample is provided in Appendix A.

3.3 The Requirement Specification

The requirements are description of features, services and functionalities that software must provide.

3.3.1 User Requirements

3.3.1.1 Functional Requirements

R1. The system must be able to provide the user with the ability to find any type of Arabic Wikipedia documents.

R2. The system shall provide the user with the ability to write search text (query) in Arabic to find results.

R3. The system shall prevent writing in any other languages except Arabic.

R4. If the query is valid the system must be able to manipulate the query linguistically.

R5. The system must search of manipulated query results in indexed Wikipedia documents.

R6. The system must be able to show the most related results at first.

R7. The system must provide the user with the ability to navigate to selected result page.

R8. The system must provide the user with the help about how to use the system.

R9. The system must provide the user with the ability to customize search engine theme (Theme color, font color, and background) from list.

R10. The system must contain support, with contact information and feedback.

R11. The system must be suitable to open it in any device.

3.3.1.2 Non-Functional Requirements

The following section show non-functional requirement the system must materialize as categorized in ISO/IEC 9126 standard.

I. Functionality

As in ISO/IEC 9126 standard the functionality contains the following sub categories that the system must materialize.

a. Accuracy

The system must provide accurate results, where it doesn't display any result that not contain the keyword.

b. Interoperability

The system must be able to send and receive data from solr and Farasa to get correct result.

II. Usability

As in ISO/IEC 9126 standard the Usability contain the following sub categories that the system must materialize.

a. Understandability

The system must provide simple interfaces that contain clear, self-descriptive and meaningful icons and buttons and the help will found in all page's menu.

b. Learnability

The system is Understandable as mentioned above so any category age (adult –kids) can learn it after 5 minutes from use it.

c. Attractiveness

The system must provide interfaces that contain the following:

- Suitable color for everyone that can be customized.
- Every page has one task.
- Text and descriptions is short, clear and concise.
- The instruction that help to use the search engine.

3.3.2 System Requirements

3.3.2.1 Functional Requirements

R1. The system must be able to provide the user with the ability to find any Arabic Wikipedia documents.

1.1 The system must index all Wikipedia document as following:

a) Solr search engine must index all XML documents that Wikimedia offers at Wikimedia dumps.

b) Solr search engine must index all the documents and store them in an inverted index.

R2. The system shall provide the user with the ability to write search text (query) in Arabic to find results

2. 1 The system must provide the user with the ability to write in search box in the main page.

R3. When the search button is selected, the system shall prevent writing in any other languages except Arabic.

3.1 The system must validate the query if it is not containing Arabic letters (another language letter or special character) system do the following:

3.1.1 The system prevents writing any other language letter or any chareter until the user enter Arabic letters.

3.1.2 The system shows massage "Only Arabic letters","أحرف عربية فقط" until the user enter Arabic letters.

R4. If the query is valid the system must be able to manipulate the query linguistically.

4.1 The system must be able to send the query to Farasa packages and solr search server to manipulate it linguistically.

4.2 The system shall be able to receive manipulated query from Farasa package.

R5. The system must search of manipulated query results in indexed Wikipedia documents.

5.1 The system must send the query to solr.

5.2 The system must send the manipulated query from Farasa to solr in advance search.

5.3 The solr must search the results.

5.4 The solr ranking the results.

5.5 System must be able to receive data(results) from solr.

R6. The system must be able to show the most related results at first.

6.1 The system must show ranking results in result page.

6.2 The system must show most related results at the paging pagination.

6.3 The system must show 10 results in one pagination as maximum.

R7. The system must provide the user with the ability to navigate to selected result page.

7.1 The system must show the result Title as a link to navigate to selected one.

7.2 The color of visited link must be changed to another color.

R8. The system must provide the user with the help about how to use the system.

8.1 The system must contain help instructions to use the system in menu.

R9. The system must provide the user with the ability to customize search engine theme.

9.1 The system must provide the user with the ability to change the following:

- **a.** Theme color.
- **b.** The background image.
- **c.** The color of the font.

By selecting them from setting in the menu.

R10. The system must contain support, that contains contact information and feedback.

10.1 The system must contain contact section in the menu that include Facebook account, Twitter account and the Email for any feedback.

R11. The system must be suitable to open it in any device.

11.1 The system pages must be responsive pages by using Boot Strap.

3.3.2.2 Non-Functional Requirements

The following section show non-functional requirement the system must materialize as categorized in ISO/IEC 9126 standard.

i. Functionality

As in ISO/IEC 9126 standard the functionality contains the following sub categories that the system must materialize.

a) Accuracy

The system must provide accurate results as following:

1. The system Doesn't display any result that not contain any query words.
2. The top results must contain identical query word.
3. The results in end pages in pagination contain just one query word.

ii. Reliability

As in ISO/IEC 9126 standard the Reliability contain the following sub categories that the system must materialize.

a) Fault tolerance

The system must be able to maintain specific level of performance by avoiding any expected fails by exception handling.

b) Availability

The system must be available 24/12/365 (24 days, 12 months in all year 365 days) by using redundancy of the system that will work auto if any error happens in the main system until it will be fixed.

iii. Usability

As in ISO/IEC 9126 standard the Usability contain the following sub categories that the system must materialize.

a) Understandability

The system must provide simple interfaces that contain clear, self-descriptive and meaningful icons and buttons, the instructions help will found in all page's menu and the system contain few pages which has unique functionality and tasks.

b) Attractiveness

The system must provide interfaces that contain the following

- Site map.
- The tasks categorized in site pages.

iv. Efficiency

As in ISO/IEC 9126 standard the Efficiency contain the following sub categories that the system must materialize.

a) Time behavior (response time)

- The system must provide search results in less than 15 second as maximum in good network quality.
- The system must navigate to selected results in less than 7 second as maximum in good network quality.

v. Maintainability

As in ISO/IEC 9126 standard the Efficiency contain the following sub categories that the system must materialize.

a) Testability

The system must traceable, good modular, and contain less coupling and high cohesion to facilitate test.

3.4 Modeling

3.4.1 UML Use Case Diagram

Use cases are a type of textual requirements specification that capture how a user will interact with a solution to achieve a specific goal. This system USE CASE shown in figure 3.1.

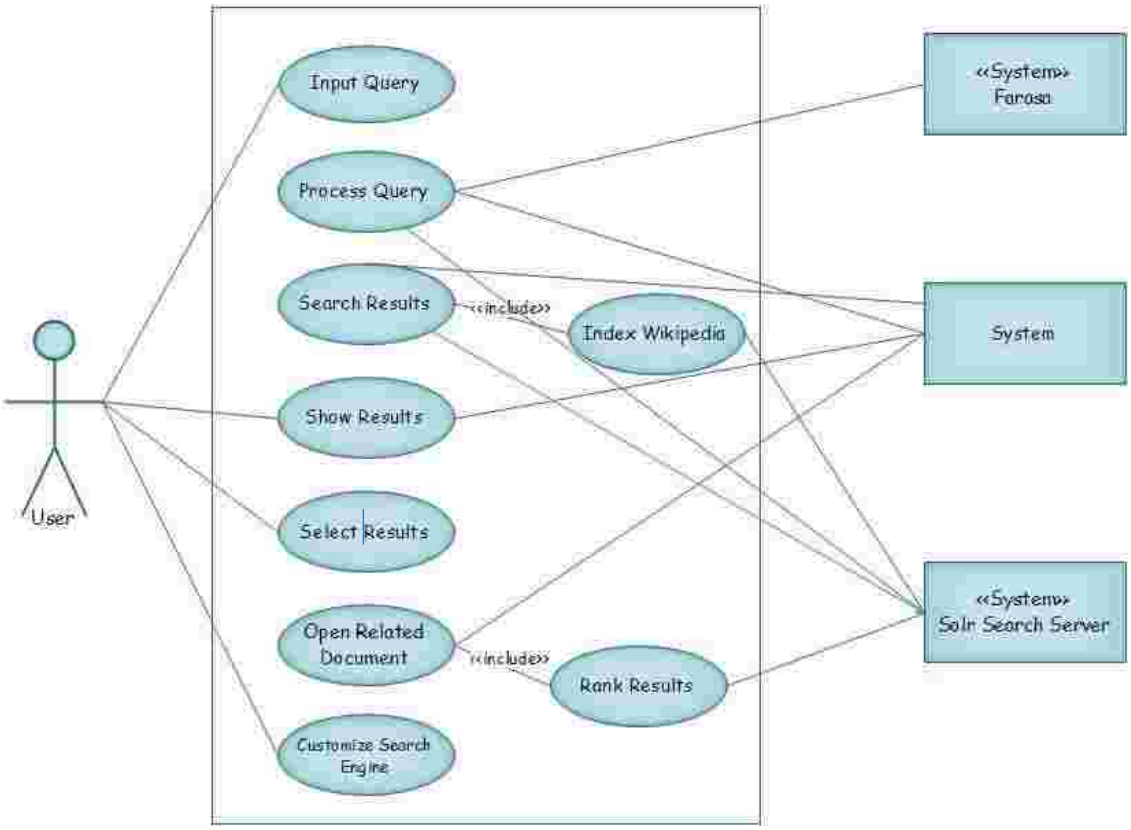

Figure 3.1: Use Case Diagram

3.4.2 Use Case Description

Use case descriptions order in table 3.1 and they listed from table 3.2-3.6.

Table 3.1: Use Case Description table order

Use Case Number	Use Case Name
1	Input Query
2	Process Query
3	Search Results
4	Show Results
5	Select Results
6	Customize Search Engine

Table 3.2: Input Query Use Case Description

Use Case Name: Input Query	ID: 1	Importance Level: High
Primary Actor: User	Use Case Type: Detail, Essential	
Stakeholders and Interests: User - want to search and get results.		
Brief Description: This use case describes how the system validate user query.		
Pre-condition: user open the website.		
Post-condition: -----		
Trigger: user Type the Query in the Search box. Type: External		
Relationships: Association: user. Include: Extend: Generalization:		
Normal Flow of Events: 1- User open the website, if the website can't open [E1] is be performed. 2- User inputs the search query; in the search box. it can be only Arabic text. If user types valid query, then Query processing use case is performed. And If user types in other language or special characters [S1] is performed.		
Sub Flows: [S1] The system will validate the input, if it isn't valid then [E2], is performed.		
Alternate/Exceptional Flows: [E1] Weak network: 1e1. User refreshes the website. 1e2. User quits the website. [E2] Query in another language: 2e1. The system will show alert "أحرف عربية فقط", "Arabic letters only" in the interface and prevent typing, until the user types only Arabic text		

Table 3.3: Process Query Use Case Description

Use Case Name: Process Query.	ID: 2	Importance Level: High
Primary Actor: Farasa, Solr, System.	Use Case Type: Detail, Essential	
Stakeholders and Interests: Farasa - it's a linguistic tool that used to process the query (especially, the Lemma, name of entity). Solr - is a Search Engine used to index Wikipedia documents and search about the query in these documents. It also can process the query (especially the Stem). System - sends the valid query as input to Farasa and Solr.		
Brief Description: This use case describes how the Farasa linguistic tool & Solr process the query.		
Pre-condition: The input Query is valid.		
Post-condition: The query will be processed (linguistic manipulation) successfully.		
Trigger: System finds the query valid and sends it to Farasa & Solr. Type: internal.		
Relationships: Association: Farasa, Solr, System. Include: Extend: Generalization:		
Normal Flow of Events: 1- System sends the valid query to Solr, then Solr processes the query [S1]. 2- Farasa linguistic tool processes the query (extracts the lemma for each word, recognized the NOE), when search by lemma is choosing. If Farasa is not sent any result [E1] is performed. 3- Farasa returns linguistic manipulation query to system. 4- System sends linguistic manipulation query to solr.		
Sub Flows: [S1] When result page is appeared the user can choose search by lemma.		
Alternate/Exceptional Flows: [E1] Valid query is not sent after 60 second (Farasa don't send any result). 1e1. The system resend query to Farasa.		

Table 3.4: Search Results Use Case Description

Use Case Name: Search Results	ID: 3	Importance Level: High
Primary Actor: System and Solr Search Engine	Use Case Type: Detail, Essential	
Stakeholders and Interests: System - sends queries to solr. Solr -is Search Engine used to index Wikipedia documents and search about the query in these documents.		
Brief Description: This use case describes how the Solr Search Engine Search Results.		
Pre-condition: Query Processed by Farasa.		
Post-condition: Solr return results.		
Trigger: system sends queries to solr. Type: internal.		
Relationships: Association: Solr, System. Include: Extend: Generalization:		
Normal Flow of Events: 1- The system sends queries to solr. 2- Solr analysis the queries (tokenization, stop word filtering and light stemming) 3- The solr search the query in Wikipedia documents. 4- The solr sends results as JSON file to system. if there is no result available [S1] is performed.		
Sub Flows: [S1] the solr search engine return 0 value and the system will show This massage " لا توجد نتائج مطابقة","not result found".		
Alternate/Exceptional Flows: ----		

Table 3.5: Show Results Use Case Description

Use Case Name: Show Results	ID: 4	Importance Level: High
Primary Actor: System.	Use Case Type: Detail, Essential	
Stakeholders and Interests: System – show the results of search in result (HTML) page after ranking them. User – Browse Results.		
Brief Description: This use case describes how the systems show Search Results.		
Pre-condition: Solr search about the query in indexed documents and return results.		
Post-condition: system successfully Show all ranking results in result (HTML) pages, with pagination.		
Trigger: Solr sends the result to system Type: internal.		
Relationships: Association: System, User. Include: Extend: Generalization:		
Normal Flow of Events: 1- The system will receive the results from Solr and show them for the user in html format using a friendly user interface. 2- The user will browse these results.		
Sub Flow: ----		
Alternate/Exceptional Flows: ----		

Table 3.6: Select Results Use Case Description

Use Case Name: Select Results	ID: 5	Importance Level: medium
Primary Actor: User.	Use Case Type: Detail, Essential	
Stakeholders and Interests: User – select the best matching result.		
Brief Description: This use case describes how the user select Results.		
Pre-condition: System show all ranking result in result (HTML) pages.		
Post-condition: Navigate to related document that is selected successfully.		
Trigger: User select any result. Type: External.		
Relationships: Association: User. Include: open related documents use case. Extend: Generalization:		
Normal Flow of Events: 1- User select the link of the best matching result. 2- system will Navigate the user to the related documents in the Wikipedia pages in new tap.		
Sub Flow:		
Alternate/Exceptional Flows: ----		

Table 3.7: Customize the Search Engine Use Case Description

Use Case Name: Customize Search Engine	ID: 6	Importance Level: medium
Primary Actor: User.	Use Case Type: Detail, Essential	
Stakeholders and Interests: User – will justify the setting of the Search Engine appearance as he prefers.		
Brief Description: This use case describes how the user customize the Search Engine appearance.		
Pre-condition: User select the setting option from the menu.		
Post-condition: Change the appearance of the Search engine.		
Trigger: User select setting from the menu. Type: External.		
Relationships: Association: User. Include: Extend: Generalization:		
Normal Flow of Events: 1- User selects the setting option from the menu. 2- The setting has three option: change font color, change theme color or change background image. 3- If the user chooses change font color, then [S1] will be performed. 4- If the user chooses change theme color, then [S2] will be performed. 5- If the user chooses change background image, then [S3] will be performed.		
Sub Flow: [S1] The font color of the page will change except the menu font color. [S2] The color of the page will change, including the menu and the background. [S3] The background will change with the image that the user chooses from the list or display it from his PC. Here the user also can clear the image if he doesn't want it.		
Alternate/Exceptional Flows: ----		

3.4.3 Class Diagram (Domain Model)

That describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. This system class diagram shown in figure 3.2.

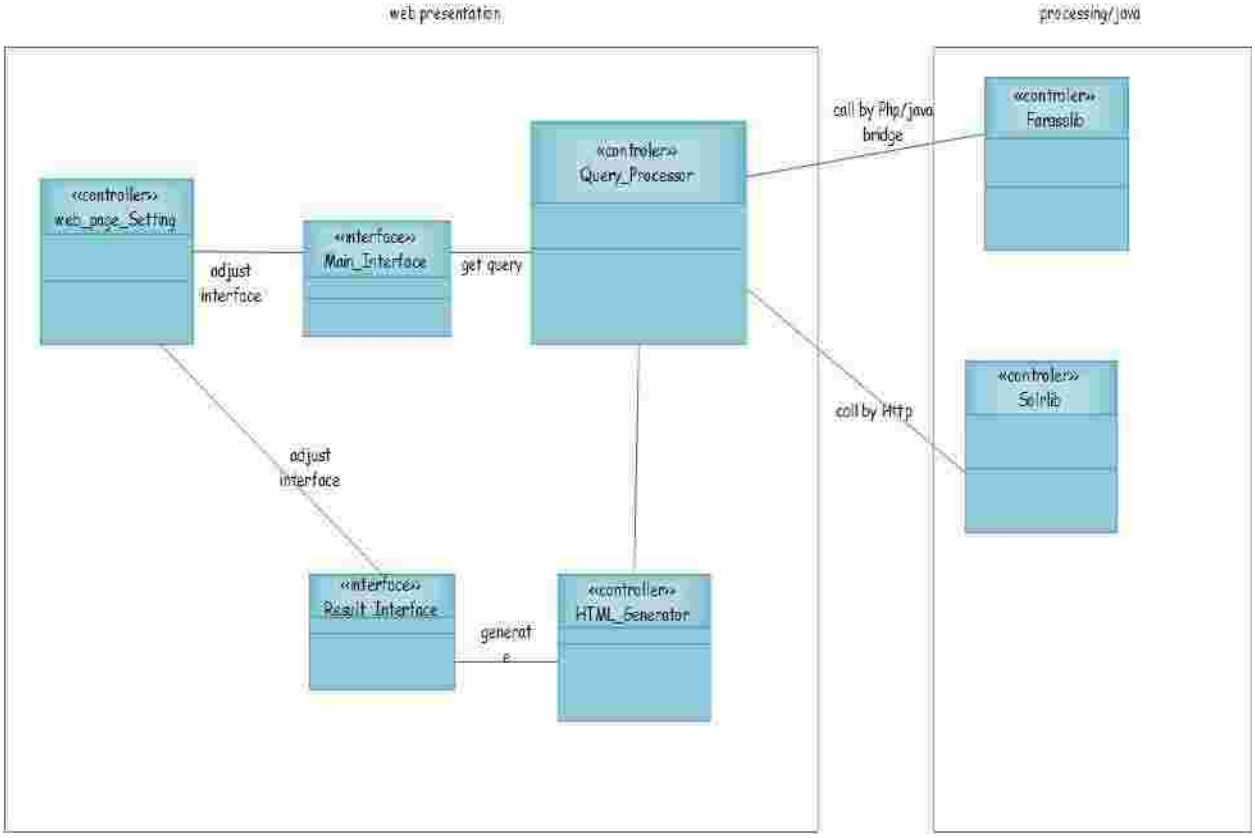

Figure 3.2: Class Diagram (Domain Model)

3.4.4 Activity Diagram

Activity diagrams portray the primary activities and the relationships among the activities in a process. In addition, it used to model the behavior in a business process independent of objects. Figure 3.3 presents an activity diagram that presents the complete search process.

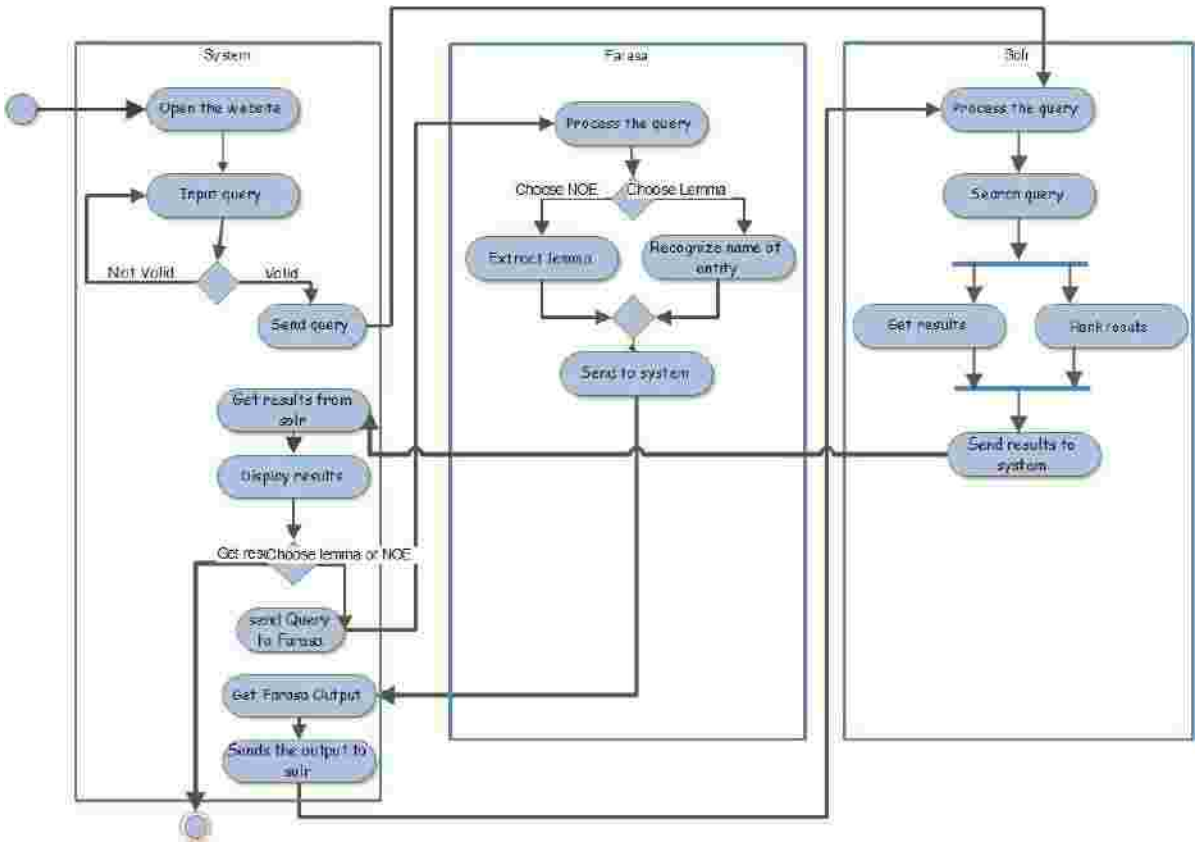

Figure 3.3: Activity Diagram

Chapter Four

Design

The analysis stage shows and lists the functions of the system to the stakeholders in a high level of abstraction using some diagrams, (e.g. Use Case diagram). Whereas the design stage shows and details how these functions can be met by the system and provides a clear vision for the developers & programmers. Design stage includes the architecture design, class diagram, sequence diagram, communication diagram and state chart diagram. So, design stage is no less importance of analysis stage too.

4. 1 Architecture Design

The architecture that used is layer architecture. It consists of three layers: presentation layer that include the User interface, then the processing layer that processes the data. Using Farasa, solr search engine and HTML generator. Finally, the data layer that includes the Wikipedia index.

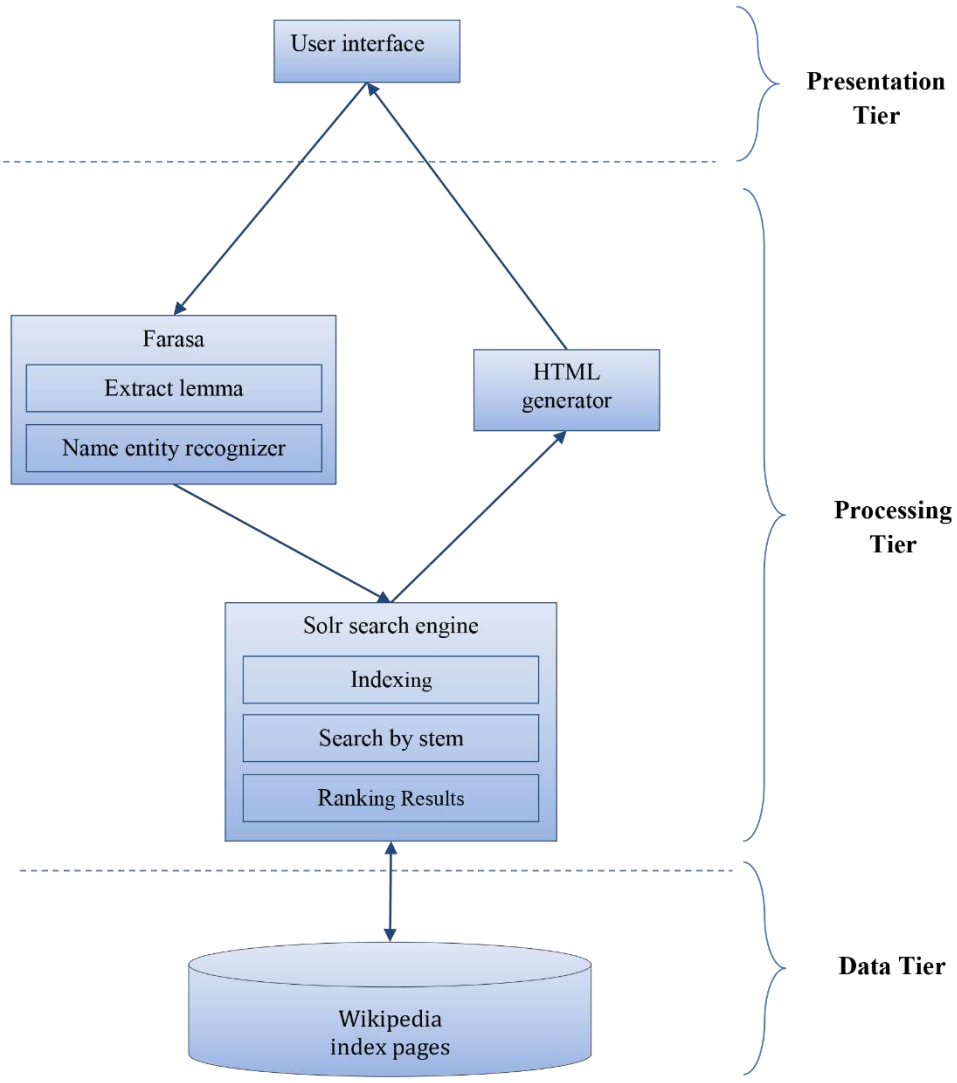

Figure 4.1: Architecture Design

4. 2 **Class Diagram** (Design Model)

Class diagram clearly map out the structure of the system by modeling its classes, attributes, operations, and relationships between objects and give general overview of the system, as shown in figure 4.2

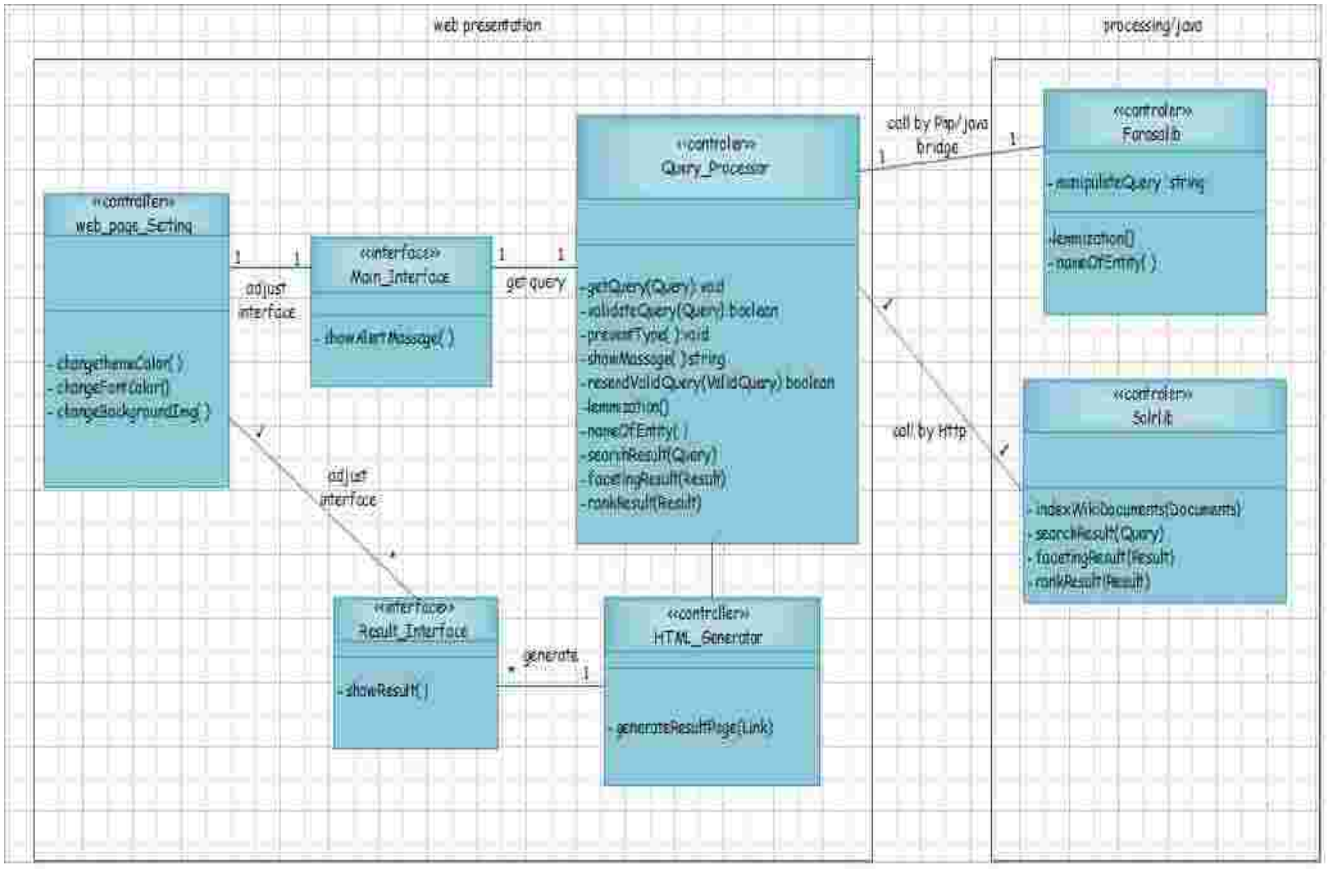

Figure 4.2: Class Diagram (Design Model)

4. 3 Sequence Diagrams

These diagrams show how the sequence of the operations and how each process gets affected by the other. Figure 4.3 illustrates the overall search process.

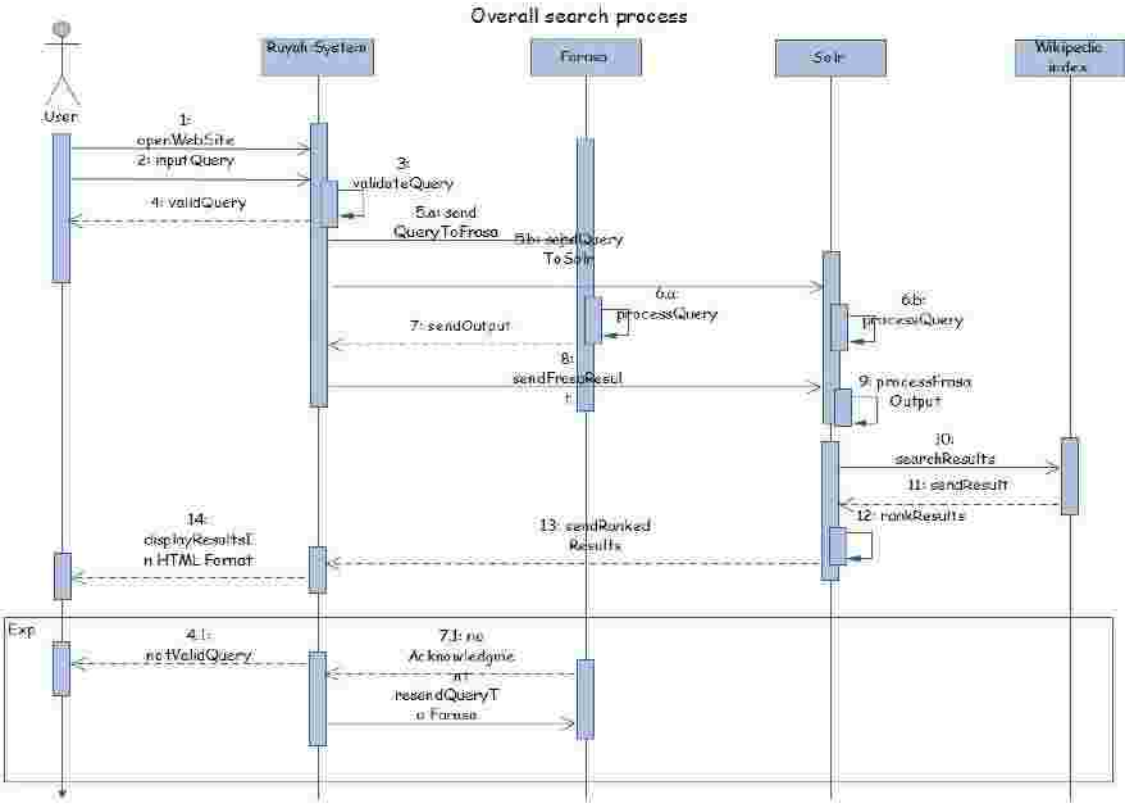

Figure 4.3: Sequence Diagram for the overall search process

Figure 4.4 illustrates the process of the input the query to the search engine.

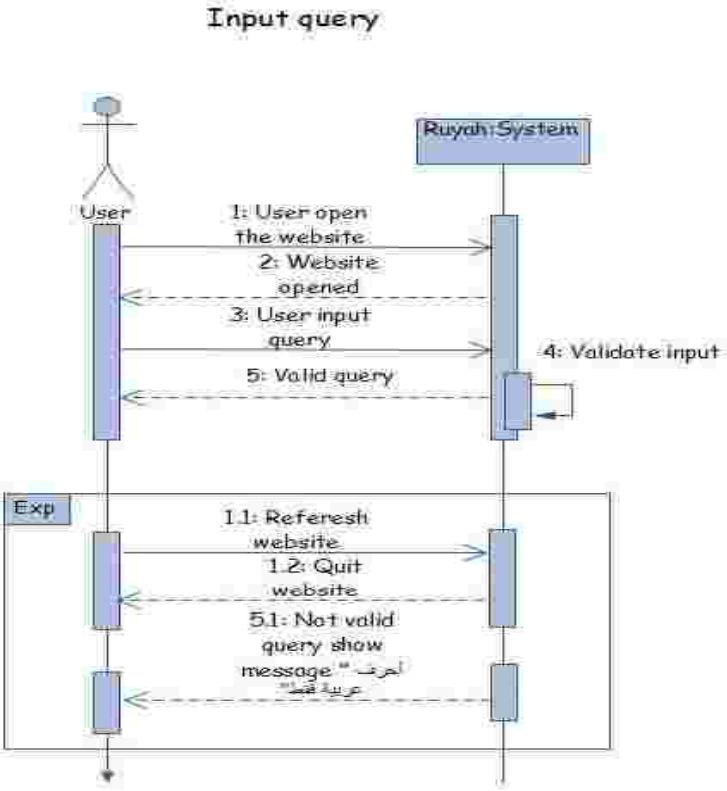

Figure 4.4: Input Query Sequence Diagram

Figure 4.5 illustrates the process the query of the user.

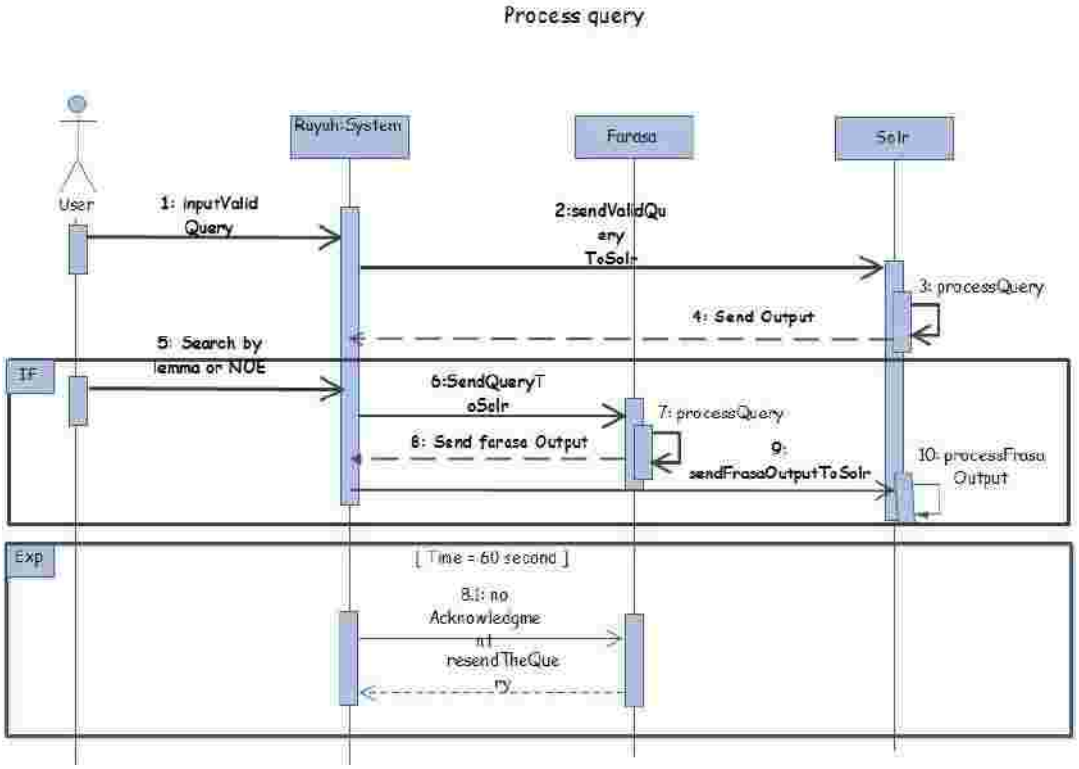

Figure 4.5: process Query Sequence Diagram

Figure 4.6 illustrates the process of the search in the Wikipedia documents and brings results to the website.

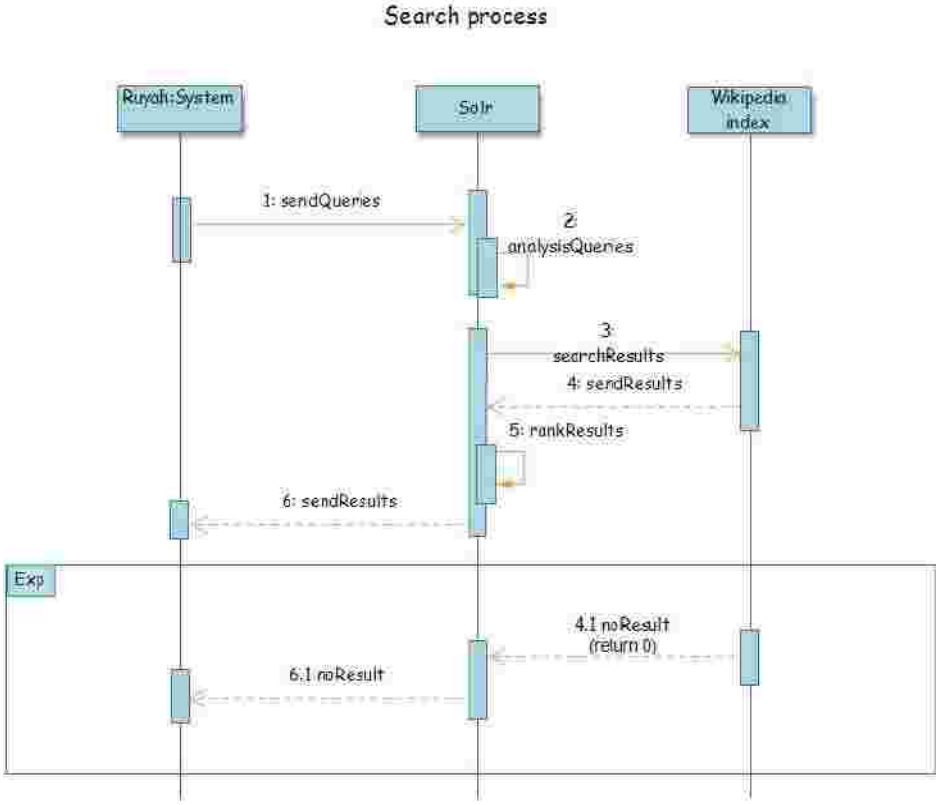

Figure 4.6: Search results Sequence Diagram

Figure 4.7 illustrates the process of the select and browse the results.

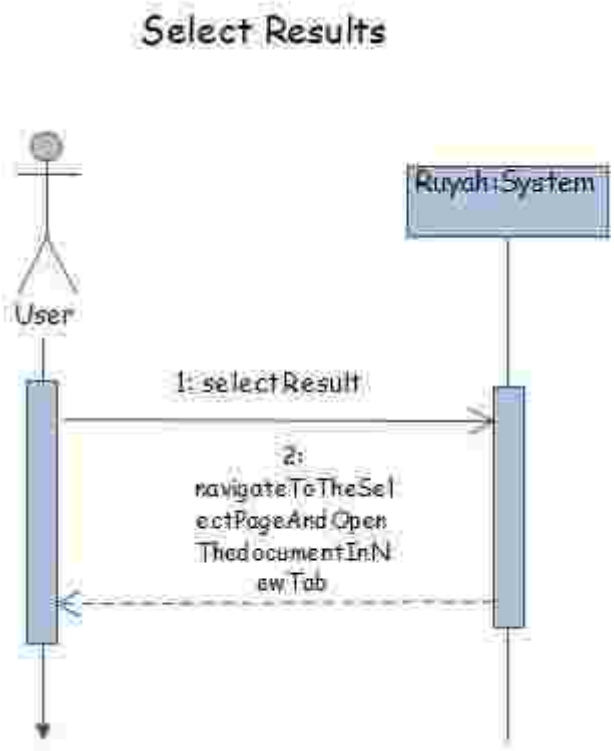

Figure 4.7: Select results Sequence Diagram

Figure 4.8 illustrates how the system shows the results.

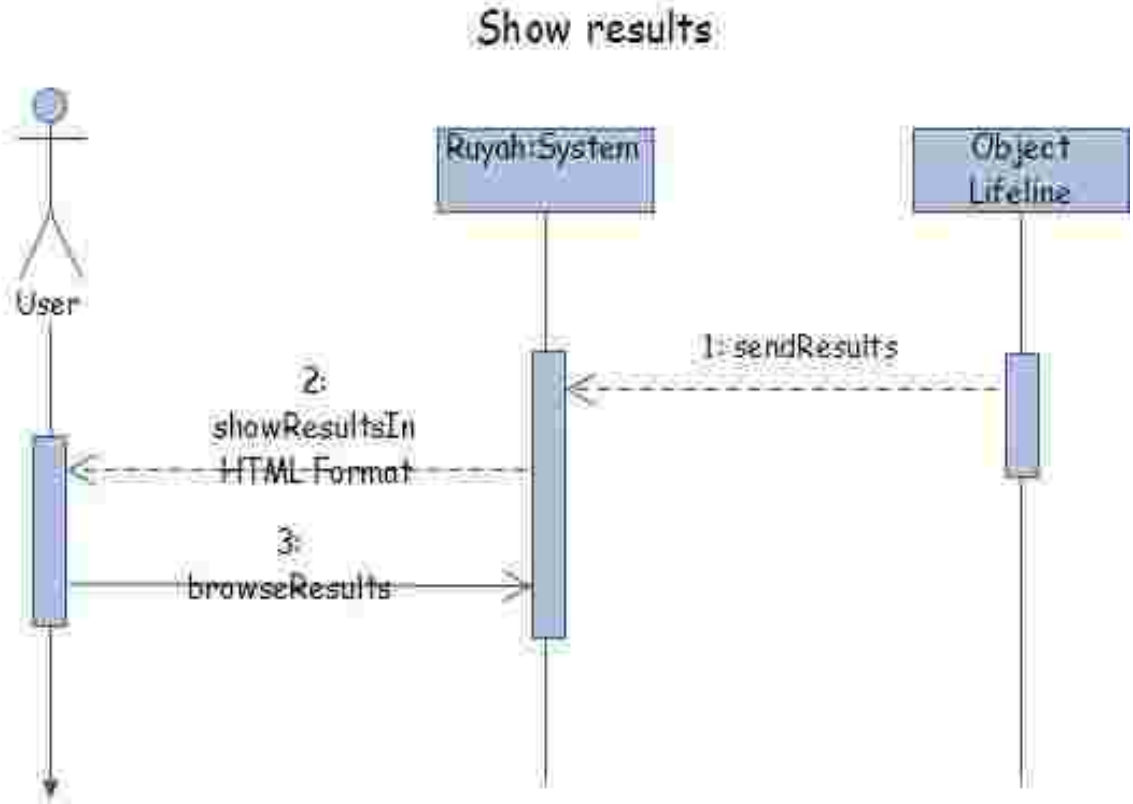

Figure 4.8: Show Results Sequence Diagram

Figure 4.9 illustrates how the user customize the Search Engine.

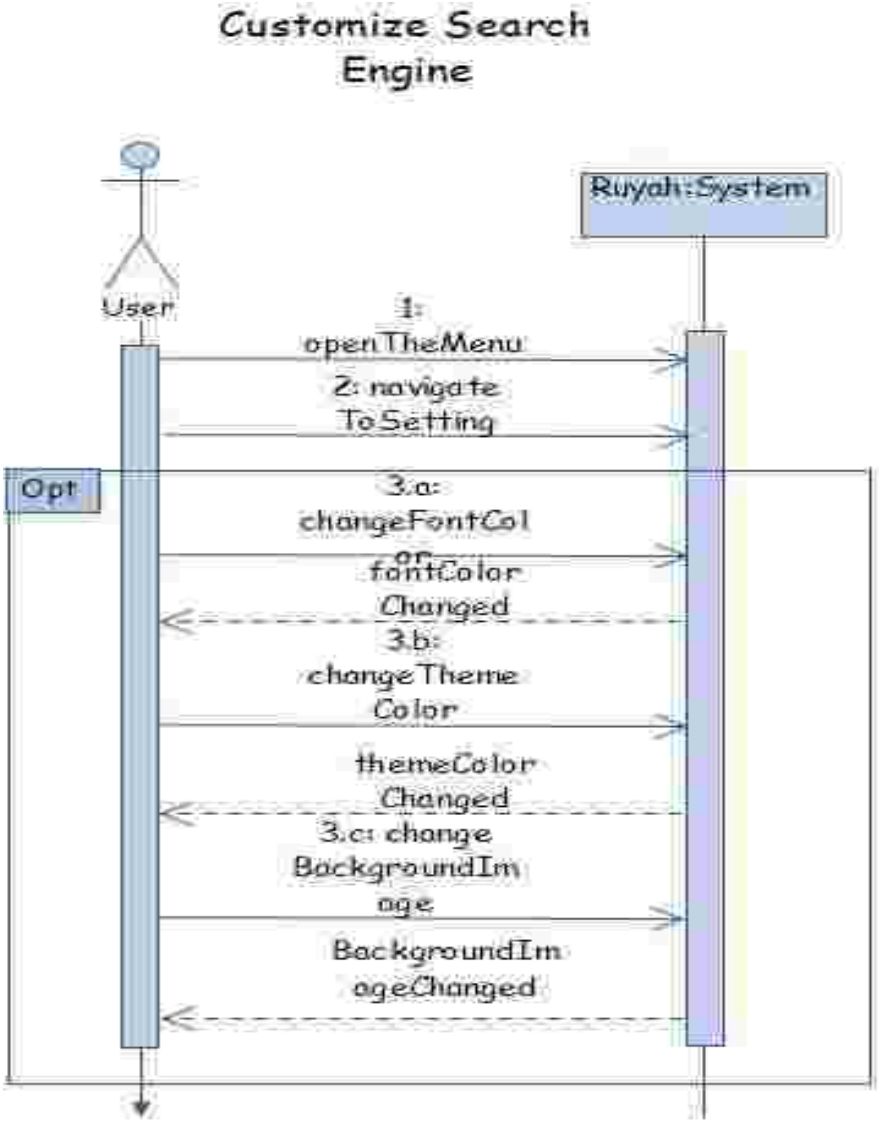

Figure 4.9: Customize Search Engine Sequence Diagram

4. 4 Communication Diagrams

It can be used to model interactions. essentially provide a view of the dynamic aspects of an object-oriented system. They can show how the members of a set of objects collaborate to implement a use case or a use-case scenario. Figure 4.10 illustrates the Input Query Communication Diagram

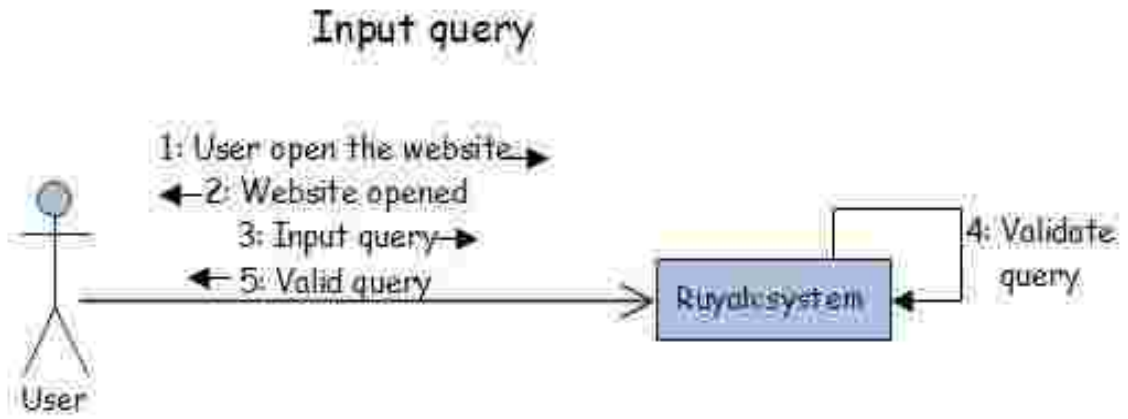

Figure 4.10: Input Query Communication Diagram

Figure 4.11 illustrates the interaction of the process the user query.

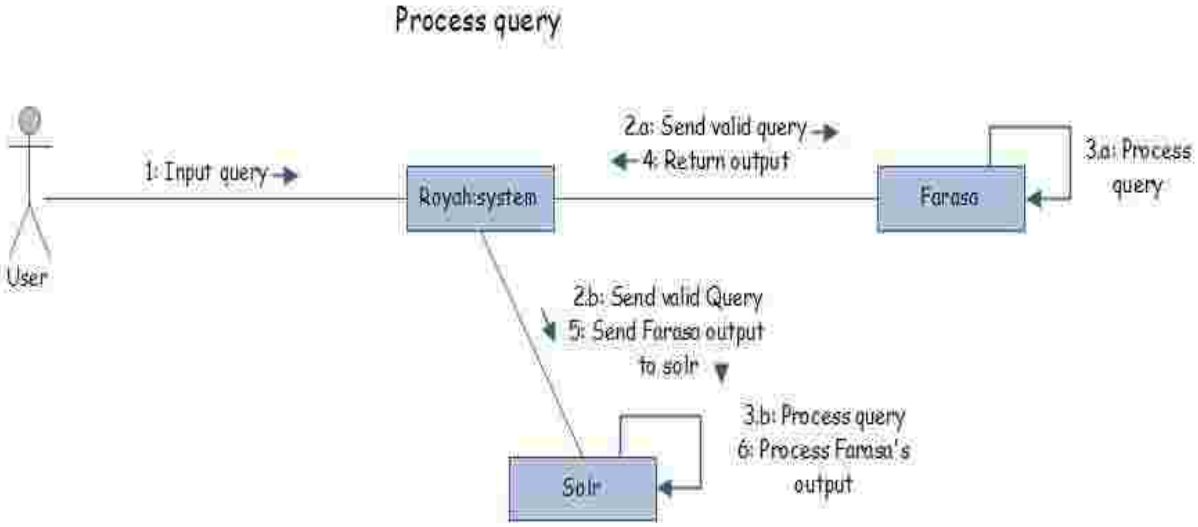

Figure 4.11: Process Query Communication Diagram

Figure 4.12 illustrates the interaction of the search process.

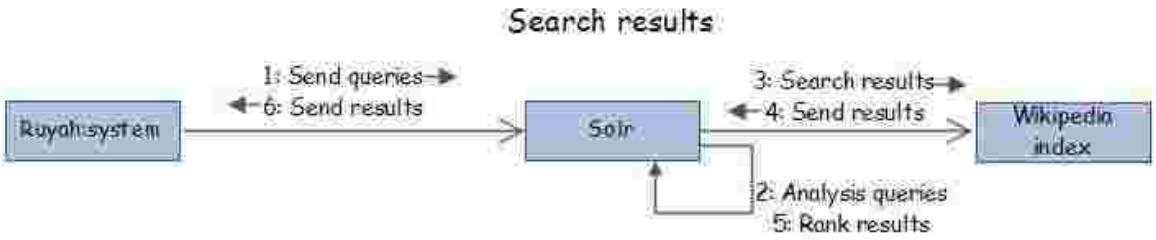

Figure 4.12: Search Results Communication Diagram

Figure 4.13 illustrates the interaction of the select results process.

Figure 4.13: Select Results Communication Diagram

Figure 4.14 illustrates the interaction of the Show results process.

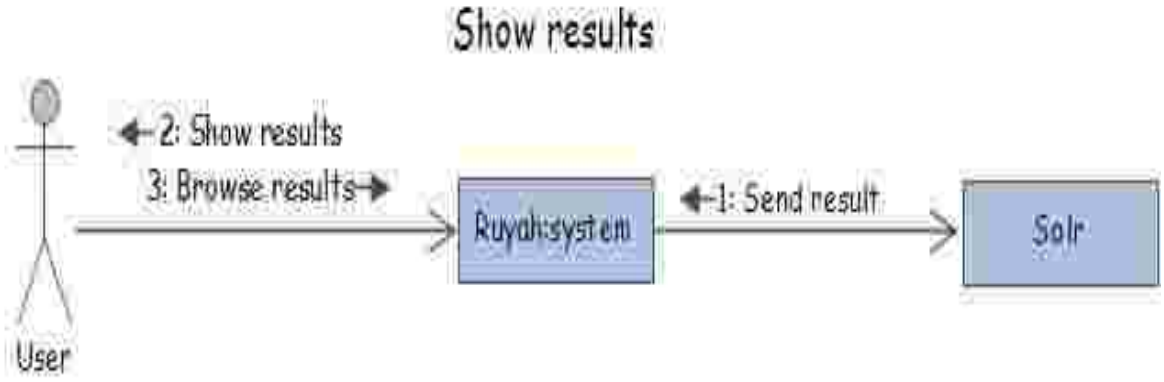

Figure 4.14: Show Results Communication Diagram

Figure 4.15 illustrates the interaction of customizing the search engine appearance.

Figure 4.15: Customize SE Communication Diagram

Chapter Five

Implementation

This chapter presents the implementation stage of how the different components that we used integrated with each other to enhance the search process by Arabic Language. Also it shows how we could implement the concepts we've learned in the search engine that built.

5. 1 The Site Map

A site map is a model of a website's content designed to help both users and search engines navigate the site. [82]

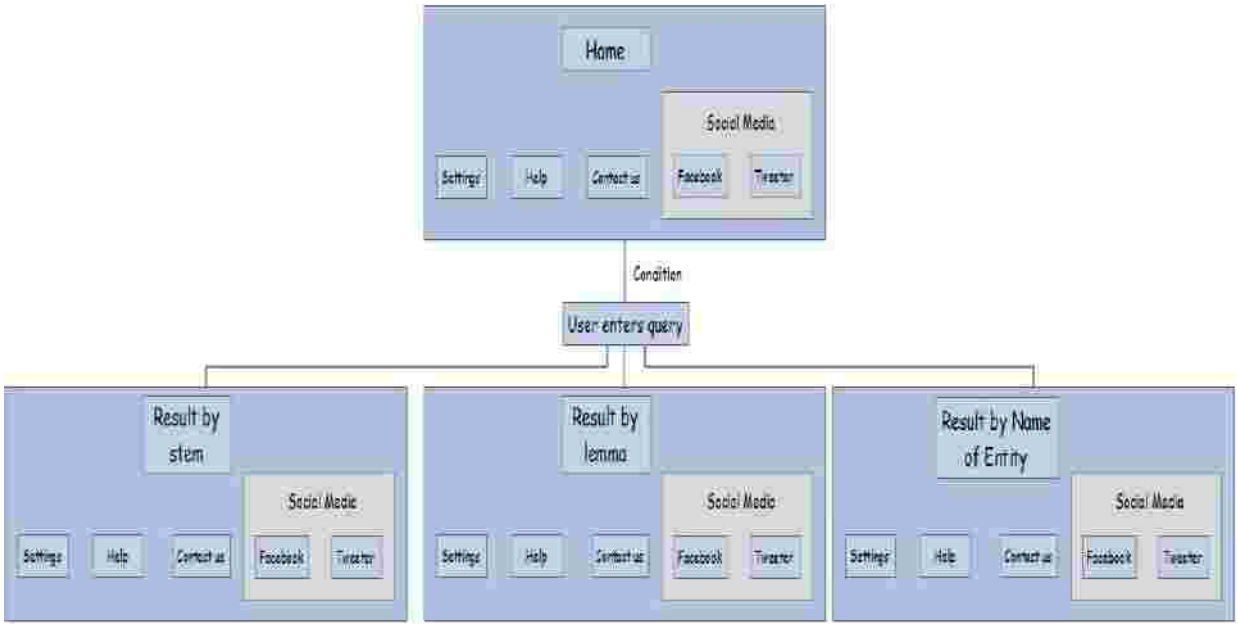

Figure 5.1: The Website Site Map

5. 2 User Interface Implementation

User interfaces (UI) should be designed from the list of user scenarios and requirements that were identified in the user analysis step. [83] This project has four interfaces, which they're: The Home page, Result by stem page, Result by lemma page and Result by Name of entity page. The website has been created be following the rules of Search Engine Optimization.

5.2.1 The Home Page

It's the first page that the user will see. It consists of a textbox, menu and Ruyah's logo. This interface allows the user to enter the query of the search. And it has gray color to be suitable for the eyes and gives the website an official look. When the user clicks on the search box, its color change from white to gray and when the user clicks out of the search box, its backs to white.

Figure 5.2: The Website Home Page

The main menu consists of Home – to back to the home page, Settings - to customize the search engine, Help – Instruction of how to use the search engine, Social Media links – Facebook and Tweeter – for Communication and Email – for send any Feedback for the developer.

Figure 5.3: The Website Menu

The setting in the menu consists of three options: Change font color – to change the color of the fonts except the menu font color, Change the Theme color – to change the color of the menu and background and Change background - to change the background with image from the displayed box or the user can display from his PC.

Figure 5.4: The Website Settings – The setting submenu

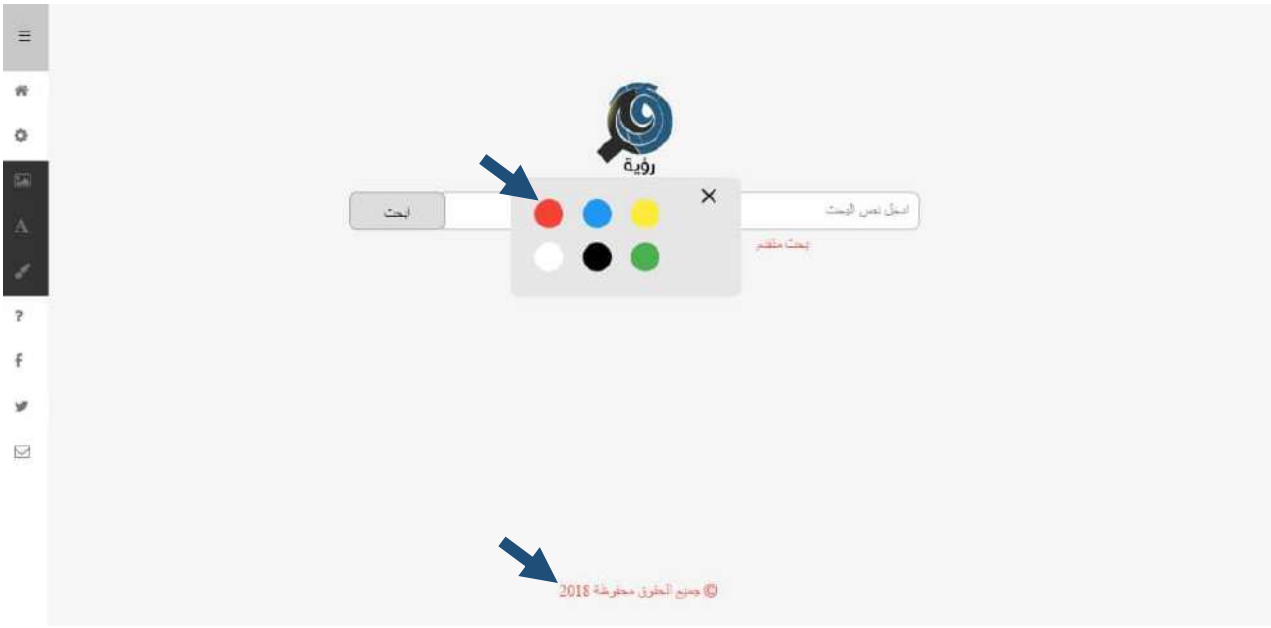

Figure 5.5: The Website Settings – Change Font color option

Figure 5.6: The Website Settings – Change Theme color option

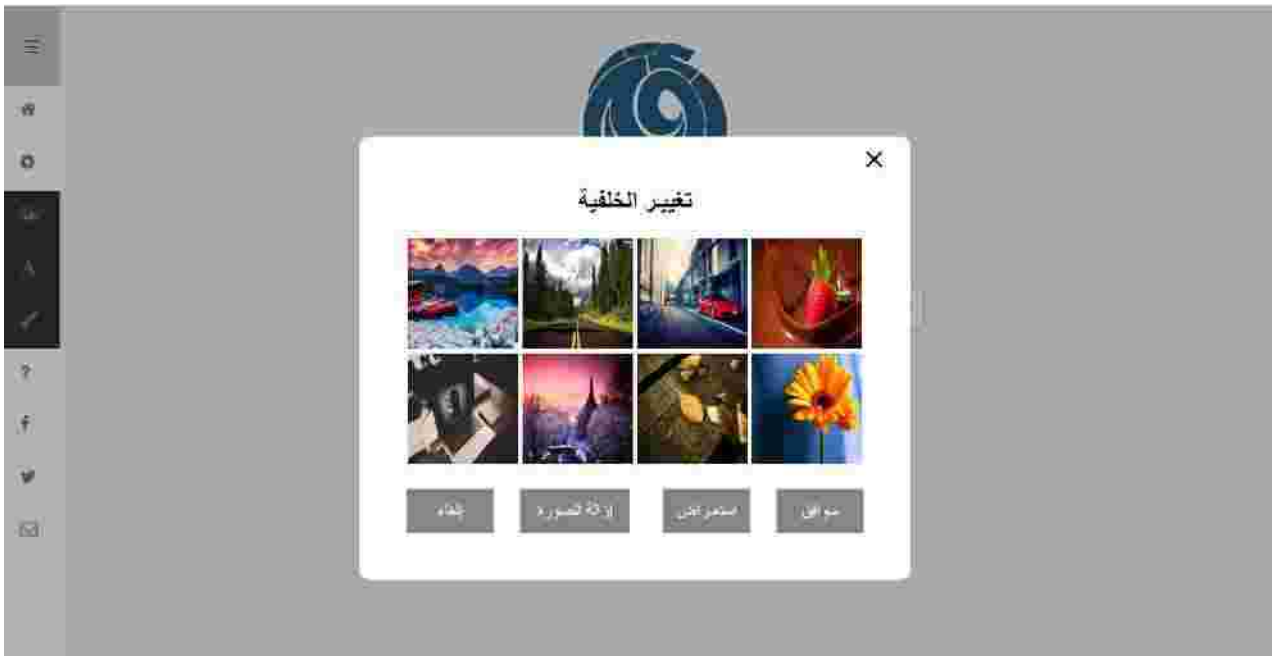

Figure 5.7: The Website Settings – Change Background option

5.2.2 The Result Page

It's the second page that the user will see. It appears after the user clicks on search and it used to display the result of the query. It consists of a textbox, menu and Ruyah's logo. This interface is little bit different from the home page. It shows the results of the Query as Google do, but it uses three different ways for search by stem, lemma and name of entity as shown in Figure 5.8 .

Figure 5.8: The Website Result Page

5. 3 Deployment Diagram

The deployment diagram describes the physical deployment of information generated by the software program on hardware component. The information that the software generates is called an artifact as shown in Figure 5.9.

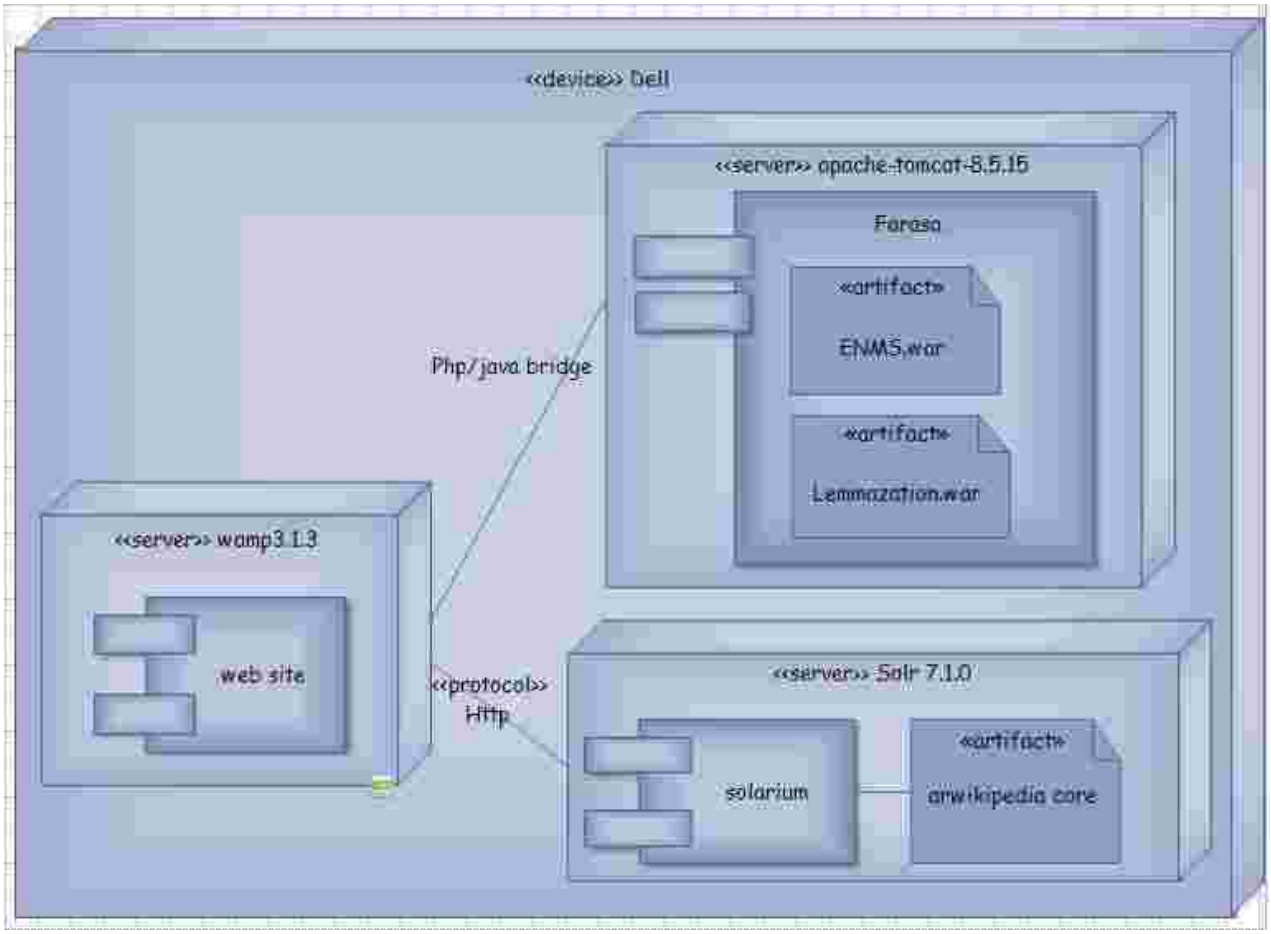

Figure 5.9: The Deployment Diagram

5. 4 The Results

The search engine that output from this project, shows a high performance when searching about the query that the user entered in the search box. The result of searching by stem of the word is the default for this search engine, so its appear in 5 seconds. In the result page there are also two ways for search in addition for the stem, which they are searching by lemma and Name of entity. But they take more time than searching by stem, because the query processing linguistically in Farasa then process again in Solr. Then it matches the query with all the related documents form the Wikipedia Index.

Figure 5.10: The Website Result Page for explain the results

Chapter Six

Conclusion & Recommendations

This chapter displays the results that have been achieved and formed an important part of a project debrief. By summarize the work done in a conclusion and discuss the future works to enhance the project.

6. 1 Conclusion

This website builds to enhance the Arabic search in the first place, by providing the search using stem and lemma of the word. So it integrates different components to achieve that purpose, which are Solr search server and Farasa. The website has a friendly user interface which is known and easy to use. It tries to provide fast search, when the user inputs the query in Arabic language and presses the search button, the results of the stem takes milliseconds to appear. But the result of the lemma & Name of entity take seconds to appear. That because, Farasa processes the query word by word not parallel then returns the output of the processing to the website, which sends it to solr to lookup for relevance documents in the index. Then it shows the results after ranking them according to the most relevant documents. As attempt to solve the latency of lemma's & name of entity's results, the website after first search saves the lemma's and the name of entity's query in the URL. So if the user navigates again to the lemma's results or name of entity the results displays faster than the previous. This website provides customization for the Appearance of the it, by changing the color of the font and theme and also change the background of it with an image. The accuracy of the website's results has been compared with the results of Google search engine. The comparison shows that, when typing the same Arabic query Google search engine recognizes only few result that related to the Wikipedia, but the new search engine recognizes a lot of results that can be hundred or thousand. The website itself builds using some of the search engine optimization rule that related to optimize the ranking of it using the meta tags, when search using any browser.

6. 2 Future work

1) Make apache solr support the search by the different dialects.
2) Integrate apache solr with Nutch (crawler).
3) Expand apache solr scope to include all webpages.
4) Develop the morphology system.
5) **Design a search mechanism with maintainable structure:** that allows adding of new linguistic resources to enrich the search process.
6) Expand the using of corpuses, to comprehend more than one corpus.
7) Rebuild Farasa in a way that make it more suitable for integration with Apache project.

REFERENCES

[1] Brin, S. and Page, L., **The Anatomy of a Large-Scale Hyper-Textual Web Search Engine,** Computer Science Department, Stanford University, Stanford, CA 94305, USA. 1998

[2] Hawking, D., **Web Search Engines,** CSIRO ICT Centre. (June 2006).

[3] Khalid, A., Hussain Z., Anwarullah, M. A., **Arabic Stemmer for Search Engines Information Retrieval,** International Journal of Advanced Computer Science and Applications, Vol.7, KSA, No.1. 2016.

[4] From Wikipedia, the free encyclopedia. **Natural Language**. [https://en.wikipedia.org/wiki/Natural_language]

[5] From Yseop. What is the difference between Natural Language Generation and Natural Language Understanding? [https://yseop.com/blog/what-is-the-difference-between-natural-language-generation-and-understanding-2/]

[6] Reiter, E., Dale, R. and Feng, Z. ***Building natural language generation systems,*** MIT Press. 2000.

[7] Reshamwala, A., Pawar, P. and Mishra, D. ***REVIEW ON NATURAL LANGUAGE PROCESSING,*** IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498, Vol.3, No.1. February 2013.

[8] Liu, D., Li, Y. and Thomas, M. A., ***A Roadmap for Natural Language Processing Research in Information Systems***, Proceedings of the 50th Hawaii International Conference on System Sciences. 2017.

[9] Hirschberg, J. and Manning, C. D. Advances ***in natural language processing***, available at: http://science.sciencemag.org/ , 2015. last visited at 25/1/2018.

[10] Gupta, V., ***A Survey of Natural Language Processing Techniques***, International Journal of Computer Science & Engineering Technology (IJCSET), ISSN: 2229-3345, Vol. 5 No. 01 Jan 2014.

[11] Tembhekar, S. and Kanojiya, M., ***A Survey Paper on Approaches of Natural Language Processing (NLP),*** Tembhekar Samta, Kanojiya Monika, International Journal of Advance Research, Ideas and Innovations in Technology, ISSN: 2454-132X, Impact factor: 4.295, Volume3, Issue3, Available online at www.ijariit.com. 2017.

[12] From Slideshare. **Natural Language Processing: L01 introduction**. Available at: [https://www.slideshare.net/ananth/natural-language-processing-l01-introduction?from_action=save], last visited at 29/1/2018.

[13] C.OSMAN, C. and ZĂLHAN, P. G., ***From Natural Language Text to Visual Models: A survey of Issues and Approaches***, Informatica Economică vol. 20, no. 4. 2016.

[14] Jurafsky, D. and Martin J. H.**, *Speech and Language Processing***, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Second Edition, Stanford University. 2009.

[15] Khurana, D., Koli, A., Khatter, K. and Singh, S., ***Natural Language Processing: State of The Art, Current Trends and Challenges***, ArXiv Preprint ArXiv:1708.05148. 2017.

[16] Liddy, E. D., ***Natural Language Processing***, Syracuse University. 1996.

[17] herbrich, R. and graepel T., ***Handbook of Natural Language Processing***, Second Edition, springer. 2010.

[18] Vlachidis, A. and Tudhope, D., ***Semantic Annotation for Indexing Archaeological Context: A Prototype Development and Evaluation***, vol 240. Springer, Berlin, Heidelberg. 2011.

[19] Habash, Y. N., ***Introduction to Arabic Language Processing***, A Publication in the Morgan & Claypool Publishers series, ISBN: 9781598297966. P 114. 2010.

[20] VERSTEEGH, K. ***The Arabic Language***. Columbia University Press, New York. 1997.

[21] ATTIA, M., ***Handling Arabic morphological and syntactic ambiguities within the LFG framework with a view to machine translation***. PhD Dissertation, University of Manchester. 2008.

[22] CIA. *CIA* ***Word Fact*** *Book*. Central Intelligence Agency, Washington, D.C. 2008.

[23] Doumi, N., Lehireche, A., Maurel, D. & Abdelali, A., ***A Semi-Automatic and Low Cost Approach to Build Scalable Lemma-based Lexical Resources for Arabic Verbs***. Information Technology and Computer Science. February 2016.

[24] ATTIA, M. ***A large scale computational processor of Arabic morphology and applications***. Master's Dissertation, Computer Engineering, Cairo University, Egypt. 1999.

[25] BEESLEY, K., ***Finite-state morphological analysis of Arabic at Xerox Research: Status and plans in 2001***. In *Proceedings of the Workshop on Arabic Natural Language Processing at the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*. 1–8. 2001.

[26] BUCKWALTER, T., ***Issues in Arabic orthography and morphology analysis***. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages (CAASL'04)*. 31–34. 2004.

[27] FARGHALY, A. ***Three level morphology for Arabic***. In *Proceedings of the Arabic Morphology Workshop (AMW'87)*. 1987.

[28] MCCARTHY, J. ***A prosodic theory of nonconcatenative morphology***. *Linguistic Inquiry. 12,* 373–418. 1981.

[29] SOUDI, A., BOSCH, A., AND G ¨ UNTER, N., eds. ***Arabic Computational Morphology: Knowledge-Based and Empirical Methods (Text, Speech, and Language Technology)***, Springer. 2007.

[30] Shaalan, K., ***Rule-based Approach in Arabic Natural Language Processing***, international Journal on Information and Communication Technologies, Vol. 3, No. 3, **p. 2. June 2010.**

[31] Azzam, R. ***Reading and writing disorders in different orthographic systems***, In the linguistics of literacy, Orthography and reading of the Arabic language. In J. Aaron & R. M. Joshi (Eds.), 203-218. (1989).

[32] Abd El-Minem, I.M. Elm al-Sarf, (***Arabic grammer***). Jerusalem: Al-Taufik Press (in Arabic). (1987).

[33] Ibrahim, R. ***Does visual and Auditory Word Perception have a Language-Selective Input? Evidence from Word Processing in Semitic languages***. Linguistic Journal, 3, 82-103. (2008).

[34] Aula, Kh., Abu-Leil, David, L. Share, Ibrahim, R., ***How does speed and accuracy in reading relate to reading comprehension in Arabic?*** The Edmond J. Safra Brain Research Center for the Study of Learning Disabilities, University of Haifa, 35, 251-276. (2014).

[35] Altantawy, M., Habash, N., and Rambow, O. ***Fast Yet Rich Morphological Analysis***, Center for Computational Learning Systems Columbia University, New York, USA, Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing, pages 116–124, Blois (France), Association for Computational Linguistics. **July 12-15, 2011. c 2011**.

[36] SANAN, M., RAMMAL, M., ZREIK, K. INTERNET, ***ARABIC SEARCH ENGINES STUDIES*.** July 16,2010.

[37] Lahjouji, W. ***Platform for Assessing and Experimenting with Complexity of Arabic Grammatical Structures***. **Spring 2015.**

[38] HABASH, N. and OWEN, R. ***Arabic tokenization, part of speech tagging and morphological disambiguation in one fell swoop***. In Proceedings of the Association for Computational Linguistics (ACL'05). 2005.

[39] SHAALAN, K. ***An intelligent computer-assisted language learning system for Arabic learners***. J. Int. Comput. Assist. Lang. Learn. 18, 1/2, 81–108. 2005a.

[40] SHAALAN, K. ***Arabic GramCheck: A Grammar Checker for Arabic, Software Practice and Experience***. John Wiley & Sons, Ltd., 643–665. 2005b.

[41] FARGHALY, A., SHAALAN, K., ***Arabic Natural Language Processing: Challenges and Solutions.*** ACM Transactions on Asian Language Information Processing, Vol. 8, No. 4, Article 14, Pub. date: December 2009.

[42] FARGHALY, A., A case for inter-Arabic Grammar. In Eligbali, A., Ed., ***Investigating Arabic: Current Parameters in Analysis and Learning***. Brill, Boston. 2005.

[43] FERGUSON, C., ***Diglossia***. WORD, 15 3, 325–340. 1959.

[44] LARKEY, L. AND CONNELL, M. E. ***Arabic information retrieval at UMASS in TREC-10***. In Proceedings of the 10th Text Retrieval Conference (TREC'01). 2001.

[45] CHOMSKY, N. ***Aspects of the theory of syntax***. MIT Press, Cambridge, MA. 1965.

[46] FARGHALY, A. ***Arabic NLP: Overview, the state of the art: Challenges and opportunities***. In Proceedings of the International Arab Conference on Information Technology (ACIT'08). 2008.

[47] Al Daoud, E., & Basata, A. ***A Framework to Automate the Parsing of Arabic Language Sentences***. *No. 2*. Jordan: The International Arab Journal of Information Technology. (2009, April 20).

[48] Al-Taani, A., Msallam, M., & Wedian, S. ***A Top-Down Chart Parser for Analyzing Arabic Sentences***. (2012).

[49] Attia, M. ***An ambiguity-controlled morphological analyzer for modern standard Arabic modeling finite state networks***. *pp. 155-159*. London: The Challenge of Arabic for NLP/MT Conference. (2006).

[50] Attia, M. ***Handling Arabic morphological and syntactic ambiguity within the LFG framework with a view to machine translation***. (M. Attia, Trans.) UK. (2008).

[51] Bataineh, B. M., & Bataineh, E. A. ***An Efficient Recursive Transition Network Parser for Arabic Language***. (2009).

[52] Alqrainy, S., Muaidi, H., & Alkoffash, M. S. ***Context-Free Grammar Analysis for Arabic Sentences***. *Volume 53 - No. 3*. Salt, Jordan: International Journal of Computer Applications (0975 - 8887). (2012, September).

[53] Othman, E., Shaalan, K., & Rafea, A. ***A Chart Parser for Analyzing Modern Standard Arabic***. (2003).

[54] Roth, R., Rambow, O., & Habash, N. ***Mada+ tokan: A toolkit for arabic tokenization diacritization, morphological disambiguation, pos tagging, stemming and lemmatization***. *In: Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR)*, (pp. 102-109). Cairo. (2009).

[55] Diab, M. ***Second generation amira tools for arabic processing: Fast and robust tokenization, pos tagging, and base phrase chunking***. 2nd International Conference on Arabic Language Resources and Tools. Citeseer. (2009).

[56] Manning, C., Klein, D., & Toutanova, K. ***Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network***. *HLT-NAACL*, pp. 252-259. (2003).

[57] Diab, M., Ghoneim, M., & Habash, N. ***Arabic Diacritization in the Context of Statistical Machine Translation***. (2007).

[58] Habash, N. Y., & Roth, R. M. ***CATiB: The Columbia Arabic Treebank***. (the ACL-IJCNLP 2009 Conference Short Papers). Association for Computational Linguistics. (2009).

[59] Dukes, K., & Buckwalter, T. ***A dependency Treebank of the Quran using traditional Arabic grammar***. Cairo, Egypt: the 7th International Conference on Informatics and Systems (INFOS). (2010, Mach 28-30).

[60] Khoufi, N., Louati, S., Aloulou, C., & Belguith, L. H. ***Supervised learning model for parsing Arabic language***. Sfax, Tunisia. (2014).

[61] Shahrour, A., Khalifa, S., & Habash, N. ***Improving Arabic Diacritization through Syntactic Analysis***. Conference on Empirical Methods in Natural Language Processing. (2015, Sep).

[62] Habash, N., & Rambow, O., ***Arabic Diacritization through Full Morphological Tagging***. New York. (2007, April).

[63] Ibrahim, M., Mahmoud, M., & El-Reedy, D., Bel-Arabi: ***Advanced Arabic Grammar Analyzer***. *No. 5*(Vol. 6). International Journal of Social Science and Humanity. (2016, May).

[64] Alfares, A. W., ***Statistical Arabic Grammar Analyzer Based on Rules Mining Approach Using Naïve Bayesian Algorithm***. Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Computer Science. Middle East University. 2017, January.

[65] FARGHALY, A. ***Subject pronoun deletion rule***. In Proceedings of the 2nd English Language Symposium on Discourse Analysis (LSDA'82). 110–117. 1982.

[66] SHAALAN, K. AND RAZA H. ***Arabic named entity recognition from diverse text types***. In Proceedings of the 6th International Conference on Natural Language Processing (GoTAL'08). B. Nordstr¨om, and A. Ranta, Eds. 2008.

[67] CHOUKRI, K. ***MEDAR: Mediterranean Arabic language and speech technology: Inventory of the HLT products, players, projects and language resources***. In

Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR'09). 2009.

[68] Attia, M., Pecina, P., Tounsi, L., Toral, A. and Genabit, J. van., ***Lexical Profiling for Arabic***, Dublin City University, Dublin, Ireland. (**2011**)

[69] Attia, M. A., ***An Ambiguity-Controlled Morphological Analyzer for Modern Standard Arabic Modelling Finite State Networks,*** School of Informatics, The University of Manchester. 2009

[70] Bhupendra R. ***Search Engine***. School of Library and Information Science. 2010.

[71] From Quora. **What are the main uses of web search engines**? Available at: [https://www.quora.com/What-are-the-main-uses-of-web-search-engines]

[72] From Wamda. **Specialized search engines from the Arab region**. Available at: [https://www.wamda.com/2013/03/10-specialized-search-engines-from-the-arab-region]

[73] From Wikipedia, the free encyclopedia. **Google Search**. Available at: [https://en.wikipedia.org/wiki/Google_Search]

[74] From Wikipedia, the free encyclopedia. **Bing**. Available at: [https://en.wikipedia.org/wiki/Bing_(search_engine)]

[75] From Wikipedia, the free encyclopedia. **Yahoo**. Available at: [https://en.wikipedia.org/wiki/Yahoo!_Search]

[76] From Extra digital. **Arabic search engines**. Available at: [https://www.extradigital.co.uk/articles/seo/arabic-search-engines.html]

[77] From Networked Life. **How do Google, Bing and Yahoo search?** Available at: [http://networkedlifeq21.wikia.com/wiki/HOW_DO_GOOGLE,_BING_AND_YAHOO_SEARCH_ENGINES_WORK%3F_WHO_IS_WINNING_THE_SEARCH_WAR%3F_by_LakshmiChaitanya_Madiraju,_Sri_Rohith_Talluri]

[78] Martínez, A. G., ***A computational model of Modern Standard Arabic verbal morphology based on generation***, Madrid. December 2012.

[79] From QCRI, **Farasa**. Available at: [http://qatsdemo.cloudapp.net/Farasa/]

[80] From QCRI, **Farasa: Fast and Accurate Arabic Word Segmenter**. Available at: [http://alt.qcri.org/farasa/segmenter.html]

[81] From Tutorialspoint, **Apache Solr tutorial**. Available at: [https://www.tutorialspoint.com/apache_solr/index.htm]

[82] From Techopedia, **Site Map**. Available at: [https://www.techopedia.com/definition/5393/site-map]

[83] From Microsoft, **Implementing a User Interface**. Available at:

[https://msdn.microsoft.com/en-us/library/windows/desktop/ff728823(v=vs.85).aspx]

[84] From Wikipedia, **Wikipedia**. Available at:

[https://en.m.wikipedia.org/wiki/Wikipedia]

[85] From Wikipedia, **Inverted Index**. Available at:

[https://en.m.wikipedia.org/wiki/Inverted_index]

[86] From Wikipedia, **Search Engine Indexing**. Available at:

[https://en.m.wikipedia.org/wiki/Search_engine_idexing]

APPENDIX A

The Questionnaire Sample

The Questionnaire was designed and written using Google Form service, that offer for free from Google Inc. and facilitate the publish of the Questionnaire. Our Questionnaire wrote in Arabic Language.

a) The Questionnaire Title: "Search Engines" – "محركات البحث".

b) Brief Description: "This Questionnaire helps for build a Search Engine that enhance the Arabic search in the Arabic Wikipedia" – "هذا الاستبيان للمساعدة في بناء محرك بحث يحسن عملية البحث في الويكيبيديا باللغة العربية".

c) Questionnaire's Questions:

1) ***العمر**:

أقل من 18

بين 18 و 30

أكثر من 30

2) ***الجنس**:

ذكر.

أنثى.

3) ***مدى إتقانك للغة العربية**:

ممتاز.

متوسط.

ضعيف.

4) ***مدى إتقانك للغة الإنجليزية**:

ممتاز.

متوسط.

ضعيف.

5) ***ماهي اللغة التي تفضل البحث بها**:

العربية

الإنجليزية

6) ***في رأيك هل تدعم جميع المتصفحات اللغة العربية بشكل جيد**:

نعم.

لا.

ربمـا.

7) ***ما الذي يمكن أن يكون حافزاَ\سبباَ للبحث باللغة العربية**:

صعوبة البحث باللغة العربية أو أي لغة أخرى.

عدم إتقان لغة أخرى غيرها.

البحث عن مصادر عربية فقط.

8) ***ما الذي يمكن أن يكون عائقاَ للبحث باللغة العربية**:

عدم دقة النتائج باللغة العربية.

اللهجـات.

صعوبة قواعد اللغة العربية.

توفر المواد والمصادر باللغة العربية.

قدم المعلومات في المصادر العلمية المتوافرة باللغة العربية.

9) ***كم إجمالي الوقت الذي تقضيه في البحث**:

أقل من ساعة.

من ساعة إلى 3 ساعات.

أكـثر.

10) ***ما غرضك الرئيسي من استخدام محرك البحث**:

للبحث العلمي.

للتسلية.

غير ذلك.

11) ***ماهو أكثر محرك بحث تستخدمه**:

قوقل (Google)

يـاهو (Yahoo)

بينغ (Bing)

12)***كم من الوقت تستغرق لإيجاد ما تبحث عنه باللغة العربية** :

أقل من ساعة.

ساعة أو أكثر.

على حسب نوعية البحث.

أكثر من ذلك.

13)***ما الذي تريد إضافته لمحرك البحث**:

تغيير خلفية محرك البحث.

تغيير ألوان محرك البحث.

وجود تعليمات لكيفية استخدام محرك البحث.

جميع ما سبق.

14) **إذا كان هناك أي تعليق أو مميزات أو آراء إضافية عن المشاكل التي تواجهك للبحث باللغة العربية ... فأضفها هنا ...**

d) Questionnaire's Results

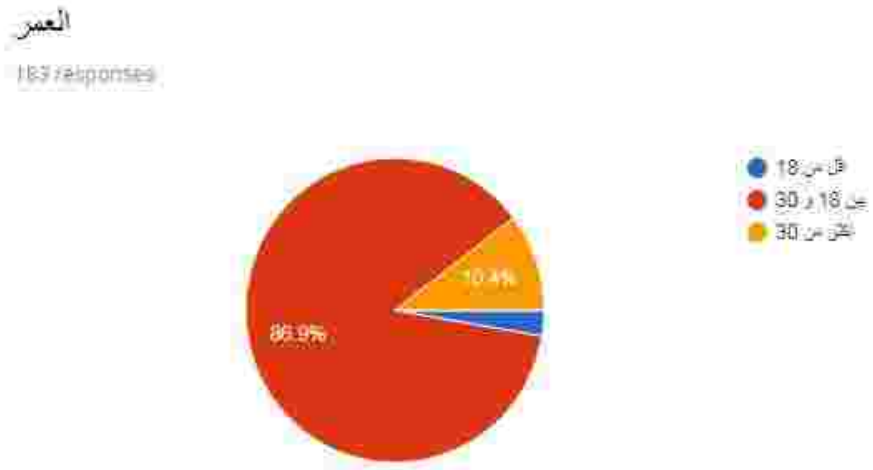

Figure A-1: Questionnaire's result of Age

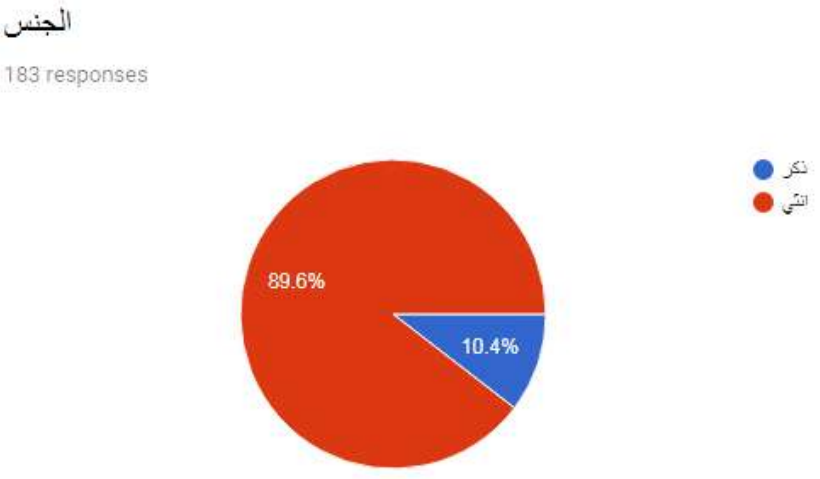

Figure A-2: Questionnaire's result of Gender

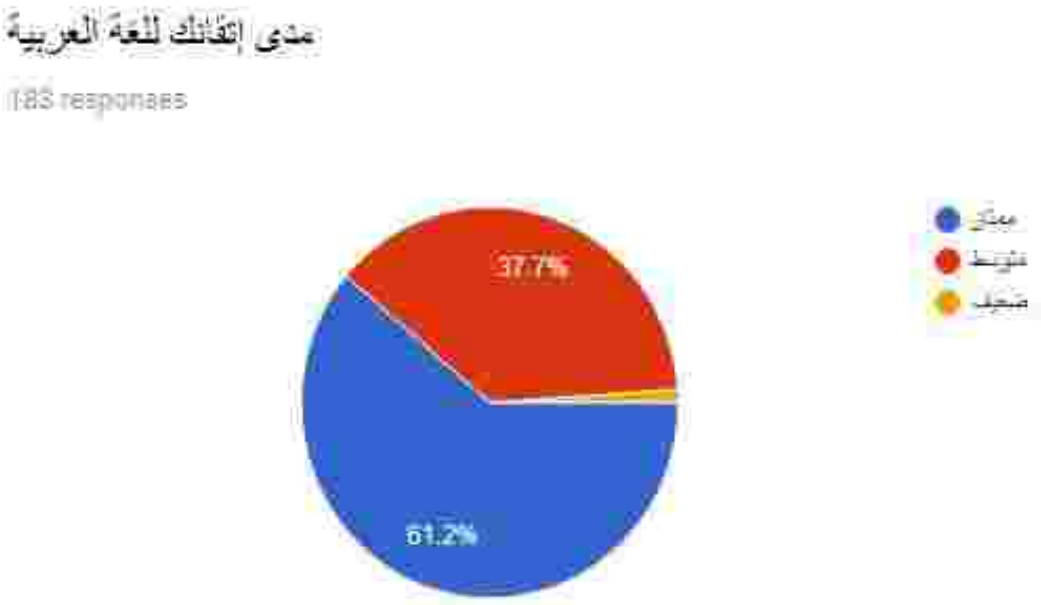

Figure A-3: Questionnaire's result of dexterity of Arabic

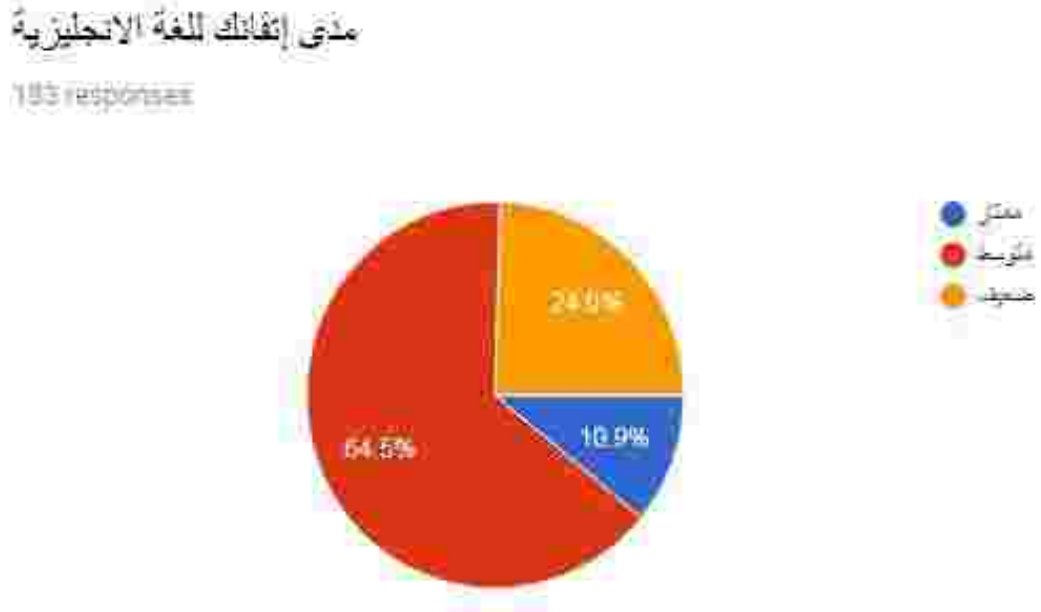

Figure A-4: Questionnaire's result of dexterity of English

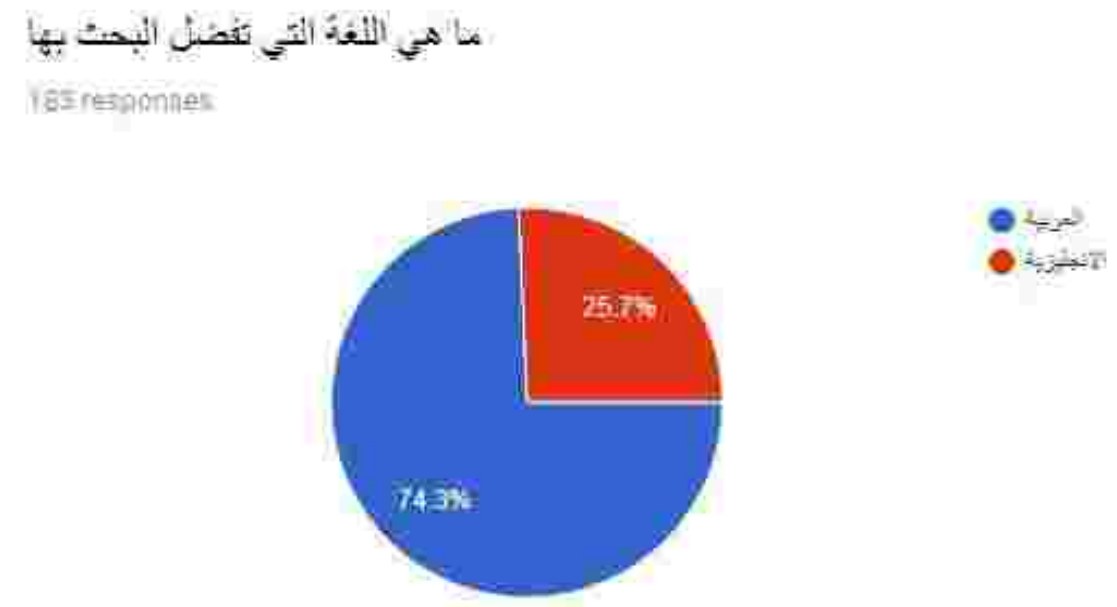

Figure A-5: Questionnaire's result of preferred search language

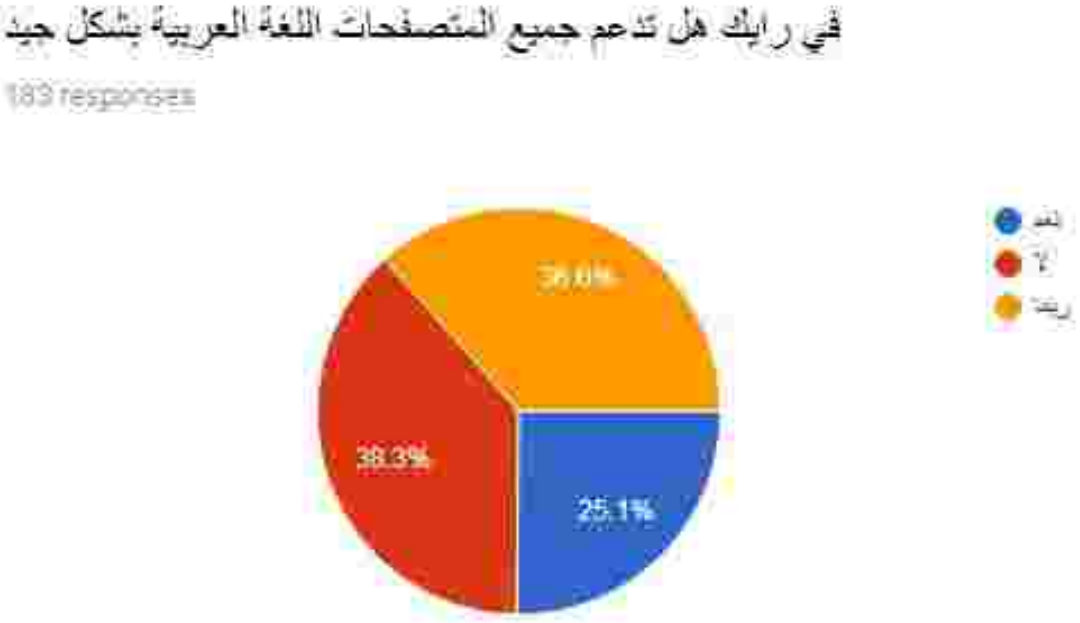

Figure A-6: Questionnaire's result of SEs that support Arabic

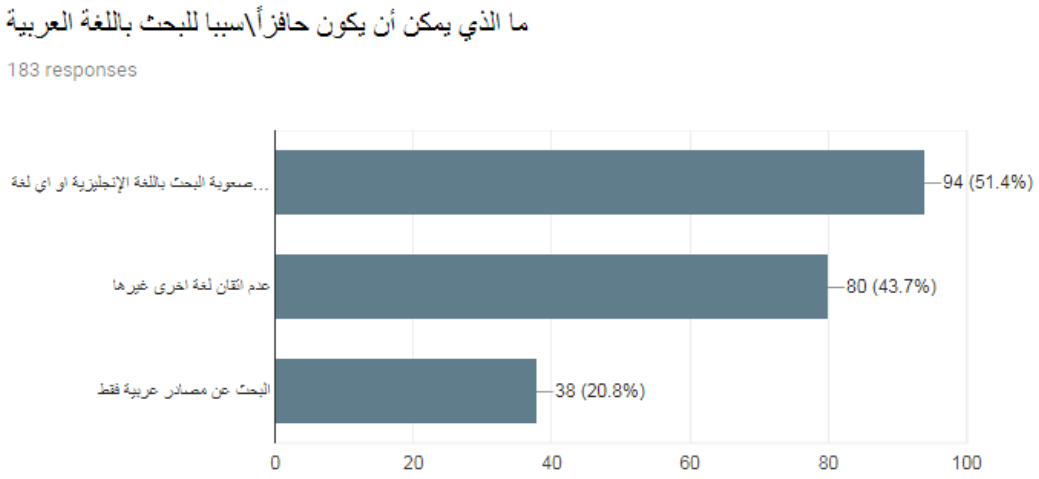

Figure A-7: Questionnaire's result of motivations for search in Arabic

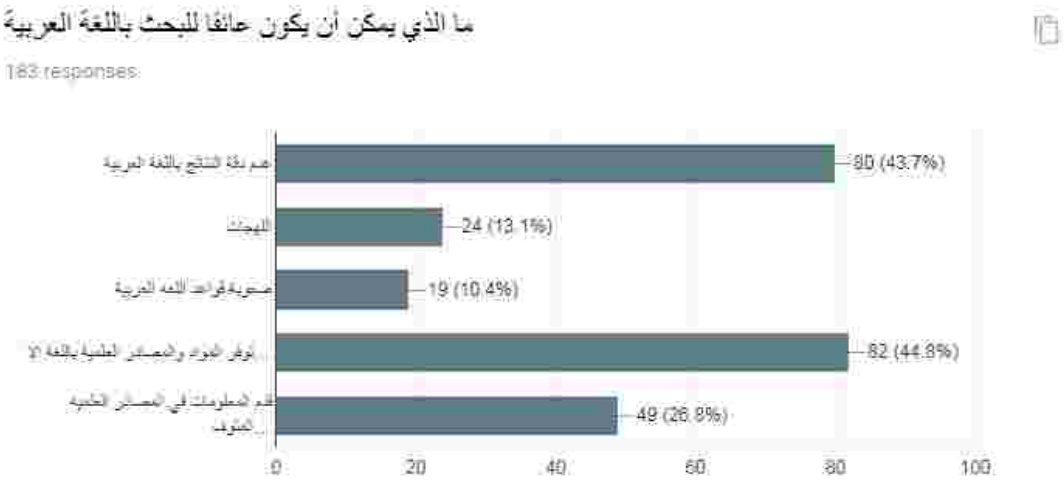

Figure A-8: Questionnaire's result of obstruction for search in Arabic

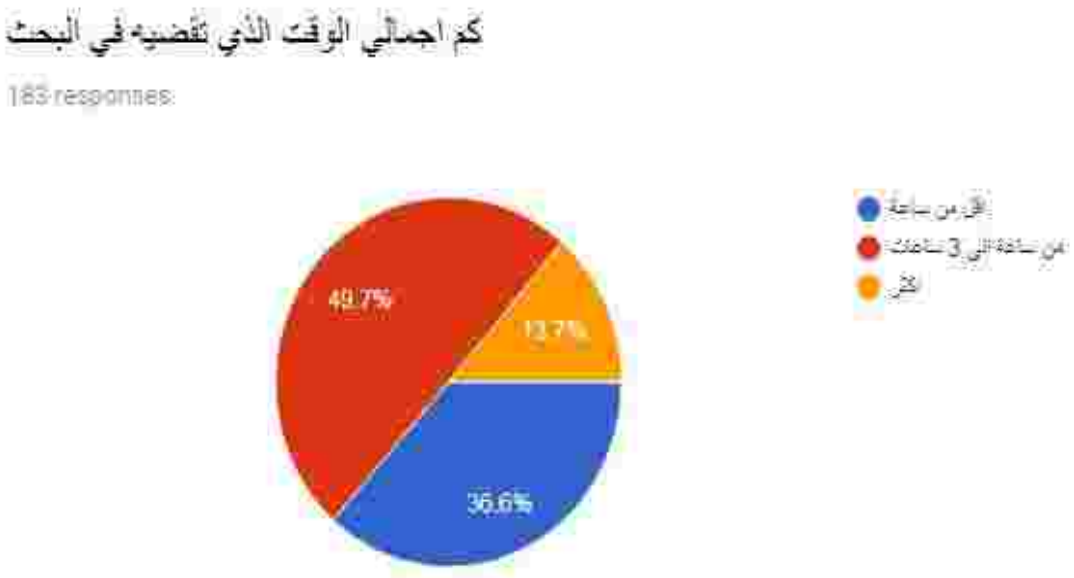

Figure A-9: Questionnaire's result of obstruction for search

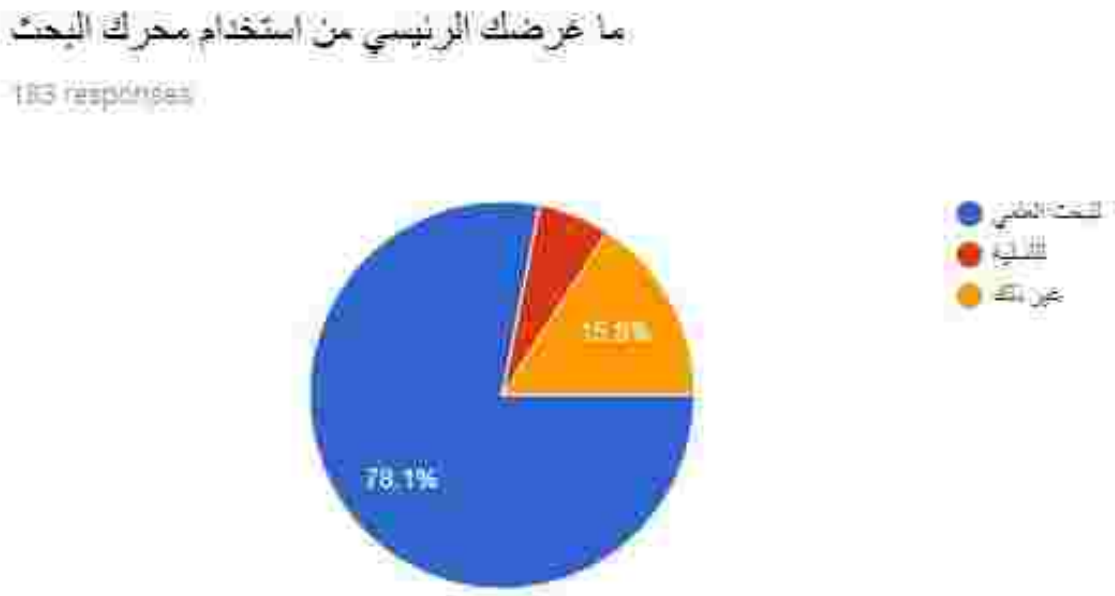

Figure A-10: Questionnaire's result of main purpose of search

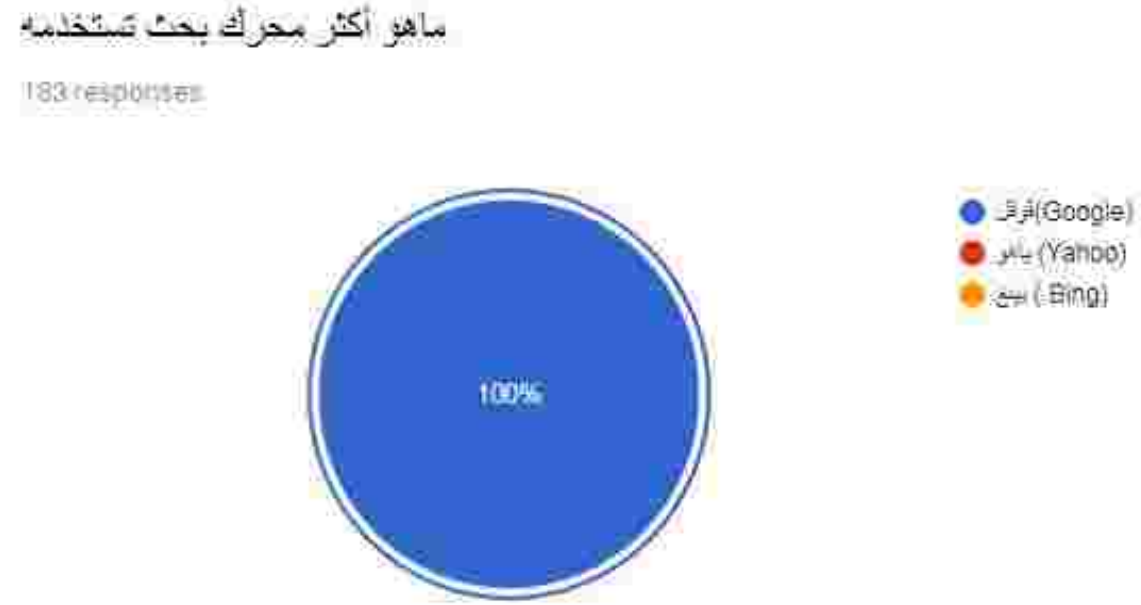

Figure A-11: Questionnaire's result of the most used Search Engine

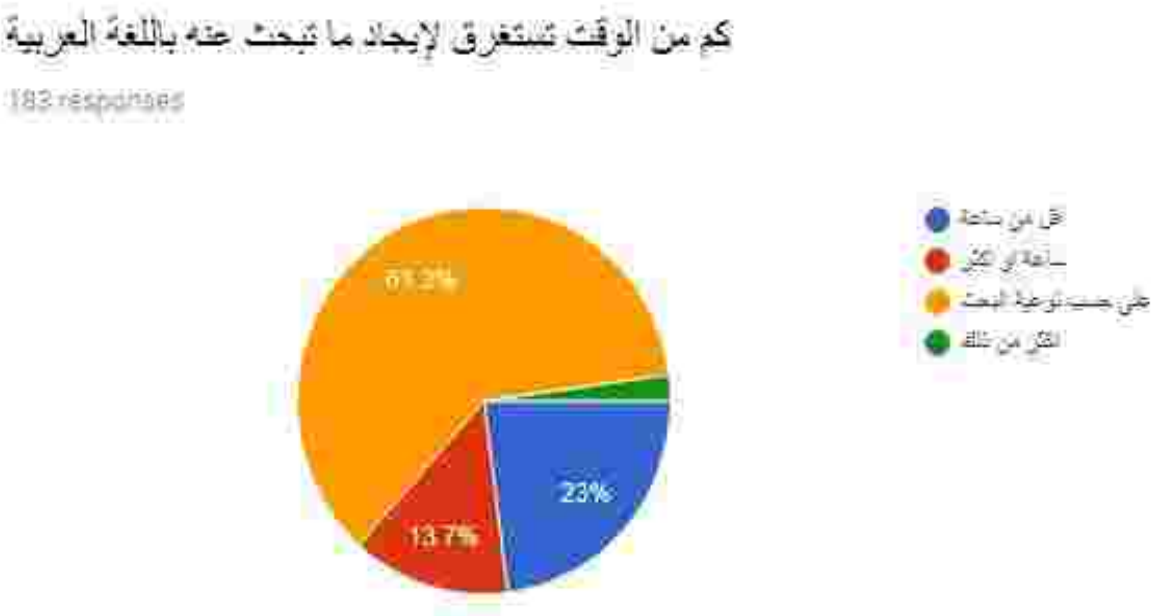

Figure A-12: Questionnaire's result of time wasted in search using Arabic Language

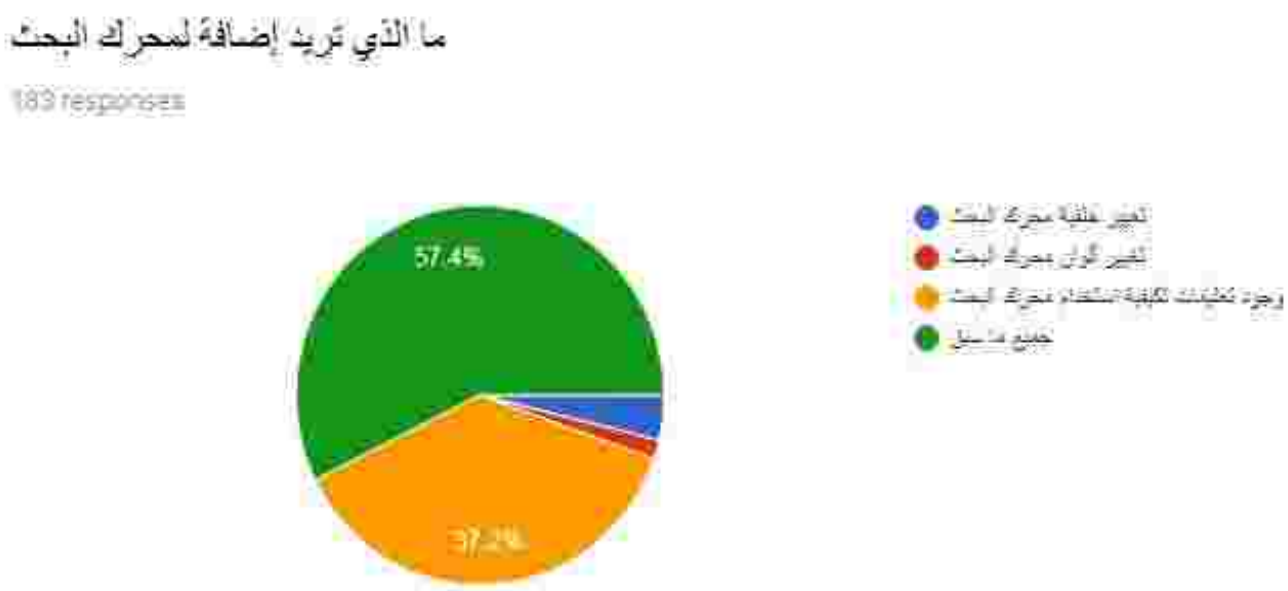

Figure A-13: Questionnaire's result of adding features to SE

- إذا كان هناك أي تعليق او مميزات أو آراء إضافية عن المشاكل التي تواجهك عند البحث باللغة العربية ... فأضفها هنا ...

26 responses

Here Showing some of the responses and comments as a sample:

2) كمية المعلومات التي نحصل عليها عند البحث باللغة العربية قليلة جدا وكذلك المواقع وخاصة اذا كان الموضوع علمي.
3) قلة المصادر المتوفرة باللغة العربية أكبر عائق لتجنب البحث باللغة العربية .. فمن يكره أن يجد ما يبحث عنه بلغته العربية فهذا يجعل الفهم أسرع و أسهل.
4) أحيانا أريد أن ابحث عن معنى أو مفهوم لجملة أو موضوع لكنه غير مفهوم لدى جوجل أيضا أفضل لو أنه هناك متصفح يأخذ ذلك المكتوب ويعطي كل الاحتمالات الممكنة ومنه تستطيع أن تأخذ منه مقصدك.
5) وجود الكثير من الإعلانات الممولة أو البوب أدز وكذلك المماطلة في الكلام والتلاعب في الجمل وغير ذلك إحتوى إعلانات بكثرة وأيضا الكثير من التعليقات السلبية بعد كل مقالة أو بحث ذلك مما يجعلنا نبحث باللغة الانجليزية لتفادي هذه الاشياء.
6) المعلومات التي باللغة الانجليزية تكون أدق واكثر أتمنى أن أجد معلومات قيمة ودقيقة وكثيفة باللغة العربية.. مع خالص امتناني.
7) استيعاب جميع اللهجات.

Component Implementation

There are five components that used to implement this project, which they are Farasa Package, Solr Search Server, Wikipedia Dump, PHP/JAVA bridge and Solarium PHP/SOLR library.

1) Farasa Packages

Farasa as mentioned in the related work part, it is a text processing toolkit for Arabic text. This tool has many packages for manipulate the Arabic text. These packages can be downloaded from the internet, then configure them to run in Windows or Linux. In this project, Farasa configured to run on windows. Also the code has been modified to be compatible with Solr search server. Two packages only used from Farasa: The Segmentation Package & The Name of Entity Package.

I) FARASA Segmenter Package

```
package com.qcri.farasa.segmenter;
import static com.qcri.farasa.segmenter.ArabicUtils.normalizeFull;
import static com.qcri.farasa.segmenter.ArabicUtils.removeDiacritics;
import static com.qcri.farasa.segmenter.ArabicUtils.tokenize;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.TreeMap;
import java.util.List;
public class TestCase
{
private enum schemes{q1, q0, atb, def};
public static String results="";
public static String resultl="";
public static void main(String[] args) throws FileNotFoundException, IOException,
ClassNotFoundException, InterruptedException, Exception
   {

      String scheme = "def";
      Boolean norm = false;
      String arg = args[0];
```

```
        String lemr="";
         lemr=lemmatize(scheme, norm,arg);
    }
            public static String lemma()
    {
return resultl;
    }
         private static String lemmatize(String[] args) throws FileNotFoundException,
          IOException, ClassNotFoundException
    {
         String scheme = "def";
        Boolean norm = false;
         String arg = args[0];
         Farasa nbt = new Farasa();
         int i, firstSuffixIndex;
         boolean lemmaFound, emptyLemmas;
         ArrayList<String> lemmas = new ArrayList<String>();
           String diacTokPOSLemma, diacTokPOSLemma2, uniqueUndiacLemmas,
        topSolution, stem, stem2, lastPrefix, firstSuffix, lineLemmas;
         String[] wordInfo, lemmaList, segments;
         String outlem="";
      if (arg != null)
      {
         // normalize Farsi letter
         arg = ArabicUtils.replaceFarsiCharacters(arg);
         ArrayList<String> words = tokenize(removeDiacritics(arg));
           lineLemmas = "";
              for (String w : words)
         {
              //lineLemmas += String.format("%s:", w);
               // Format of the file: Word      Diac     Tokenization   POS     Lemma
                        UniqueUndiacLemma
              diacTokPOSLemma = nbt.hmWordDiacTokPOSLemma.get(w);
              lemmaFound = false;
              emptyLemmas = false;
              if (diacTokPOSLemma != null)
              {
```

```
                    wordInfo = diacTokPOSLemma.split("\t");
                    if (wordInfo.length == 5)
                    {
                       uniqueUndiacLemmas = wordInfo[4];
                       lemmaList = uniqueUndiacLemmas.split("،");
                       if (lemmaList.length >= 1)
                       {
                          // Take the first lemma (ordered by frequemcy)
                          lineLemmas += String.format("%s ", lemmaList[0]);
                          lemmaFound = true;
                       }
                       else
                       {
                          emptyLemmas = true;
                       }
                    }
                 }
                   if (!lemmaFound)
                 {
        List<String> prefixes = Arrays.asList("ك", "س", "ف", "ال", "ل", "ب", "و"); // f, w, l, b,
k, Al, s
                    topSolution = w;
                    if (!nbt.hmSeenBefore.containsKey(w))
                    {
                       TreeMap<Double, String> solutions = nbt.mostLikelyPartition(w, 1);
                       topSolution = w;
                       if (solutions.size() > 0)
                       {
                          topSolution = solutions.get(solutions.firstKey());
                       }
                       topSolution = topSolution.replace(";", "").replace("++", "+");
                       nbt.hmSeenBefore.put(w, topSolution);
                       nbt.hmSeenBefore.put(w, topSolution);
                    }
                    else
                    {
```

```
topSolution = nbt.hmSeenBefore.get(w).replace(";", "").replace("++", "+");
 }
segments = topSolution.split("\\+");
if (segments.length == 1)
{
   lineLemmas += String.format("%s ", segments[0]);
}
else
{
   stem = "";
   lastPrefix = "";
   firstSuffix = "";
   firstSuffixIndex = segments.length;
   for (i = 0; i < segments.length; i++)
   {
      if (prefixes.contains(segments[i]))
      {
         lastPrefix = segments[i];
         continue;
      }
      stem = segments[i];
      if (i < segments.length - 1)
      {
         firstSuffix = segments[i + 1];
         firstSuffixIndex = i;
      }
      break;
   }
   if (stem.isEmpty())
   {
      stem = lastPrefix;
   }
    diacTokPOSLemma = null;
   if (!firstSuffix.isEmpty())
   {
```

```
    diacTokPOSLemma2 = null;
    stem2 = "";
    if ((firstSuffixIndex < segments.length - 1) &&
        firstSuffix.equals("ت"))
    {
       // Try taa marbouta first
       stem2 = String.format("%s%s", stem, "ة");
       diacTokPOSLemma2 =
        nbt.hmWordDiacTokPOSLemma.get(stem2);
    }
    if (diacTokPOSLemma2 == null)
    {
       stem2 = String.format("%s%s", stem, firstSuffix);
           diacTokPOSLemma2 =
      nbt.hmWordDiacTokPOSLemma.get(stem2);
    }
    if (diacTokPOSLemma2 != null)
    {
       diacTokPOSLemma = diacTokPOSLemma2;
       stem = stem2;
    }
}
  if (diacTokPOSLemma == null)
{
    diacTokPOSLemma = nbt.hmWordDiacTokPOSLemma.get(stem);
}
if (diacTokPOSLemma == null)
{
    if (stem.endsWith("ؤ") || stem.endsWith("ئ"))
    {
       stem2 = stem.substring(0, stem.length() - 1);
       stem2 += "ء";
       diacTokPOSLemma2 =
          nbt.hmWordDiacTokPOSLemma.get(stem2);
         if (diacTokPOSLemma2 != null)
```

```
                {
                    diacTokPOSLemma = diacTokPOSLemma2;
                    stem = stem2;
                }
            }
        }
        lemmaFound = false;
        emptyLemmas = false;
        if (diacTokPOSLemma != null)
        {
            wordInfo = diacTokPOSLemma.split("\t");
            if (wordInfo.length == 5)
            {
                uniqueUndiacLemmas = wordInfo[4];
                lemmaList = uniqueUndiacLemmas.split("ء");
                if (lemmaList.length >= 1)
                {
                    // Take the first lemma (ordered by frequemcy)
                    lineLemmas += String.format("%s ", lemmaList[0]);
                    lemmaFound = true;
                }
                else
                {
                    emptyLemmas = true;
                }
            }
        }
        else
        {
            lineLemmas += String.format("%s ", stem);
        }
    }
  }
}
  lineLemmas = lineLemmas.trim();
  if (norm)
  {
     lineLemmas = normalizeFull(lineLemmas);
```

```
            }
            lineLemmas += "\n";
            outlem =lineLemmas ;
            resultl =outlem;
        }
          return outlem ;
       }
}
```

II) FARASA Name of Entity Package

```
package com.qcri.farasa.ner;
import com.qcri.farasa.pos.FarasaPOSTagger;
import com.qcri.farasa.segmenter.Farasa;
import java.util.ArrayList;
public class ENMS
{
    public static String result="";
    public static void main(String[] args)throws  ClassNotFoundException,
  InterruptedException, Exception
    {
      String arg = args[0];
      EN(arg);
    }
      public static String ner()
    {
   return result;
    }

    public static void EN(String arg) throws  InterruptedException,
 ClassNotFoundException, Exception

    {
            Farasa segmenter = new Farasa();
   FarasaPOSTagger tagger = new FarasaPOSTagger(segmenter);
   ArabicNER ner = new ArabicNER(segmenter, tagger);
       String topSolution ="";
```

```
String out = "";
String line = arg;
if (arg != null)
{
ArrayList<String> output = ner.tagLine(line);
int loc = 0;
for (String s : output)
{
if (s.toLowerCase().contains("/O".toLowerCase()))
{
}
else{
String plusSign = " ";
if (loc == 0)
{
plusSign = "";
}
topSolution +=plusSign + s.trim();
loc++;
}
}
topSolution = topSolution.trim();
topSolution += "\n";
out=topSolution;
result=out;
} } }
```

2) Solr Search Server

As mentioned in the related work part, solr is an open-source search platform which is used to build search applications. It used a special database called "NoSQL Database". The NoSQL technique used the inverted index to store the data inside it. In this project, Solr used to index the Wikipedia document and search on it. The files of solr have been

configured and modify to add the data import handler that used to index a huge amount of data. Because Wikipedia has millions of documents, that can't be handled using normal database.

I) Data-config.xml File

That is used to configure data- import handler in index process. This file is adding by the developer, then identify the data source to handle the Wikipedia doc as following:

```
<dataConfig>
    <dataSource name="a" type="FileDataSource" encoding="UTF-8" />
    <dataSource name="b" type="FileDataSource" encoding="UTF-8" />
    <document>
    <entity name="page"
          dataSource="a"
          processor="XPathEntityProcessor"
          stream="true"
          forEach="/mediawiki/page/"
          url="C:/Users/HP/Desktop/wiki/arwiki.xml"
          transformer="RegexTransformer,DateFormatTransformer"  >
        <field column="title"     xpath="/mediawiki/page/title" />
        <field column="id"        xpath="/mediawiki/page/id" />
        <field column="text"      xpath="/mediawiki/page/revision/text"  />
        <field column="timestamp" xpath="/mediawiki/page/revision/timestamp"
          dateTimeFormat="yyyy-MM-dd'T'hh:mm:ss'Z'" />
    </entity>
    </document>
</dataConfig>
```

II) managed-schema File

Is used to identify fields, their type and how to process each type in index process. It modified by adding the fields of the Wikipedia with their types as following:

```
<field name="_root_" type="string" docValues="false" indexed="true" stored="false"/>
 <field name="_text_" type="text_general" multiValued="true" indexed="true" stored="false"/>
 <field name="_version_" type="plong" indexed="false" stored="false"/>
 <field name="id" type="string" multiValued="false" indexed="true" required="true" stored="true"/>
 <field name="text" type="text_ar" indexed="true" stored="true"/>
 <field name="timestamp" type="pdate" indexed="true" stored="true"/>
 <field name="title" type="string" indexed="true" stored="false"/>
 <field name="titleText" type="text_ar" indexed="true" stored="true"/>
```

Then add the Arabic field to recognize the Arabic Language as the following:

```
<fieldType name="text_ar" class="solr.TextField"
  positionIncrementGap="100">
 <analyzer>
  <tokenizer class="solr.StandardTokenizerFactory"/>
  <filter class="solr.LowerCaseFilterFactory"/>
  <filter class="solr.StopFilterFactory" words="lang/stopwords_ar.txt"
   ignoreCase="true"/>
  <filter class="solr.ArabicNormalizationFilterFactory"/>
  <filter class="solr.ArabicStemFilterFactory"/>
 </analyzer>
```

III) Solrconfig.xml File

As mentioned in related work part, this file contains the definitions and index-specific configurations related to request handling and response formatting, along with indexing, configuring, managing memory and making commits. Here just add the library of the data import handler and register it, as following:

```
<!-- Data import handler -->
 <requestHandler name="/dataimport"
 class="org.apache.solr.handler.dataimport.DataImportHandler">
  <lst name="defaults">
   <str name="config">data-config.xml</str>
```

```
    </lst>
  </requestHandler>
  <lib dir="../../../dist/" regex="solr-dataimporthandler-.*\.jar" />
```

Then add the Search options and query structure as following:

```
<requestHandler name="/browse" class="solr.SearchHandler"
useParams="query,facets,velocity,browse">
  <lst name="defaults">
    <str name="echoParams">explicit</str>
  <!-- Query settings -->
    <str name="defType">edismax</str>
  <str name="rows">10</str>
  <str name="fl">titleText,id,score,text</str>
    <str name="qf">titleText^10.0 text^5.0 </str>
  <str name="pf">titleText^10.0 text^5.0 </str>
  <str name="pf2">titleText^3  text^2 </str>
  <str name="pf3">titleText^4  text^3 </str>
    <str name="ps">5</str>
    <str name="ps2">5</str>
    <str name="ps3">5</str>
    <str name="qs">9</str>
    <!-- Highlighting defaults -->
    <str name="hl">on</str>
    <str name="hl.fl">text</str>
    <str name="hl.preserveMulti">true</str>
    <str name="hl.encoder">html</str>
    <str name="hl.simple.pre">&lt;b&gt;</str>
    <str name="hl.simple.post">&lt;/b&gt;</str>
    <str name="f.titleText.hl.fragsize">0</str>
    <str name="f.titleText.hl.snippets">1</str>
    <str name="f.text.hl.snippets">1</str>
    <str name="f.text.hl.fragsize">2</str>
    <str name="f.text.hl.maxAlternateFieldLength">750</str>
 </lst>
```

3) PHP/JAVA Bridge

Is used to communicate between PHP and JAVA languages. In this project it used for the communication between the PHP code (The website) and the Farasa Packages, the code as following:

```
<?php
mb_internal_encoding( 'UTF-8' );
header( 'Content-Type: text/html; charset=UTF-8' );
ini_set( "allow_url_include", 1 );
class farasaLib{
    /* Member functions */
    public function lemmization( $query ) {
        require_once( "http://localhost:8090/TestCase/java/Java.inc" );
        $session = java_session();
        $seglem = new java( "com.qcri.farasa.segmenter.TestCase" );
        $seglem->main( array( $query ) );
        return $seglem->lemma();
    }
    public function nameOfEntity( $query ) {
        require_once( "http://localhost:8090/ENMS/java/Java.inc" );
        $session = java_session();
        $en = new java( "com.qcri.farasa.ner.ENMS" );
        $en->main( array( $query ) );
        $entityname = $en->ner();
        $Entities = array();
        $tokencount = str_word_count( $entityname );
        $pieces = explode( " ", $entityname );
        for ( $i = 0; $i < $tokencount; $i++ ) {
            $Entities[$i] = $pieces[$i];
            $Entities[$i] = preg_replace("[-]", ' ',$Entities[$i]);
            $Entities[$i] = preg_replace("[/]", ' ',$Entities[$i]);
        }
        for ( $i = 0; $i < $tokencount ; $i++ ) {
            $append="";
            if ( strpos( $Entities[$i], 'I' ) == true ) {
```

```
                $Entities[$i] = preg_replace("/[a-zA-Z0-9]+/", ' ',$Entities[$i]);
                    $append .= $Entities[$i];
                    $Entities[$i]=null;
                    for ( $j = $i; $j < $tokencount - 1; $j++ ) {
                        if(strpos( $Entities[$j+1], 'I' ) == true ) {
                            $Entities[$j+1] = preg_replace("/[a-zA-Z0-
                9]+/", ' ',$Entities[$j+1]);
                            $append .= $Entities[$j+1];
                            $Entities[$j+1]=null;
                        }
                        elseif(strpos( $Entities[$j], 'B' ) == true )
                            break;
                    }
                    $Entities[$i-1] .= $append;
                }
                elseif(strpos( $Entities[$i], 'B' ) == true ){
                $Entities[$i] = preg_replace("/[a-zA-Z0-9]+/", ' ',$Entities[$i]);}
            } return $Entities;
        }
}
```

4) Solarium PHP/SOLR Library

Is a Solr client library for PHP. The code as following:

```
<?php  // create a client instance
$client = new Solarium\Client($config);
// get a select query instance
$query = $client->createSelect();
// get the dismax component and set a boost query
$dismax = $query->getDisMax();
$dismax->setqueryfields('titleText^10  text^5');
$dismax->setphrasefields('titleText^10  text^5');
$dismax->setphraseslop(5);
$dismax->setqueryphraseslop(9);
$edismax = $query->getEDisMax();
$edismax->setphrasebigramfields('titleText^3  text^2');
$edismax->setphrasetrigramfields('titleText^4  text^3');
```

```
$edismax->setphrasebigramslop(5);
$edismax->setphrasetrigramslop(5);
// this query is now a dismax query
$query->setQuery($_COOKIE["query"]);
$query->setFields(array('id','titleText', 'score'));
//$query->setStart(0)->setRows(10);
// this executes the query and returns the result
//$resultset = $client->select($query);
// display the total number of documents found by solr
        $perpage = 10;
        if ( isset( $_GET[ "page" ] ) ) {
                $page = intval( $_GET[ "page" ] );
        } else {
                $page = 1;
        }
        $calc = $perpage * $page;
        $start = $calc - $perpage;
        $query->setQuery($_COOKIE['query']);
        $query->setStart( $start )->setRows( 10 );
        // this executes the query and returns the result
        $resultset = $client->select( $query );
        $count=$resultset->getNumFound();
        if ( $resultset ) {
                // show documents using the resultset iterator
                foreach ( $resultset as $document ) {
                        echo '<hr/><table>';
                        // the documents are also iterable, to get all fields
                        foreach ( $document AS $field => $value ) {
                                // this converts multivalue fields to a comma-separated string
                                if ( is_array( $value ) )$value = implode( ', ', $value );
                                if ( $field == 'titleText' ) {
          echo "<a href='https://ar.wikipedia.org/wiki/" . $value . "'>" . $value . '<br/><a/>';
                } else echo '<tr><th>' . $field . '</th><td>' . $value . '</td></tr>';
                        }                       echo '</table>';                    } } ?>
```