

MA-004-525-1

Université de BLIDA 1
Faculté des Sciences
Département d'Informatique



MASTER THESIS

Option : Traitement Automatique de la Langue

CONTRIBUTION TO THE CONSTRUCTION OF AN
OPENSOURCE PLATFORM DEDICATED TO THE
SIMULATION OF NLP TOOLS FOR THE ARABIC LANGUAGE

By:

Zidane Nouredine

Zidoun Mohamed Abdelrahmane

In front of a jury composed of:

President: Mme TOUBALINE Nesrine

University of Blida 1

Examiner: Ms NASERI Ahlem

University of Blida 1

Supervisor: Mr HOCINI Hatem

Algerian Academy of Arabic Language

Supervisor: Ms YKHLEF Hadjer

University of Blida 1

Guest: Mme KHELOUT Fatiha

Algerian Academy of Arabic Language

MA-004-525-1

Année universitaire 2018/2019

Abstract

Arabic Natural Language Processing has quickly grown into an active field of research during the last decade. Working on the Arabic language is challenging due to its morphological richness and flexibility. Thereby, the processing of the Arabic language requires the development of appropriate tools. To this end, our primary goal is to construct an opensource platform dedicated to the simulation of natural language processing tools for the Arabic language. Our platform amalgamates various tools, which ensures their interoperability and their reusability.

Keywords: Arabic natural language processing (ANLP), Platforms, Interoperability, Reusability.

Résumé

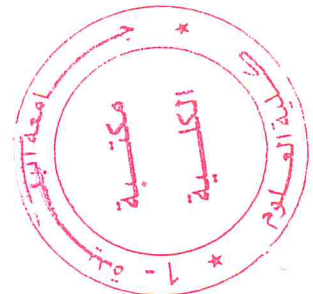
Le traitement automatique de la langue arabe est rapidement devenu un domaine de recherche actif au cours de la dernière décennie. Travailler sur la langue arabe est un défi en raison de sa richesse morphologique et de sa flexibilité. De ce fait, le traitement de la langue arabe nécessite le développement d'outils appropriés. À cette fin, notre objectif principal est de construire une plateforme opensource dédiée à la simulation d'outils de traitement du langage naturel pour la langue arabe. Notre plateforme regroupe divers outils, ce qui assure leur interopérabilité et leur réutilisabilité.

Mots clés : Traitement du langage naturel arabe, Plates-formes, Interopérabilité, Réutilisabilité.

ملخص

نمت معالجة اللغة العربية بسرعة لتصبح مجالاً نشطاً للبحث خلال العقد الماضي. العمل على اللغة العربية يمثل تحدياً بسبب ثرائه المورفولوجي ومرورته. وبالتالي، فإن معالجة اللغة العربية تتطلب تطوير الأدوات المناسبة. تحقيقاً لهذه الغاية، يتمثل هدفنا الأساسي في إنشاء منصة مخصصة لمحاكاة أدوات معالجة اللغة العربية. تجمع منصتنا بين الأدوات المختلفة، مما يضمن قابلية التشغيل البيئي وإعادة استخدامها.

كلمات المفاتيح: معالجة اللغة العربية، المنصات، التشغيل المتداخل، إعادة الاستخدام.



Acknowledgement

Above all, we thank Allah to have given us faith, strength and courage.

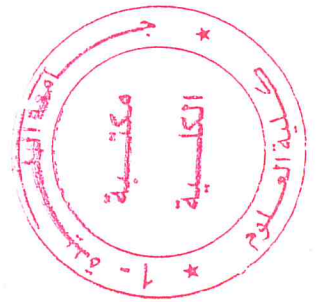
We wish to express our gratitude to all the people who helped us to accomplish this thesis:

First, we would like to express our gratitude to our supervisors, Mr. HOCINI Hatem for his guidance, advice and patience.

Many thanks to Ms. YKLEF Hadjer for the time and the effort she has dedicated for us, she has always been there to enlighten us the way and she did not hesitate with her constructive advice.

We also thank all the teachers who contributed in our education.

We express our gratitude to the board of examiners for reading and evaluating our work.



Contents

INTRODUCTION	1
PART I. CONCEPTS AND GENERALITIES	4
CHAPTER 1. THE GENERAL PRINCIPLES OF AUTOMATIC PROCESSING OF THE ARABIC	
LANGUAGE	5
1.1 Introduction.....	5
1.2 Particularity of the Arabic language	5
1.3 Arabic language grammar.....	6
1.4 Language Resources	7
1.5 Main tools of ANLP	9
1.6 Conclusion	12
CHAPTER 2. PLATFORMS AND TOOLS FOR AUTOMATIC PROCESSING OF THE ARABIC	
LANGUAGE	14
2.1 Introduction.....	14
2.2 NLP platforms.....	14
2.3 ANLP platforms.....	16
2.4 Criticism of the existing and problems of interoperability	23
2.5 For a standard and flexible solution dedicated to ANLP	23
2.6 Conclusion	24
PART II. STUDY, MODELING AND IMPLEMENTATION	25
CHAPTER 3. DESIGN OF A FLEXIBLE ARCHITECTURE FOR ANLP TOOLS PLATFORM.....	26
3.1 Introduction.....	26
3.2 Architecture of the platform	26
3.3 Interoperability & reusability between tools	38
3.4 Conclusion	41
CHAPTER 4. IMPLEMENTATION	42
4.1 Introduction.....	42
4.2 Development environment.....	42
4.3 Description of AAAL platform.....	43

4.4 Flow	44
4.5 Outputs:.....	54
4.6 Conclusion	55
CONCLUSION	56
REFERENCES.....	57
APPENDIX.....	64

List of Figures

FIGURE 1.1: DERIVATION OF THE ROOT كَتَب [13].	7
FIGURE 1.2: EXAMPLE OF AN ARABIC MORPHOLOGICAL ANALYZER [17].	9
FIGURE 1.3: AN EXAMPLE OF ARABIC SENTENCE PARSED USING STANFORD PARSER [29].	11
FIGURE 2.1: MADAMIRA ARCHITECTURE [9].	17
FIGURE 2.2: THE SUITE OF MADA AND THE SUITE OF AMIRA [40].	18
FIGURE 2.3: ARCHITECTURE OF SAFAR PLATFORM [42].	20
FIGURE 2.4: TYPICAL PROCESSING PIPELINE OF ARANLP [12].	22
FIGURE 3.1: GLOBAL ARCHITECTURE OF THE AAAL PLATFORM.	27
FIGURE 3.2: PROCESS OF SEGMENTING MODULE.	29
FIGURE 3.3: PROCESS OF STEMMING MODULE.	31
FIGURE 3.4: PROCESS OF POS TAGGING MODULE.	33
FIGURE 3.5: PROCESS OF PARSING MODULE.	35
FIGURE 3.6: PROCESS OF NER MODULE.	36
FIGURE 3.7: PROCESS OF THE MID-VOCALIZATION.	38
FIGURE 4.1: MAIN WINDOW IN SIMPLE MODE.	45
FIGURE 4.2: MAIN WINDOW IN ADVANCED MODE.	46
FIGURE 4.3: TOKENIZATION MODULE.	47
FIGURE 4.4: SEGMENTATION MODULE.	48
FIGURE 4.5: STEMMING MODULE.	49
FIGURE 4.6: POS TAGGING MODULE.	50
FIGURE 4.7: PARSING MODULE.	51
FIGURE 4.8: NER MODULE.	52
FIGURE 4.9: NUMBERS FUNCTIONS.	53
FIGURE 4.10: VOCALISATION MODULE.	54
FIGURE 4.11: OUTPUT FILES.	54

List of Tables

TABLE 2.1: SAFAR PLATFORM INTEGRATED TOOLS [43].....21

TABLE 3.1: EXAMPLES AND RESULTS [50].....37

TABLE 3.2: COMPARISON BETWEEN TOOLS BEFORE AND AFTER THEIR INTEGRATION IN OUR
PLATFORM.40

INTRODUCTION

1. Context and aim of the project

Natural Language Processing (NLP) has gained increasing attention during the last decade [1]. It is widely acknowledged that the construction of a flexible platform dedicated mainly to the NLP is highly needed. Moreover, these platforms can be merged within other frameworks in order to produce high-level applications such as machine translation [2], text classification [3], automatic summarization [4] and question answering [5].

Our project is part of the work related to the Arabic Natural Language Processing (ANLP) field. The primary goal is to construct **an opensource platform dedicated to the simulation of NLP tools for the Arabic language**. We aim at addressing two major flaws:

- ◆ Working on the Arabic language is challenging due to its morphological richness and flexibility. Thereby, the processing of the Arabic language requires the development of appropriate tools dedicated to ANLP. Nowadays, many tools for ANLP have been developed; for instance: morphological analyzers, syntactic parsers, search engines, machine translation systems, etc. Meanwhile, the development of ANLP applications requires the use of several tools at once, each one of them dealing with a different level of language (morphology, syntax, semantics and pragmatics). However, *these tools are not always encapsulated in homogeneous and interoperable architectures*. Also, *the output of one tool is not directly exploitable by another tool*. It is mainly due to the individuality of researchers work. Their main purpose is the result and not the interoperability, the reusability and the shareability of the integrated tools.
- ◆ Many platforms have been constructed in the field of NLP such as GATE [6], NLTK [7] and OpenNLP [8]. These platforms are more or less mature for the Latin languages. Furthermore, they cannot be easily adapted for the Arabic language. To this end, the ANLP community has built some platforms that are dedicated mainly to the Arabic language such as MADAMIRA [9], ATKS [10], SAFAR [11] and AraNLP [12]. However, *these platforms do not contribute in*

solving the problems of interoperability and reusability except for SAFAR platform, which was developed recently and it still suffers from many problems [11].

In order to address the aforementioned limitations, we have developed a **flexible platform that gathers ANLP tools leaning on the aspects of interoperability and reusability**. Our platform guarantees:

- The reuse of the integrated tools in other tasks such as the segmenting task that was used as preprocessor of the parsing task.
- The improvement of the result obtained from tasks like the Named Entity Recognition (NER) by adding a new category of named entities. We have also added a backup lists in order to give the ability to developers to integrate new named entities.

2. Structure of the thesis

The reminder of this thesis is structured into two parts.

Part one: Concepts and Generalities

This part introduces the field of ANLP and its principles. It also includes a description of the existing platforms in both fields NLP and ANLP.

Chapter 1: The general principles of the automatic processing of the Arabic language

In this chapter, we first discuss the particularity of the Arabic language; then we present the grammatical levels of it and the resources of the language in the NLP field. Finally, we define some of the main ANLP tools that are widely used by developers.

Chapter 2: Platforms and tools for automatic processing of the Arabic language

In the second chapter, we describe existing platforms of NLP and ANLP, presenting their architectures if available, mentioning their composed modules. Then, we criticize existing platforms in ANLP field. Based on our analysis, we propose a solution for a flexible ANLP platform.

Part two: Study, conception and implementation

In this part, we thoroughly present our work and contributions in this part.

Chapter 3: Design of a flexible architecture for ANLP tools platform

This chapter presents the conception of our platform. We first introduce our architecture and we describe the modules integrated in it. We also illustrate our work with examples and schemes of each module. Finally, we provide a comparative study among the integrated tools for the aspects of interoperability and reusability.

Chapter 4: Implementation

In this chapter, we present the development environment and the libraries used to implement each module. Then, we describe the progress steps of processing in our platform and illustrate it with screenshots.

Conclusion and perspectives

We synthesize the construction of our platform and integrated modules. We also offer perspectives. A user manual is provided at the end as an appendix.

**PART I. CONCEPTS AND
GENERALITIES**

Chapter 1. The general principles of automatic processing of the Arabic language

1.1 Introduction

Natural Language Processing is a crucial part in the field of artificial intelligence. Its main purpose is to make the natural language understandable to the machine. The machine can not understand the language as a human being. However nowadays technologies allow to developers to extract many available information from many resources using programs to build platforms and libraries that save a lot of time and effort.

The Arabic language is one of the most used languages (nearly 422 million person). Therefore, it is interesting and challenging because it is rich linguistically and completely different from occidental languages, also due to the region it occupies. It splits up to classic Arabic and modern Arabic (MSA). The modern Arabic comes from the classic Arabic and it is used in many fields such as the press, scientific texts, political debates... etc.

1.2 Particularity of the Arabic language

The increasing development of the Arabic language in the field of NLP has led to the emergence of different tools at all levels of the language (morphological, syntactical, semantical). The Arabic language is one of the toughest and most complex languages because of many challenges, such as the agglutination in Arabic and dispensability of vowel diacritics. Also, the Arabic language is written from right to left.

The Arabic alphabet contains 28 letter and all of those letters are consonants and each consonant has a different pronunciation. Most of its letters change their form depending on its position in the word. Contrary to the French or English languages the vowels in Arabic are not letters; they are diacritics sings.

In Arabic language the sentence does not have a regular form like in English or in French. For example, a verb on its own can be considered as a sentence **سمعتك** (I heard you). Moreover, Arabic does not have capital letters. Many ambiguities such as Derivational ambiguity, Inflectional ambiguity, Morphological ambiguity. Therefore, the Arabic is a rich and a complex morphological language.

1.3 Arabic language grammar

The traditional grammar of the Arabic language includes two categories of rules, syntax and morphology. The latter is divided into:

- **Flexional morphology:** which deals with variations of morpho-phonological forms and not with variations of meaning.
- **Derivational morphology:** with the root-and-schema model, where the root provides a general abstract meaning and the schema assigns the grammatical category simultaneously with functional and semantic features

1.3.1 Morphology:

Morphology is the study, identification, analysis and description of the minimal meaning bearing units that constitute a word. The minimal meaning bearing unit of a word is called a morpheme.

- **The root:** Roots are at the origin of most Arabic words; it is the basis of all forms of verbs and certain Arabic names as it shown in Figure 1.1.
- **The scheme:** It is a predefined form that characterizes a class of verbs or nouns: verbal schemas and nominal schemes. Schemes are models with different structures that are applied to the root to create a word.
- **The lemma:** It can be analyzed as a root inserted into a schema. It is the intersection between a graphic form and a meaning. The knowledge of the lemma or the couple (root, schema) makes it possible to deduce the different inflected forms of a verb or a noun.

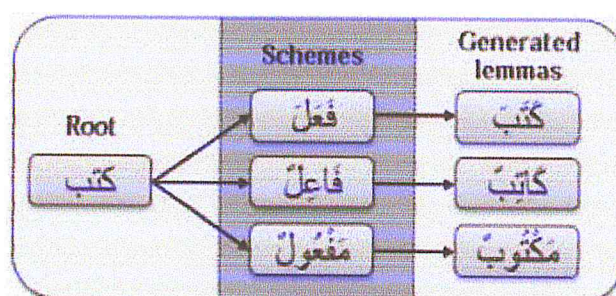


Figure 1.1: Derivation of the root كُتِبَ [13].

1.3.2 Syntax

Syntax is the linguistic discipline interested in modeling how words are arranged together to make larger sequences in a language. It describes the proper order of the words to make phrases and sentences [14]. In Arabic, there are two types of sentences:

- Verbal sentences: It has three elements which are the verb, the subject and the object.
- Nominal sentence: It is a sentence with no verb in it (verbless sentence).

1.3.3 Semantic

Semantics is the study of the meaning of linguistic expressions [14]. It is used to study the changes in the meaning by analyzing the linguistic structure (phonetically, morphologically, lexically and syntactically).

The morphological richness of the Arabic language leads to more ambiguity than the other languages such as English. As the other languages, Arabic contains the basic concepts of synonymy, homonymy and semantic roles [14].

A first level of modeling consists of constituting classes of words (semantic categories). These classes include words whose meaning is similar, or at least (for general classes) words that have some common semantic properties.

1.4 Language Resources

Language Resources play a vital role in the applications of languages since they feed the different processes of NLP systems. We can classify them into two categories:

1.4.1 Lexicons

A lexicon is a collection of information about the words of a language, about the lexical categories to which they belong. In practice, a lexical entry can include further information about the roles the word plays, such as feature information; for example, whether a verb is transitive, intransitive, ditransitive, etc., what form the verb takes (e.g. present participle, or past tense, etc.).

A Monolingual lexicon

A monolingual dictionary explains the meaning of a word in the language that you are learning.

B Multilingual Lexicons

There are two types of multilingual lexicons: Those who are interested in matching two languages, often for a specific purpose (bilingual lexicons), and those whose more ambitious goal is to develop a generic mechanism that allows the parallelization of information lexical for a seemingly arbitrary number of languages.

1.4.2 Corpus

A corpus is a very large collection of text (often many billions of words) produced by real users of the language and used to analyze how words, phrases and language are used in general. It is used by linguists, lexicographers, social scientist, experts in NLP and in many other fields. It is also used for generating various language databases employed in software development such as predictive keyboards, spell check, grammar correction, text/speech understanding systems, text-to-speech modules and many others.

A Corpus categories

Many categories of the corpus are used in NLP field and there are some of the most used corpora [15]:

- **Monolingual corpus:** It is the most frequent type of corpus. It contains texts in one language only. The corpus is usually tagged for parts of speech and is used by a wide range of users for various tasks from highly practical ones, e.g. checking the correct usage of a word or looking up the most natural word combinations.

- **Multilingual corpus:** It contains texts in several languages, which are all translations of the same text and are aligned in the same way as parallel corpora.
- **Learner corpus:** It is a corpus of texts produced by learners of a language. The corpus is used to study the mistakes and problems that learners have when learning a foreign language.
- **Specialized:** It contains texts limited to one or more subject areas, domains, topics etc. Such corpus is used to study how the specialized language is used.

1.5 Main tools of ANLP

1.5.1 Morphological analyzers

Morphological analyzers are preprocessors for text analysis. Many Text Analytics applications invoke them to perform their tasks [16]. It is a group of methods that share the same structure. The purpose of morphological analyzers is to produce and develop tools and resources that expand the area of the Arabic word structure analysis, particularly, morphological analysis as it shown in Figure 1.2.

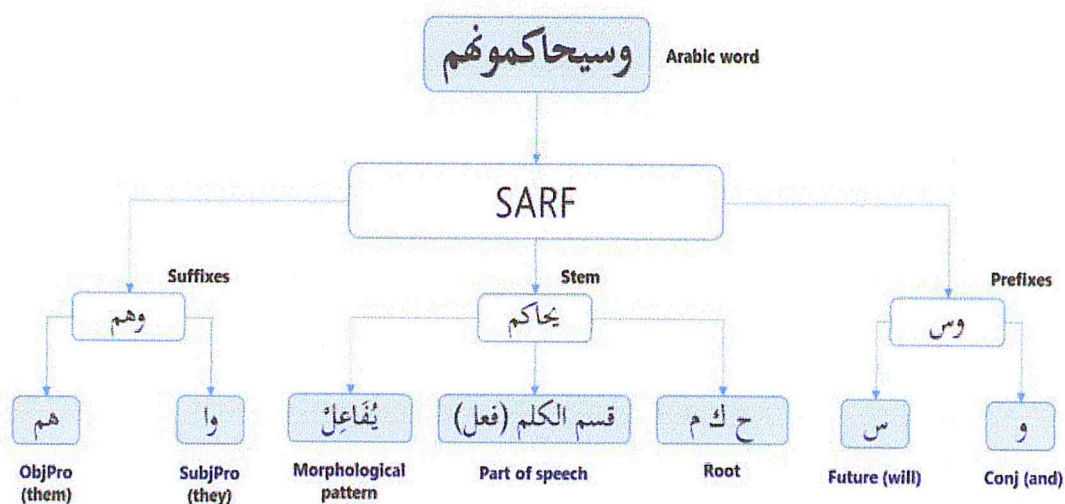


Figure 1.2: Example of an Arabic morphological analyzer [17].

A Arabic morphological analyzers

- Buckwalter Arabic Morphological Analyzer (BAMA): works on standard Arabic and English, applied in NLP, machine translation and information retrieval [18].
- Arabic Lexeme-based Morphological Generation and Analysis (AlMORGEANA) [19].
- Xerox Arabic: morphological analysis and generation, it accepts typed Modern Standard Arabic words and returns morphological analyses [20].
- Functional Arabic Morphology (ElixirFM) [21].

Morphological Analysis and Generation for Arabic and its Dialects (MAGEAD) performs an on-line analysis or generation from a root+pattern+features representation. It has separate phonological and orthographic representations [22].

1.5.2 Stemming

Stemming is the process of producing morphological variants of a root/base word. Also, it is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and NLP. It is a part of linguistic studies in morphology, artificial intelligence, information retrieval and extraction.

A Arabic stemmers

- Khoja Arabic stemmer [23].
- Sebawai & Al-Stem [24].
- Tashaphyne [25].

1.5.3 POS tagging

POS tagging is the process of selecting the most likely sequence of **syntactic categories** for the words in a sentence. It determines grammatical characteristics of the words, such as part of speech, grammatical number, gender, person, etc.

A Arabic part of speech taggers

- **Arabic Part-of-speech Tagger (APT)**: It was developed using a combination of both statistical and rule-based techniques. It is widely acknowledged that hybrid taggers produce the highest accuracy rates [26].
- **Stanford Log-linear Part-Of-Speech Tagger**: This software is a Java implementation of the log-linear part-of-speech taggers [27].
- **Toolkit for POS tagging “AMIRA”**: The POS tagging system optionally produces the PATB standard tag set of 25 tags or uses the extended set that has 72 tags. The user has the flexibility to request tokenized or non-tokenized POS tagged output [28].

1.5.4 Parsing

Parsing is the automatic analysis of a sentence with respect to its syntactic structure. Given a CFG (context-free grammar) as it shown in Figure 1.3. It consists of deriving a phrase structure tree assigned to the sentence by the grammar with ambiguous grammars. Each sentence may have many valid parse trees.

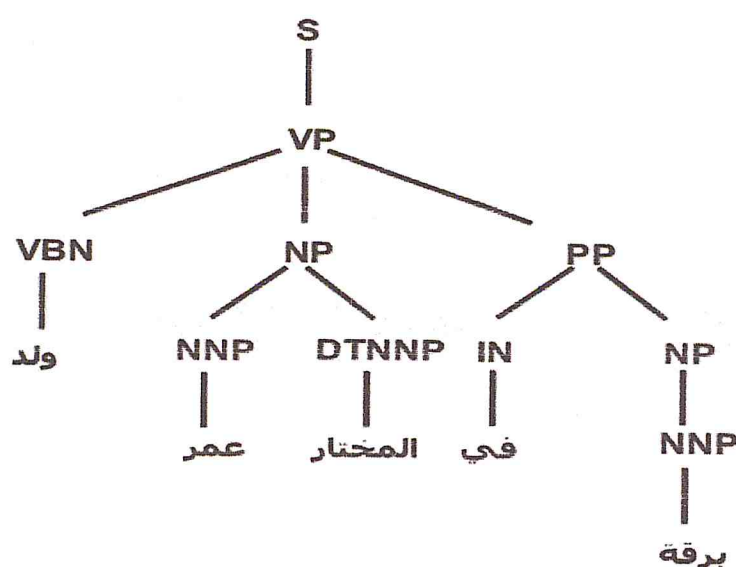


Figure 1.3: An example of Arabic sentence parsed using Stanford parser [29].

B Arabic parsers

- **The Stanford parser:** A natural language parser is a program that works out the grammatical structure of sentences. For instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences [30].
- **MaltParser:** It is a system for data-driven dependency parsing, which can be used to induce a parsing model from treebank data and to parse new data using an induced model [31].
- **FARASA dependency & constituency parsers:** FARASA (which means "insight" in Arabic), is a fast and accurate text processing toolkit for Arabic text. It consists of the segmentation/tokenization module, POS tagger, Arabic text Diacritizer, and Dependency Parser [32].

1.5.5 Tokenization

A tokenizer divides text into a sequence of tokens, which roughly correspond to "words". It is a preprocessing step for many applications in the field of NLP. It is helpful for many applications such as language modeling (LM) and information retrieval (IR).

A Arabic tokenizers

- **Pyarabic:** It is a specific Arabic language library for Python. It provides basic functions to manipulate Arabic letters and text [33].
- **NLTK word tokenizer** [7].
- **Polyglot tokenizer** [34].

1.6 Conclusion

In this chapter, we have first described the field of Arabic Natural Language Processing, giving some generalities of the Arabic language. In addition, we have mentioned the particularity of the Arabic language and its challenges in the field of ANLP. Then we have presented about the Arabic language grammar and have defined its main areas (morphology, syntax and semantic). We

have mentioned the language resources needed in NLP such as corpora and lexicons. Finally, we have reviewed some ANLP tools; for each tool we have given a definition and illustrated their use through figures.

Chapter 2. Platforms and tools for automatic processing of the Arabic language

2.1 Introduction

Working on NLP nowadays is a big challenge especially for Arabic language. This is mainly due to diversity of the domain tools in different levels such as developing languages, manipulated inputs or outputs, internal and external representations of results. This leads to a lot of complications in interoperability between these different tools and their reusability in new contexts. The main objective is to propose a solution to standardise the whole aspects shared by the processing tools of Arabic language. Moreover, the solution should guarantee different services with better flexibility, and address the problematic of interoperability and reusability. The solution combines several tools and platforms such as NLTK, OpenNLP, GATE...

2.2 NLP platforms

Various NLP tools and platforms have been proposed in the literature. The following sections summarize the most relevant ones.

2.2.1 NLTK:

NLTK is a platform for building Python programs to work with human language data. It provides corpora and lexical resources such as WordNet, also libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning [35].

It adapts linguists, engineers, students, educators and researchers [36].

NLTK ensures [37]:

Simplicity: To provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data.

Consistency: To provide a uniform framework with consistent interfaces and data structures, and easily-guessable method names.

Extensibility: To provide a structure into which new software modules can be easily accommodated, including alternative implementations and competing approaches to the same task.

Modularity: To provide components that can be used independently without needing to understand the rest of the toolkit.

NLTK provides many modules such as: Parsing, tagging, Finite State Automata, Type Checking, Type Checking and Text Classification [35].

2.2.2 GATE

General Architecture for Text Engineering is an open source java suite of tools which provides many natural languages processing tasks. It includes a community for students, developers and researchers.

Languages currently handled in GATE include English, Chinese, Arabic, Bulgarian, French, German, Hindi, Italian, Cebuano, Romanian, Russian, and Danish [6]. It is stable, robust, and scalable infrastructure for Natural Language Engineering.

GATE components are one of three types [38]:

- Language Resources (LRs) represent entities such as lexicons, corpora or ontologies.
- Processing Resources (PRs) represent entities that are primarily algorithmic, such as parsers, generators or ngram modellers.
- Visual Resources (VRs) represent visualization and editing components that participate in GUIs.

A Processing Resources

GATE provides the following processing resources [38]:

- ANNIE
- The tokenizer
- The sentence splitter
- The tagger

- The gazetteer
- The semantic tagger
- The orthomatcher
- The coreferencer

2.2.3 OpenNLP

OpenNLP is a natural language processing toolkit based on machine learning. It offers many NLP tasks such as language detection, tokenization, sentence segmentation ... etc. [8].

The absence of a global architecture is the major issue of the OpenNLP toolkit. In addition, the absence of modules that handle the Arabic language [8].

OpenNLP provides many components such as: sentence detector, tokenizer, name finder, part-of-speech tagger, chunker, parser... etc.

A OpenNLP tasks

It provides many tasks that deals with the natural language:

- Named Entity Recognition
- Summarize
- Searching
- Tagging (POS)
- Feedback Analysis
- Translation

2.3 ANLP platforms

Even that there is a lack in the field of NLP in dealing with the Arabic language, there are some ANLP platforms that have been developed.

2.3.1 MADAMIRA

MADAMIRA is a java toolkit for morphological analysis and disambiguation of Arabic and its dialects. It combines two Arabic processing systems **MADA** and **AMIRA** [9].

MADAMIRA uses machine learning algorithms to select the linguistic features. It provides seven tasks of NLP as it shown in Figure 2.1: Tokenization, morphological disambiguation, Part-of-

Speech tagging, lemmatization, diacritization, named entity recognition and base phrase chunking.

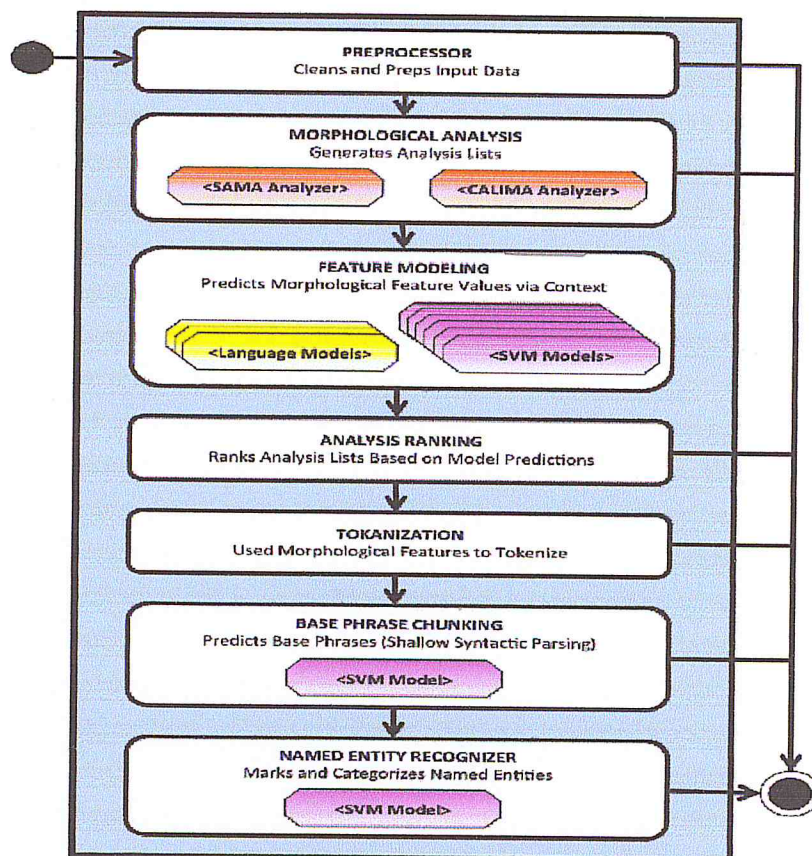


Figure 2.1: MADAMIRA Architecture [9].

B MADA

Is a system for Morphological Analysis and Disambiguation for Arabic. Given raw Arabic text [39], MADA adds morphological and lexical information by disambiguating in one operation. TOKAN is a general tokenizer dedicated to the Arabic. It tokenizes the MADA disambiguated text.

C AMIRA

Is a set of tools that includes a tokenizer, a part of speech tagger (POS) and a base phrase chunker (BPC) and a light syntactic parser [39] as it shown in Figure 2.2. Contrarily to MADA, AMIRA tools use a unified framework that classify every constituent problem. It includes two functionalities:

- a. **AMIRA-POS** works through an SVM based classification approach using the character n-grams as feature, the POS tag set has 72 tags [40] and it is the set of tags that the AMIRA uses as a first step. It provides the flexibility at the input level, either input a raw or a tokenized text.
- b. **AMIRA-TOK** learns clitic tokenization generalizations from the clitic segmentations present in the Penn Arabic Treebank (PATB) [39]. It segments off many clitics such as (conjunction proclitics +و w+, +ف f+, prepositional proclitics +ك k+, +ل l+, + b+, future marker proclitic +س s+, verbal particle proclitic +ل l+... etc. [39].
- c. **AMIRA-BPC** (Base Phase Chunking) forms syntactic phrases (NP, VP or PP) and it is appropriate for the Arabic language. It recognizes nine types of chunked phrases using a phrase IOB tagging scheme (Inside, Outside and Beginning of the phrase) [40]. Also, it accepts any level of pretreatment on the input text and produce BPC tags on raw input text or the form of AMIRA-TOK consistent schemes [40].

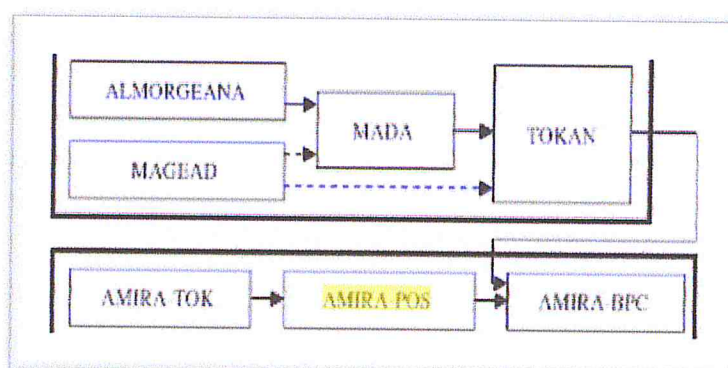


Figure 2.2: The suite of MADA and the suite of AMIRA [40].

2.3.2 ATKS (Arabic Toolkit Service)

ATKS provides a set of APIs for basic processing of the Arabic written language. ATKS was designed to be useful for the Arabic developer by offering many NLP APIs. It was proposed by Microsoft; all its components are available for academic use through a web service [41].

A ATKS components

The ATKS offers reliable and quality components for Arabic NLP researchers [10]:

- ❖ **Colloquial to Arabic Converter** provides translation of Egyptian colloquial text into modern standard Arabic along with rich mapping information.

- ❖ **Diacritizer:** The automatic Diacritizer component performs vowel restoration on input Arabic text.
- ❖ **Named Entity Recognizer (NER)** Detects and classifies named entities for persons, locations and organizations categories.
- ❖ **Arabic Parser** determines the grammatical structure of Arabic sentences. The Parser relies heavily on the Part-of-Speech (POS) Tagger and the Named-Entity Recognizer to identify the correct part of speech and to identify the entities.
- ❖ **Part of Speech Tagger** identifies the correct part of speech. It resolves the ambiguity on both the stem and the case-ending levels.
- ❖ **SARF (Morphological Analyzer)** provides all possible morphological analyses for an input Arabic word. It provides all possible morphological analyses for any given input Arabic word.
- ❖ **Speller** Detects and corrects misspelled words, provides correction candidates. Also, improves the accuracy of Arabic text processing components.
- ❖ **Transliterator** converts text from Romanized Arabic (Arabic written in English characters) to native Arabic script and vice versa. A common example of that is the transliteration of named entities.

2.3.3 SAFAR (Software Architecture for Arabic language pRocessing)

It is a Java-based framework dedicated to ANLP. It is open source, portable, modular, extensible, flexible and offers an integrated development environment (IDE) [42] as it shown in Figure 2.3.

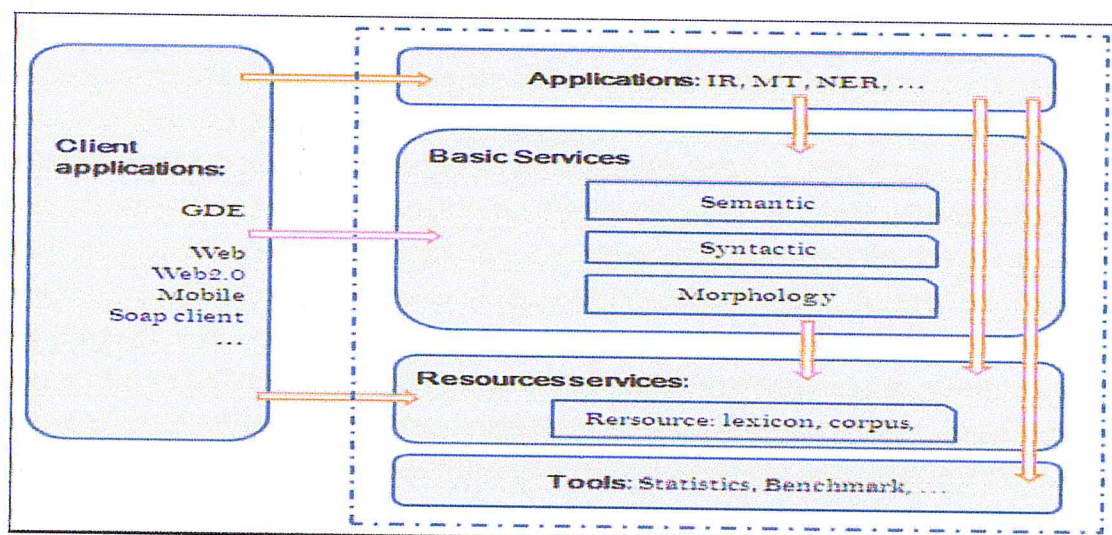


Figure 2.3: Architecture of SAFAR platform [42].

The utility of these layers can be summarized as follows [43]:

- *The client application layer* contains the applications that use the service of the other layers.
- *The applications layer* contains a high-level that used the layers listed above.
- *The basic service layer* contains the main three regular layers for processing the language.
- *The resource services layer* provides the necessary resources such as corpora and lexicon.
- *The tools layer* includes a set of technical services.

A Tools in SAFAR platform

SAFAR platform provides many tools in one layer and the user can call any method that suits his input text. We cite below some the main tools used in SAFAR [11] as it shown in Table 2.1:

- **Stemmers:**
SAFAR provides the following stemmers: ISRI, Khoja, Light10, Tashaphyne, and Motaz stemmer.
- **Morphological analyzers:**
From many morphological analyzers SAFAR provides: Alkhalil, BAMA and MADAMIRA morphological analyzer.
- **Syntactic parsers:**

SAFAR provides the most common parser: Stanford Parser.

- Applications:
Stem Counter, Sentence Processor, Morpho-Syntactic Processor, Summarizer, Moajam Tafaoli and Moajam Moaassir.
- Utilities:
Normalization, Sentence splitter, Tokenization, Transliteration and Benchmark.
- Resources:
Particles lexicon, Al wassit dictionary, Contemporary dictionary and Ontology.

Table 2.1: SAFAR platform integrated tools [43].

SAFAR layers	Type	Name of the tool
Morphology	Morphological analyzers	Alkhalil, BAMA
	Stemmers	Khoja, Light10, ISRI, Motaz stemmer, Tashaphyne
Syntax	Syntactic parsers	Stanford Parser
Utilities	Utilities	SAFAR Normalizer, SAFAR Sentence Splitter, SAFAR Tokenizer, SAFAR Transliterator, SAFAR Benchmark
Applications	Applications	Sentence processor, Stem counter, Morphosyntactic processor, Q/A application [12]

2.3.4 AraNLP

It is a java-based toolkit for the processing of Arabic text. AraNLP brings together most of the vital Arabic text preprocessing tools into one single library that can be accessed easily. It includes a sentence detector, tokenizer, light stemmer, root-based stemmer, part-of-speech tagger (POS-tagger), word segmenter, normalizer and diacritic remover [12] as it shown in Figure 2.4.

Those tools are developed as java classes in the AraNLP either they were existed and integrated or developed from the scratch.

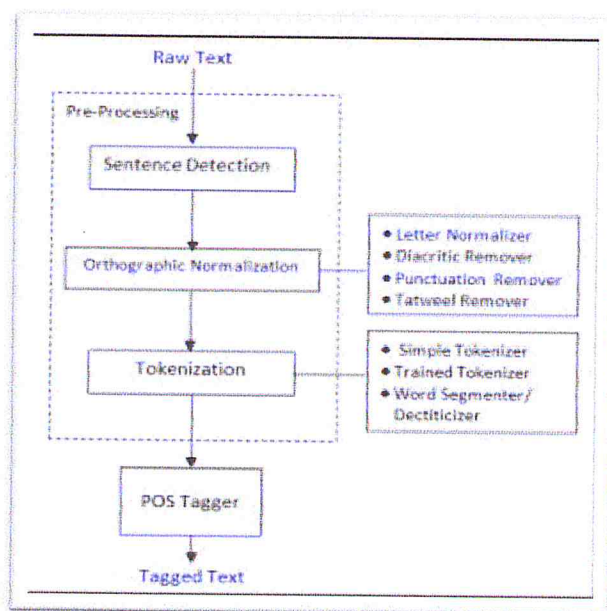


Figure 2.4: Typical processing pipeline of AraNLP [12].

A AraNLP modules

It provides the following modules [12]:

- Tokenization: the developers built a model that detects token boundaries using MaxEnt machine learning, basing on a training corpus.
- Sentence Boundary Detection: the model was built by the developers of AraNLP because of the lack of Arabic sentence separators.
- Stemming: AraNLP support both light stemmers and root stemmers in order to take in all the potential.
- Word Segmentation & POS Tagging: the AraNLP links up to the Stanford Arabic word segmenter and POS tagger.
- Arabic Normalization: it removes all the diacritics (Vowel Diacritics, Nunation Diacritics and the Shadda) also, it removes the punctuation.

2.4 Criticism of the existing and problems of interoperability

The different tools and platforms that we have seen above have proven their efficiency in the field of NLP. Except that in dealing with the Arabic language these platforms show a big lack, aside from the platforms that are dedicated mainly to ANLP. The main purpose is to improve these platforms in order to have tools that are interoperable and reusable to apply them on the Arabic language.

In general, these tools do not offer an architecture for the Arabic language which makes the integration of new tools harder. The ANLP community has not offered any platform similar to the platforms mentioned such as GATE, OpenNLP, NLTK...etc. The same problem with tools. Till now, ANLP community did not developed any promising tools that can solve the problem of interoperability and reusability. This latter makes the cohabitation of tools even harder because of the disconnection of the different developers who work in separated laboratories and using different techniques, different platforms, different languages.

These limitations make the development of ANLP tougher but knowing the problem is half of the solution. Leaning on these limitations ANLP community must develop proper platforms for the Arabic language. Lately there are some platforms that have seen the light and they are looking very promising. Many tools have been integrated in these platforms and they are dedicated to the Arabic language.

The problem of interoperability and reusability between the tools is the main challenge that all developers of the ANLP community must work on in order to save time and work with the tools that are already developed. Platforms such as GATE, NLTK and OpenNLP do not show any problem of interoperability between their tools because the tools in these platforms are homogeneous, which makes the creation of a complex application easier.

The Arabic language is one of the challenges that the developers have faced because it is a rich language comparing to the Latin languages. Also, these platforms do not cover the need of the ANLP community because they do not integrate modules and resources for ANLP.

2.5 For a standard and flexible solution dedicated to ANLP

The aim of this work is to develop a platform that covers these points:

- Open source and multi-platform
- Flexibility
- Opening, to integrate new modules that proven their efficiency
- Extensibility, to develop new applications according to the need
- Perfect match with the nature of the Arabic language and its constraints
- Diversity at the output level (XML, HTML, CVS etc.)

The existence of a platform for the ANLP is highly needed for the standardization and to solve the problems of reusability, of interoperability and the integration of all the tools of development in the field.

Nowadays, we have seen some evolution in the field of ANLP with the appearance of some platforms that provide some of the tools that are used in the platforms mentioned before and deal with Arabic language and its complexity.

2.6 Conclusion

In this chapter, we have described some of the existing platforms such as NLTK, GATE and the platforms that are dedicated to the Arabic language such as SAFAR, AraNLP and ATKS. We have also represented the architecture platforms if it's available by mentioning their components, tools and applications. Then we have given a criticism about the existing and about the problems of interoperability and reusability between tools. In addition, we have mentioned the limitation that faced the ANLP community. The Arabic language is a very complex and ambiguous language as we have mentioned above. It rises a big challenge ahead of us and ahead any developer in the field of ANLP. Finally, we have introduced a standard and a flexible solution for a platform that will may solved some of the ANLP community problems.

**PART II. STUDY, MODELING
AND IMPLEMENTATION**

Chapter 3. Design of a flexible architecture for ANLP tools platform

3.1 Introduction

NLP platforms provides many functionalities in dealing with the language by integrating the necessary tools that suits the architecture of the platform. The main goal of our work is to build a flexible platform combining numerous ANLP tools and different resources. The proposed platform must be interoperable and reusable in order to get a homogeneous structure. In addition, it must deal with different types of inputs.

In order to show the difference of tools features before and after integrating them in our platform, we have done a comparative study between the ANLP tools for the aspects: interoperability and reusability by defining these aspects. Then, we illustrate the comparative study with tables.

3.2 Architecture of the platform

The combination of these tools will produce high-level applications such as machine translation, named entity recognition and sentiment analysis that will be presented in a global architecture. The platform will be used by researchers and developers in the field of NLP and it will be open sourced with the intention of doing more improvements on the platform. Figure 3.1 shows the global architecture of the AAAL platform.

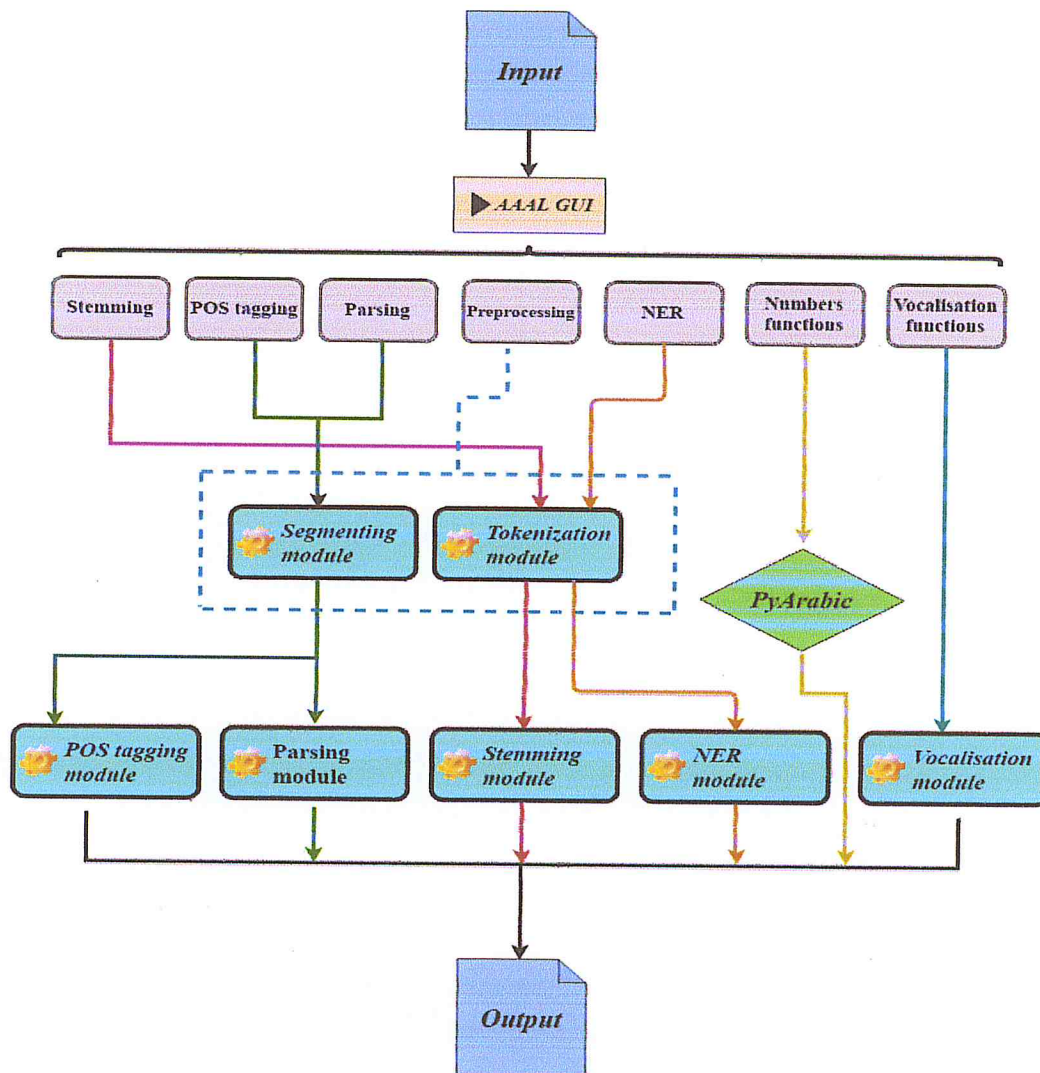


Figure 3.1: Global architecture of the AAAL platform.

3.2.2 Inputs

Our platform deals with several types of resources and gives many choices to users and developers to process different types of files with no need to convert the files each time. It processes Txt files, HTML (HyperText Markup Language) files, XML (Extensible Markup Language) files. In addition, it allows processing input text manually set by the end user. It also provides advanced tools available only when processing input files.

3.2.3 Tools

A Preprocessing

It is the first step for many applications. Our platform provides the following preprocessing modules:

- **Tokenization module** is the process that identifies the text boundaries of words and sentences. We have used **Polyglot** tokenizer [34] that basically splits the input text into sentences or words. It also gives the morphemes of one only word as an input. *The advantage of our platform is that the user is able to get list of morphemes from not only one word but any type of resource mentioned above.* Therefore, we have reused the tool of Polyglot tokenizer in combination with Polyglot morphology (word morphemes).
- **Stop words removal** is the process that verify and remove the useless words in the text. We have used the NLTK Arabic bag of words (Arabic stop words). In addition, *our platform allows developers to add new stop words in case of their absence in NLTK.*
- **Segmentation module:** Due to their complexity, many languages such as Arabic and Chinese require another preprocessing tool including tokenization. The segmentation process is based on splitting the text into meaningful units.

In our platform, we have used the **Stanford Segmenter** [44] which segments clitics from words (only). Segmenting clitics attached to words reduces lexical sparsity and simplifies syntactic analysis. However, the ordinary Stanford Arabic segmenter processes stop words and NER that are not listed in its corpus. This latter problem deteriorates the result and can negatively affect the upcoming processing blocks. In order to address this shortcoming, we have combined Name entity Recognition (NER) from Polyglot and stop words from NLTK. Our combination improves the performance of the ordinary segmenter as indicated in the following examples. The main steps are shown in Figure 3.2.

تقع مدينة لندن ببريطانيا. كما انها كانت تسمى قديما لندليام

Before

After

تقع مدينة لندن ب بريطانيا . كما ان ها كانت تسمى قديما لنديام
تقع مدينة لندن ب بريطانيا . ك ما ان ها كانت تسمى قديما ل نديام

تعرف مدينة لندن قديما بلنديام. وهي مشهورة بجمالها

تعرف مدينة لندن قديما ب لنديام . وهي مشهورة ب جمال
تعرف مدينة لندن قديما بلنديام . و هي مشهورة ب جمال
ها

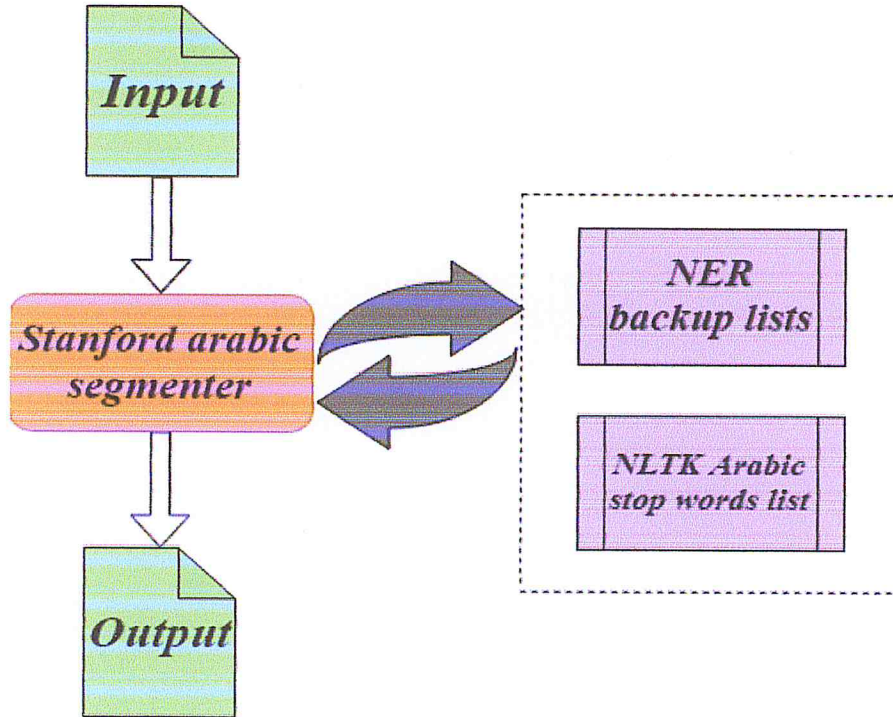


Figure 3.2: Process of segmenting module.

B Stemming

In our platform, we have used **Tashaphyne Arabic light stemmer** that proved a configurable stemmer for Arabic text. It offers many functionalities and features from which we have chosen [45]:

- Extracting roots.
- Extracting stem.
- Star word (getting the rest of word except its root).
- Affixes (getting the rest of word except its stem).

The results of stemming show a lack in dealing with entity named, vocalized texts and stop words. To cope with this shortcoming, we have introduced a preprocessing step to the stemmer which combines **Polyglot NER**, **PyArabic strip vocalization** and **NLTK bag of stop words**. This process begins with strip vocalization of the input text. Then by stemming all the words except the entity named and the stop words to get the best results from the stemmer as indicated by the following the example. The main steps are shown in Figure 3.3.

قام بلاك روك بتأسيس شركة نوكيا في كيلانيمي و هي تقع في جنوب فنلندا	
Before	After
قم ل ر أسس شرك ك ف لنم ه قع ف جنب لند	قم بلاك روك أسس شركة نوكيا في كيلانيمي و هي قع في جنوب فنلندا

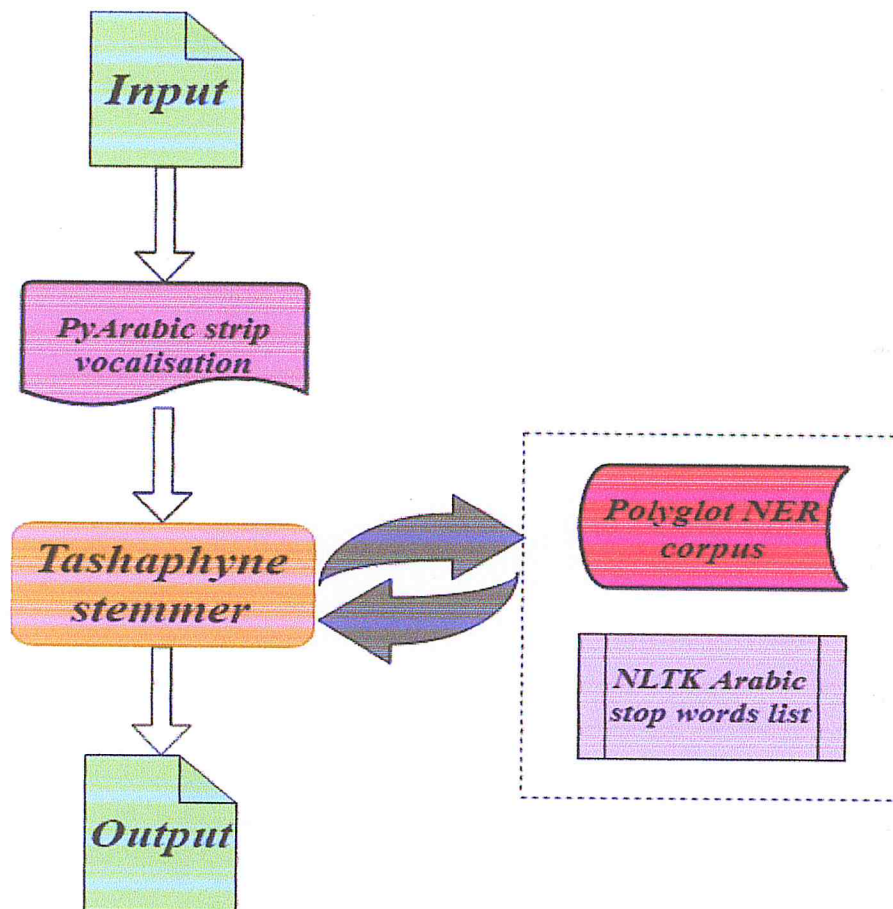


Figure 3.3: Process of stemming module.

C Pos tagging module

A Part-Of-Speech Tagger is a piece of software that reads text and assigns parts of speech to each token, such as noun, verb, adjective, etc. However, computational applications generally use more fine-grained POS tags like 'noun-plural'.

In our platform, we have used the **Stanford Arabic POS tagger** [27] because it provides a set of tags for many languages including the Arabic language. In addition, it is open source and available to be used in different programming languages.

After using the Stanford Arabic POS tagger, we have noticed that it does not process the related pronounces to Arabic word which is solved by segmentation. In order to avoid the use of the **Stanford Arabic Segmenter** then the Stanford Arabic POS tagger, we have managed to use the Stanford Arabic Segmenter as preprocessing step of the Stanford Arabic POS tagger which has improved result of the tagging, as shown in Figure 3.4.

It is worth noting that the set of tags is defined in English (JJ for adjective, VBD for verb in past tense). In our platform we have been able to define the set of tags in Arabic (JJ: صفة, VBD: فعل ماض). Furthermore, our platform offers access to developers to the set of tags and the ability to set them as shown in the following example.

قمت بزيارة مدينة الأبيار وهي تقع بالجزائر العاصمة

BEFORE	After
قمت/VBD	قمت: فعل ماض
زيارة/NN	ب: ضمير متصل
مدينة/NN	زيارة: اسم مفرد
الأبيار/DTNNP	مدينة: اسم مفرد
وهي/PRP	الأبيار: اسم حر. اسم مفرد. اسم علم
تقع/VBP	و: ضمير منفصل
بالجزائر/DTNNP	هي: ضمير للمخاطب أو للغائب
العاصمة/DTNN	تقع: فعل ماضي يعود لضمير مفرد
	ب: ضمير متصل
	الجزائر: اسم حر. اسم مفرد. اسم علم
	العاصمة: اسم حر. اسم مفرد

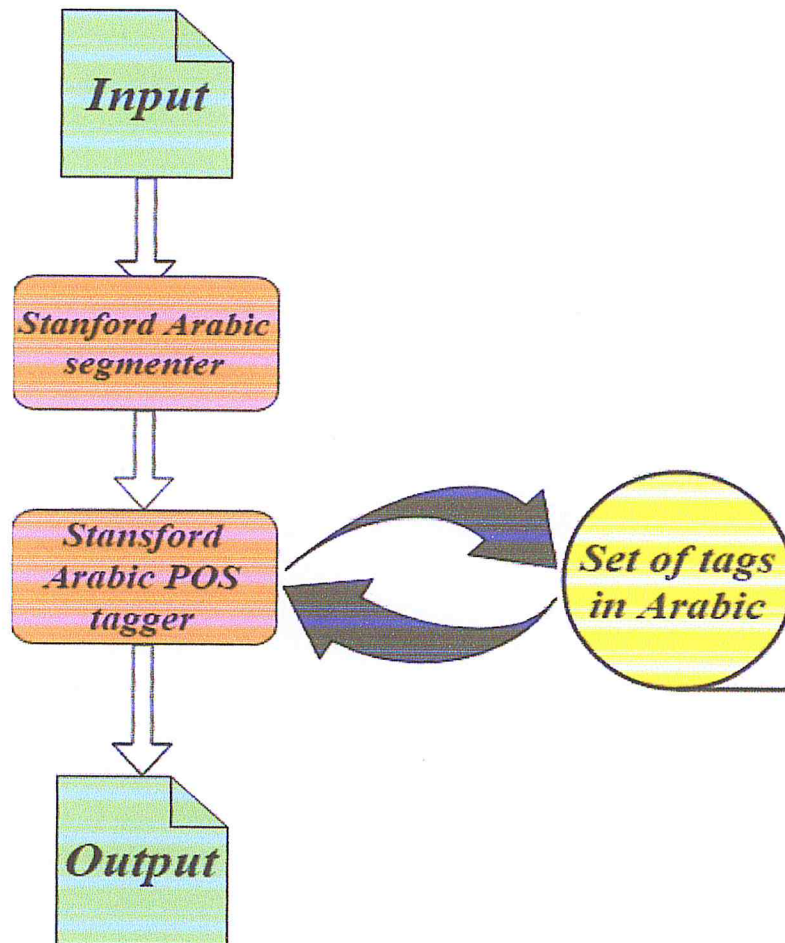


Figure 3.4: Process of POS tagging module.

D Parsing module

Parsing of Arabic sentences is a necessary process for many NLP applications such as machine translation [2], question answering [46] and information retrieval [47]. The **Stanford Arabic parser** [30] is one of the best parsers used in the field of NLP as it has proved efficiency and flexibility. For these characteristics we have chosen to use it in our platform. Arabic parsers are based on the **Penn Arabic Treebank (PATB)**.

In order to improve the results of Stanford Arabic parser, we have introduced the **Stanford Arabic segmenter** as preprocessor like we did before with the **Stanford Arabic POS tagger** as shown in Figure 3.5. The following example illustrates the results of our parser.

تعرف مدينة لندن قديما بلندليام، وهي مشهورة بجمالها	
Before	After
(ROOT (S (VP (VBP تعرف) (NP (NN مدينة) (NP (NNP لندن))) (NP (NN قديما) (JJ بلندليام))))))	(ROOT (S (VP (VBP تعرف) (NP (NN مدينة) (NP (NNP لندن))) (NP (NN قديما) (JJ بلندليام))))))
(ROOT (S (NP (PRP وهي)) (NP (JJ مشهورة)) (NP (NN بجمالها))	(ROOT (S (CC و) (NP (PRP هي)) (NP (JJ مشهورة)) (PP (IN ب)) (NP (NN جمال) (NP (PRP\$ ها))))))

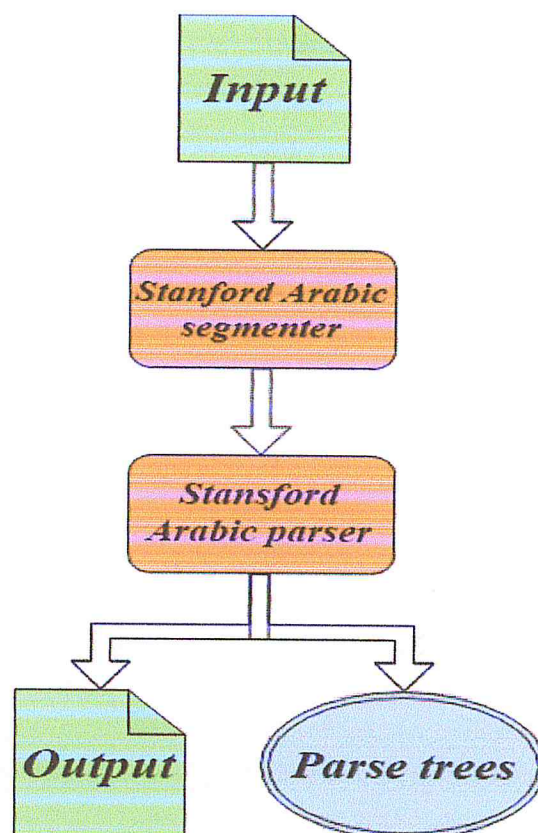


Figure 3.5: Process of parsing module.

3.2.4 Applications

A Named entity recognition module

The Polyglot NER is a task that aims to extract from plain text phrases that correspond to entities. It currently supports 40 languages including the Arabic language. It recognizes three categories of entities [48]:

- Locations (cities, countries, regions, continents, neighborhoods, administrative divisions ...).
- Organizations (sports teams, newspapers, banks, universities, schools, non-profits, companies, ...).
- Persons (politicians, scientists, artists, athletes ...).

We have integrated the **Polyglot NER** in our platform by adding a preprocessing step represented in text tokenization (**Polyglot tokenizer**) to split the text into sentences then into words as shown in Figure 3.6. In case Polyglot NER fails to recognize entities, we have

incorporated a mechanism which uses a **backup lists**, allowing developers to add a new entry in the named entities record as it is shown in Figure 3.6. In addition, we have provided developers with the ability to add new categories of named entities. For instance, we have created a new category called **Money** (العملات النقدية) which contains money units as indicates in the example.

يقع بنك الخليج العربي مدخل بنرتوتة، وهي مدينة توجد غرب الجزائر العاصمة، يتم في هذا البنك تداول العملات النقدية مثل الدينار و الدرهم وغيرها

Before	After
I-ORG ['العربي', 'الخليج', 'البنك'] I-LOC ['الجزائر', 'العاصمة']	I-ORG ['العربي', 'الخليج', 'البنك'] I-LOC ['الجزائر', 'العاصمة'] I-MONEY ['الدينار'] I-MONEY ['الدرهم']

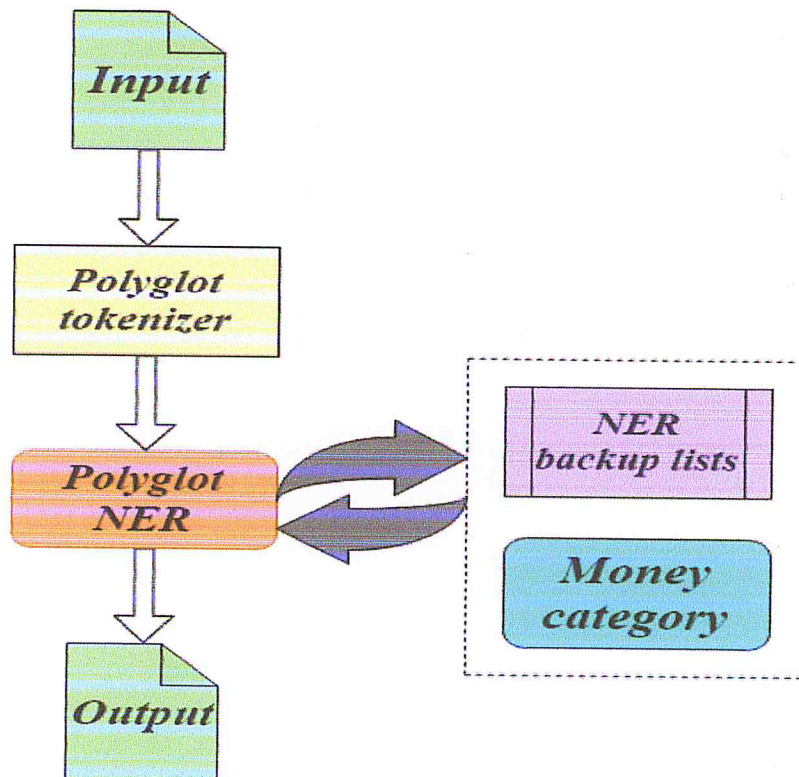


Figure 3.6: Process of NER module.

B Number functions

The `number.py` is a software library that combines the characteristics and functions required by the programmer to deal with Arabic texts. It provides many functionalities such as [49]:

- Convert number to words.
- Convert words into numbers.
- Extract numerical expressions.
- Extract numerical expressions with their context.

We have used this library in our platform to manipulate Arabic letters and numbers, and to provide the user with an additional feature to analyze texts.

C Sentence and Word splitter

The purpose of this module is to identify the boundaries of sentences first, and then tokenize each sentence to identify the words that compose it. In order to achieve this task, we have used the Polyglot tokenizer. It is worth noting that our platform offers various preprocessing tools which are required for many tasks in NLP. We can cite NER.

D Vocalisation module

We have integrated the **Shakkala** project in our platform that use recurrent neural network for Arabic text vocalization. This tool can be used in many applications such as enhance text-to-speech systems or search results. The accuracy of this version has reached almost 95% and more in some cases. Table 3.1 shows an example of this module [50].

Table 3.1: Examples and results [50].

Real output	Predicted output
فَإِنْ لَمْ يَكُونَا كَذَلِكَ أَتَى بِمَا يَفْتَضِيهِ الْحَالُ وَهَذَا أَوْلَى	فَإِنْ لَمْ يَكُونَا كَذَلِكَ أَتَى بِمَا يَفْتَضِيهِ الْحَالُ وَهَذَا أَوْلَى
قَالَ الْإِسْنَوِيُّ وَسَوَاءٌ فِيمَا قَالُوهُ مَاتَ فِي حَيَاةِ أَبِيهِ أَمْ لَا	قَالَ الْإِسْنَوِيُّ وَسَوَاءٌ فِيمَا قَالُوهُ مَاتَ فِي حَيَاةِ أَبِيهِ أَمْ لَا
طَابَعَةٌ ثَلَاثِيَّةٌ الْأَبْعَادِ	طَابَعَةٌ ثَلَاثِيَّةٌ الْأَبْعَادِ

We have applied the PyArabic function “reduce Tashkeel” to get the text mid-vocalizer as shown in Figure 3.7. Furthermore, we have added the strip vocalization and normalize Hamza functions from PyArabic to enrich our platform and to reuse these functions in other tools.

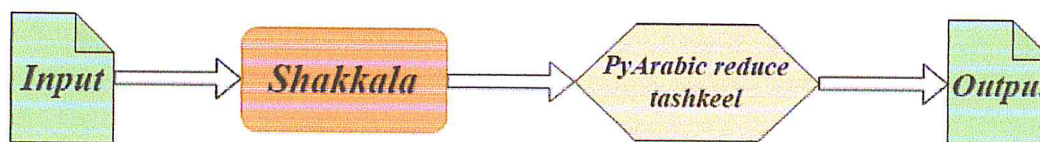


Figure 3.7: Process of the mid-vocalization.

3.3 Interoperability & reusability between tools

3.3.1 Interoperability between tools

It is challenging to integrate some existed tools in building new applications and platforms. Moreover, it is difficult to use the output of a tool as an input of another tool. This is owing to their differences at the inputs/outputs level and architecture. Some tools are standalone tools that process only one task for example a syntactic parser relies separately on tokenizer and POS tagger. The combination of these tools into platforms cannot be done easily.

Once the tool is integrated into a processing framework. It becomes a component that adapts with the framework. The framework specifies the resource and the processing models allowing interoperability between components; hence, getting homogeneous components in their architecture. The ANLP community deals with the same challenge of interoperability. This is mainly due to the lack of mature and flexible platforms such GATE, OpenNLP and NLTK and the lack of Arabic processing components within these platforms.

3.3.2 Reusability between tools

The availability of NLP tools and linguistic resources has led to the development of reusable and shareable tools. Most of the tools were built depending on the necessity of applications or platforms that reduce their reusability. One of the solutions is to standardize the input/output (XML or lately JSON) of the tool that will be integrated into the platform architecture.

Due to the diversity of Arabic tools that have been developed lately, the choice of a reusable tool became much easier than before. This is due to the diversity of programming languages that have been used; for instance, Python and Java.

3.3.3 Comparison between AAAL platform tools

Now that we have presented the modules used in the AAAL platform, we address hereafter the comparative study of the modules before and after their integration in our platform. This study is presented in Table 3.2.

Table 3.2: Comparison between tools before and after their integration in our platform.

Module	Interoperable		Reusable		Inputs		Outputs		Prerequisites	
	Before	After	Before	After	Before	After	Before	After	Before	After
<i>Parsing module</i>	✓	✓	✓	✓	Txt files	Txt, HTML and XML files	Txt	Txt and JSON files Parse trees	NLTK functions	NLTK functions Stanford Arabic segmenter
<i>Segmenting module</i>	✓	✓	✓	✓	Txt files	Txt, HTML and XML files	Txt	Txt and JSON files	NLTK functions	NLTK functions NER backup lists
<i>POS tagging module</i>	✓	✓	✓	✓	Txt files	Txt, HTML and XML files	Txt	Txt and JSON files	NLTK functions	NLTK functions Stanford Arabic segmenter Arabic set of tags
<i>Stemming module</i>	✓	✓	only with python	only with python	Txt	Txt, HTML and XML files	Txt	Txt and JSON files	-	PyArabic strip vocalization Polyglot NER corpus NLTK Arabic stop words list
<i>Tokenizing module</i>	✓	✓	only with python	only with python	Txt	Txt, HTML and XML files	Txt	Txt and JSON files	-	-

Table 3.2 presents the tools used in our platform before the integration. It shows that the inputs and the outputs of each tool were only Txt files or written text. The process of other types of inputs such as XML and HTML files is not available within these tools. In addition, the outputs are only written texts which leads to reusability issues when we need to reuse a tool as a preprocessor of an application or another tool. The reusability of Tashaphyne stemmer and Polyglot tokenizer depend on using Python programming language. The Stanford Arabic tools (parser, segmenter and POS tagger) require NLTK functions in order to process Arabic language texts.

Table 3.2 also presents the modules used in our platform. We notice the variety of the inputs in our modules (Txt, HTML and XML files), which guarantees the process of a large type of files. We also notice the variety at the output level of each module (Txt, JSON files and parse trees an additional output of Parsing module) which leads to get reusable and shareable modules that can be used in other applications, projects and preprocessors of other tools. In addition, the Parsing, Segmenting module, POS tagging and stemming modules require preprocessing tools, lists and corpora in order to improve the results obtained. These combinations yield interoperable tools that will be the foundation for building a flexible platform.

3.4 Conclusion

In this chapter, we have described our platform where the purpose is to build a platform that gathers ANLP tools in a homogeneous architecture. In addition, we have presented a comparative study between ANLP tools.

The conception of our platform has also been described in this chapter. This conception will be put in function in the next chapter.

Chapter 4. Implementation

4.1 Introduction

In this chapter, we present the implementation of the AAAL platform. First, we begin with presenting the development environment and describing the used programming language. Then we describe the main functionalities of our platform, providing the primary steps of text analysis and mentioning the used libraries.

4.2 Development environment

We have used Python programming language along with PyCharm Integrated Development Environment.

4.2.1 Python

Python is an interpreted, object-oriented programming language. It is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.

It was created by Guido van Rossum in the late 1980s as a successor to the ABC language. Python 2.0 released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0 released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. It is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.

4.2.2 PyCharm

PyCharm is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django. It is developed by the Czech company JetBrains.

PyCharm is cross-platform working on Windows, Mac OS X and Linux. PyCharm has a Professional Edition and a Community Edition. It integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including matplotlib and NumPy.

It has many features such as:

- Coding Assistance and Analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes.
- Project and Code Navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages.
- Integrated Unit Testing, with line-by-line coverage.
- Version Control Integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with changelists and merge.

4.3 Description of AAAL platform

Our platform was developed with Python 3.6 using the environment PyCharm that allows to use different **Application Programming Interfaces (API)** from several programming languages such as Java. We have used many functions supplied by different libraries:

- Tokenization
 - Polyglot [48]
 - PyICU [51]
 - PyCLD2 [52]

- Segmenting
 - NLTK [7]
 - StanfordSegmenter.jar [44]
- Stemming
 - PyArabic [33]
 - Setuptools [53]
- POS tagging
 - StanfordPOSTagger.jar [27]
- Parsing
 - StanfordParser.jar [30]
- Named Entity Recognizing
 - Polyglot [48]
- Numbers functions
 - PyArabic [33]
- Vocalisation functions
 - Tensorflow [54]
 - Keras [55]
 - Shakkala [50]
 - PyArabic [48]

4.4 Flow

In this section, we present the different stages of the process of the AAAL platform, from the input until the output, going through the processing steps.

First, we start with the Main window of the platform that offers two modes:

- ❖ Simple mode that allows users to enter a text written in the text box provided in the interface of the platform in order to process it as it is shown in Figure 4.1.
- ❖ Advanced mode that allows users to import files in different formats such as Txt, HTML and XML as it is shown in Figure 4.2.

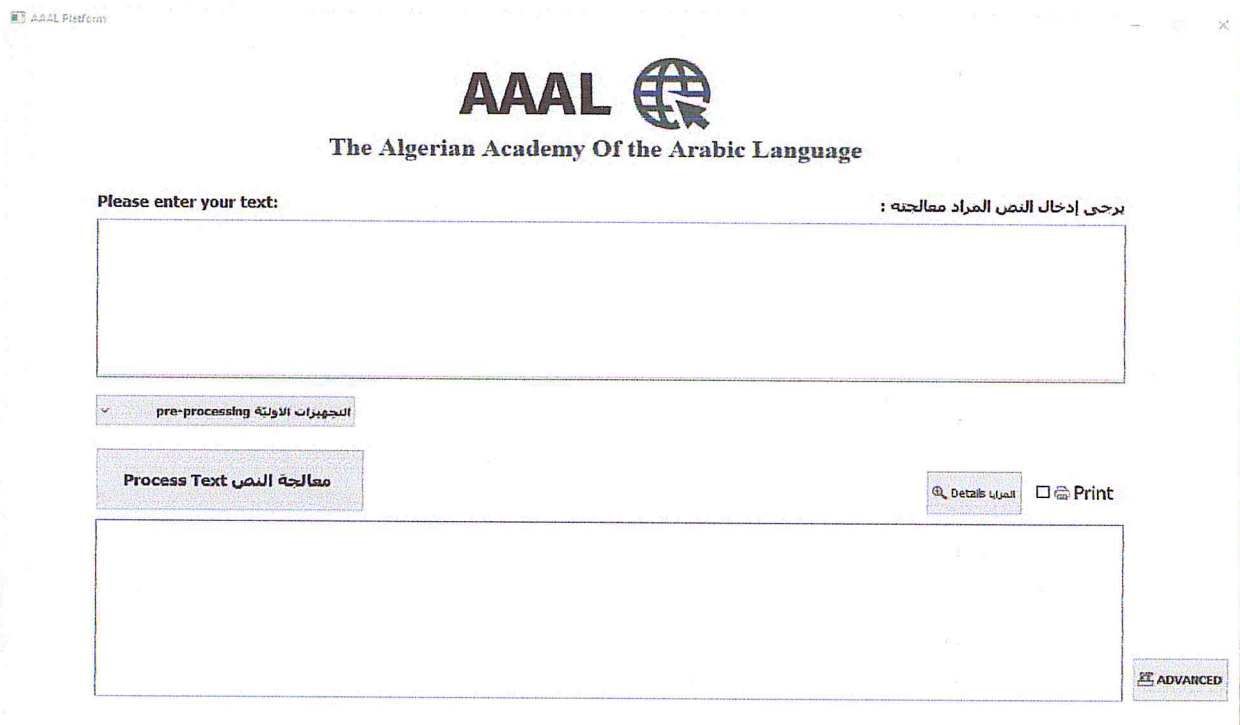


Figure 4.1: Main window in simple mode.

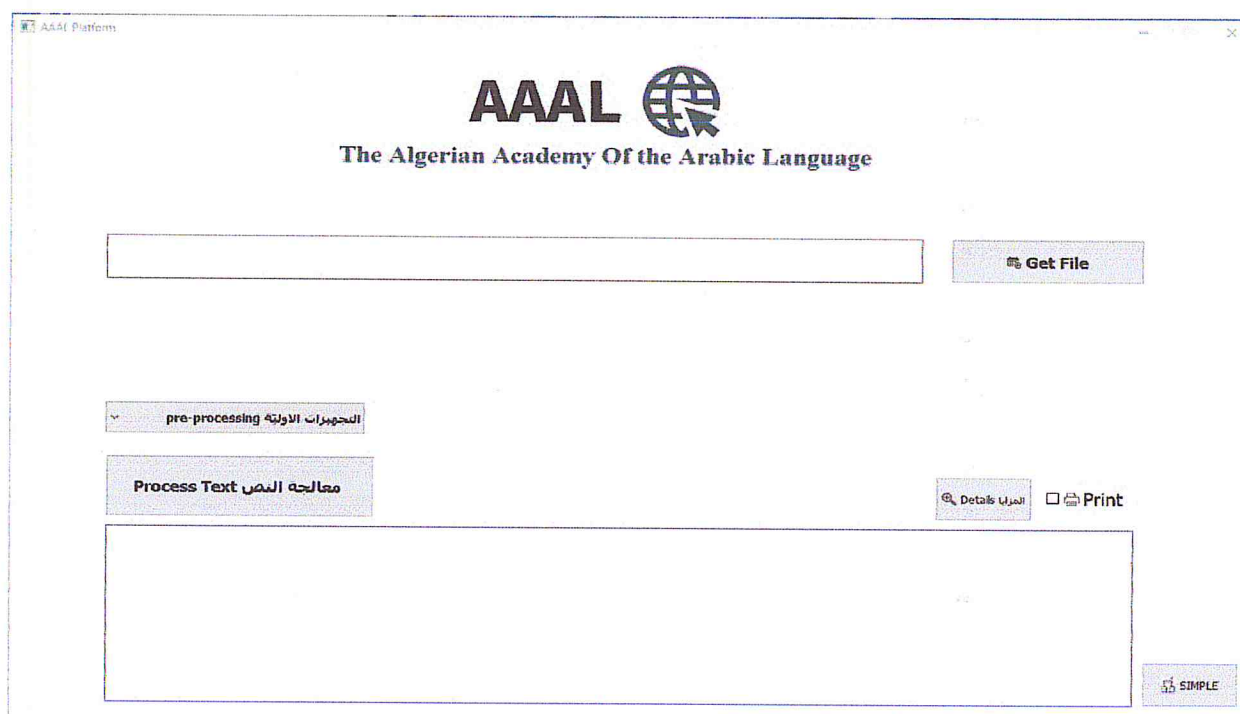


Figure 4.2: Main window in advanced mode.

4.4.2 Processing modules

Our platform offers the possibility of processing texts such as Segmentation and Parsing, as well as the display of treated texts.

The processing modules implemented in the AAAL platform are:

A Preprocessing modules

It splits up into two modules:

Tokenization module


It plays a fundamental role in our platform as a pre-processor of other modules such as the NER module and the Stemming module. It guarantees more efficient results for the tokenization module and for the modules depend on it.

It provides 4 tasks that are shown in Figure 4.3:

- Word token.
- Sentence token.

- Morphemes.
- Stop words removal.

AAAL Platform



AAAL The Algerian Academy Of the Arabic Language

Please enter your text: يرجى إدخال النص المراد معالجته :

تقع مدينة لندن ببريطانيا. كما انها كانت تسمى قديما " لنديام "

pre-processing النجويرات الاولى

word token استخراج الكلمات

sentence token استخراج الجمل

morphemes اقسام الكلمة

Removing Stop-Words برع الكلمات الزائدة

Process Text معالجة النص

[Details المزيد](#)

Print

----- Extracting Word Tokens -----

[تقع, 'مدينة', 'لندن', 'بريطانيا', 'كما', 'انها', 'كانت', 'تسمى', 'قديما', 'لنديام', '']

----- Extracting Sentence Tokens -----

[Sentence("تقع مدينة لندن ببريطانيا"), Sentence("كما انها كانت تسمى قديما " لنديام")]

ADVANCED

Figure 4.3: Tokenization module.

Segmentation module

It is a crucial task in our platform. It is a preprocessor of POS tagging module and Parsing module. From the Stanford Arabic segmenter, we have developed our module which provides better results using the preprocessors. It is shown in Figure 4.4.

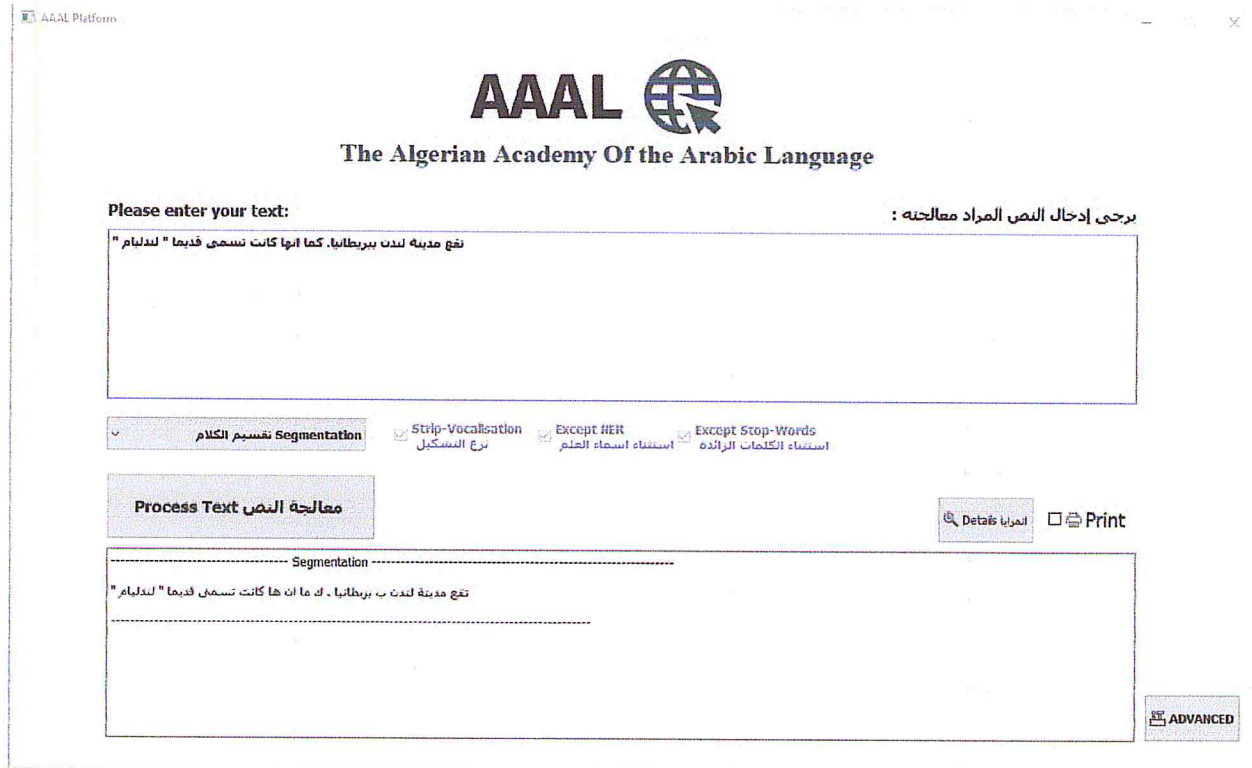



Figure 4.4: Segmentation module.

B Stemming module

It was built from the Tashaphyne stemmer. We have improved the results of the stemmer by using several preprocessors, which has led to the improvement of the obtained results.

It provides the following tasks which are given in Figure 4.5:

- Extract root.
- Extract stem.
- Get star word.
- Get affixes.

AAAL 

The Algerian Academy Of the Arabic Language

Please enter your text: يرجى إدخال النص المراد معالجته :

قام بلاك روك بنأسيس شركة نوکيا في كيلانيمي و هي تقع في جنوب فيلندا

Stemming Words تجذير الكلمات Extract root استخلاص الجذر Extract Stem استخلاص الجذع Get Star Word استخراج الكلمة المجعومة Get Affixes استخراج الكلمات الزائدة
 Strip-Vocalisation ترفع التشكيل Except NER استثناء اسماء العلم Except Stop Words استثناء الكلمات الزائدة

Process Text معالجة النص [المزيد](#) Print

Extracting Words Stems

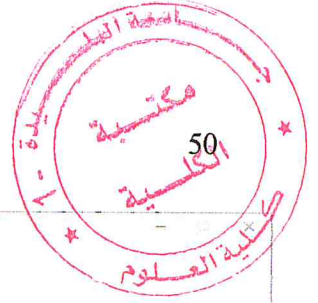
قام
بلاك
روك
أسيس
شركة
نوکيا
في
كيلانيمي
و
هي

ADVANCED

Figure 4.5: Stemming module.

C POS tagging module

It is the task that links the morphological level to the syntactic level. We have developed the module starting from the Stanford Arabic POS tagger that we have improved. It also offers a set of Arabic tags that are shown in Figure 4.6.



The Algerian Academy Of the Arabic Language

Please enter your text:

يرجى إدخال النص المراد معالجته :

فمت بريارة مدينة الأبيار وهي تقع بالجزائر العاصمة

Part-of-Speech (POS) tagging تصنيف Strip-Vocalisation نزع التنوين word token استخلاص الكلمات morphology أقسام الكلمة

Process Text معالجة النص

Details التفاصيل Print

POS Tagging

فمت: فعل ماض منصوب،
ب: ضمير متصل،
ريارة: اسم مفرد،
مدينة: اسم مفرد،
الأبيار: اسم جزر، اسم مفرد، اسم علم،
و: ضمير متصل،
هي: ضمير للمخاطب أو للثاني،
تقع: فعل ماضى يعود للضمير مفرد،
ب: ضمير متصل،
الجزائر: اسم جزر، اسم مفرد، اسم علم،


ADVANCED

Figure 4.6: POS tagging module.

D Parsing module

This module represents the results as Txt and JSON files, also as a parse tree that reflects the syntax of the input. The results obtained by the Parsing module are more precise owing to the use of the Stanford Arabic segmenter as preprocessor. In addition, we have added other tasks such as “Strip-Vocalisation” and “Word token” as it is shown in Figure 4.7.

AAAL Platform

AAAL 

The Algerian Academy Of the Arabic Language

Please enter your text:

قامت بزيارة مدينة الأبيار وهي تقع بالجزائر العاصمة

Strip-Vocalisation word token morphology
نزع التنكييل استخلاص الكلمات اقسام الكلمة

Parsing الإعراب

Process Text معالجة النص

```

(VP
 (VBD قامت)
 (PP
 (IN ب)
 (NP (NP (NP (NP (DTNN الأبيار) (NP (DTNN مدينة) (NP (NN زيارة) (NN (NP (CC و) (S
 (S
 (HP (FRP وهي))
 (VP
 (VBP تقع)
 (PP (IN ب) (NP (NP (DTNHP الجزائر)) (NP (DTNN العاصمة))))))))))))))))))

```

Details Print

ADVANCED

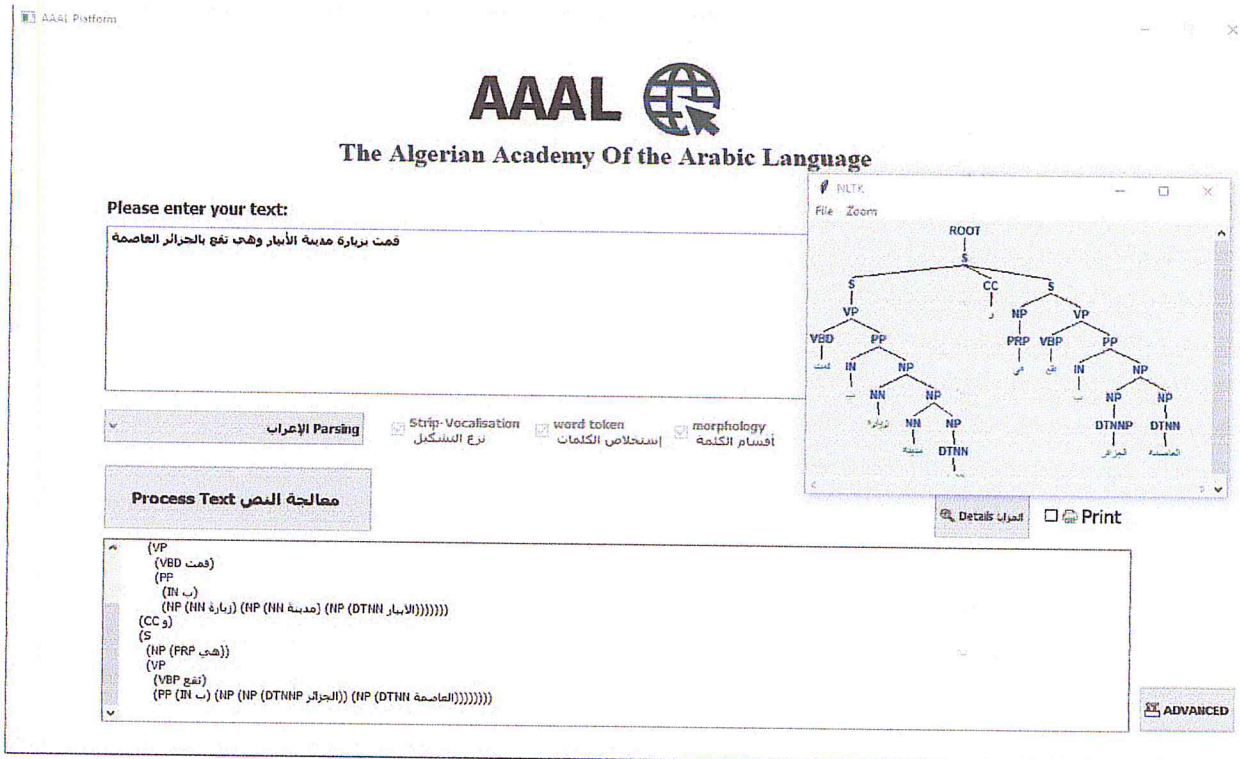



Figure 4.7: Parsing module.

E NER module

The NER module is for the purpose of extracting named entities from predefined categories in **Polyglot NER** (location, person and organization). The integration of the new category “**Money**” enriches the NER module, which also improves the results of the modules that requires NER in our platform. The backup lists that we have integrated also enhance the results of the NER module. The results of these improvements are shown in Figure 4.8.

AAAL Platform

AAAL 

The Algerian Academy Of the Arabic Language

Please enter your text: يرجى إدخال النص المراد معالجته :

يقع بنك الخليج العربي مدخل بنزوتة، وهي مدينة توجد غرب الجزائر العاصمة، يتم في هذا البنك تداول العملات النقدية مثل الدينار و الدرهم وغيرها

Named Entity Recognition **تحديد الأسماء**
 Strip-Vocalisation **برغ السنكجيل**
 word token **استخلاص الكلمات**
 morphology **أقسام الكلمة**

معالجة النص
 التفاصيل
 Print

----- Named Entity Recognition -----

يقع بنك الخليج العربي مدخل بنزوتة، وهي مدينة توجد غرب الجزائر العاصمة، يتم في هذا البنك تداول العملات النقدية مثل الدينار و الدرهم وغيرها

I-ORG [بنك، الخليج، العربي]

I-LOC [الجزائر، العاصمة]

I-LOC [بنزوتة]

I-MONEY [الدينار]

I-MONEY [الدرهم]



 **ADVANCED**

Figure 4.8: NER module.

F Numbers functions

In order to increase the diversity of the modules used in our platform, we have integrated the **Numbers functions** that offers many tasks. The execution of these tasks in the AAAL platform is shown in Figure 4.9.

AAAL 
The Algerian Academy Of the Arabic Language

C:/Users/PC/Desktop/eg.txt Get File

Numbers Functions وظائف الارقام Convert Number to Words تحويل عدد الى كلمة Convert text to number تحويل الكلمات إلى أعداد
 Extract number words with context استخلاص العبارات العددية مع سياقها Extract number words استخلاص العبارات العددية

Process Text معالجة النص Details التفاصيل Print

Extract Numbers with Context

[[وحدات، 'خمسة وثلاثة وعشرون، 'ديساراً'، 'الفانثرت، 'ثلاثة عشر، 'دعيراً]]

Extract Numbers Text

['خمسة وثلاثة وعشرون، 'ثلاثة عشر']

SIMPLE

Figure 4.9: Numbers functions.

G Vocalisation module

It is one of the most interesting and challenging tasks in the ANLP field. We have integrated “Shakkala” project in our platform, which is a deep learning Arabic text vocalization. In addition to other tasks such as **Normalise_Hamza**, **Text Mid-vocalisation** and **strip vocalisation** that are shown in Figure 4.10.

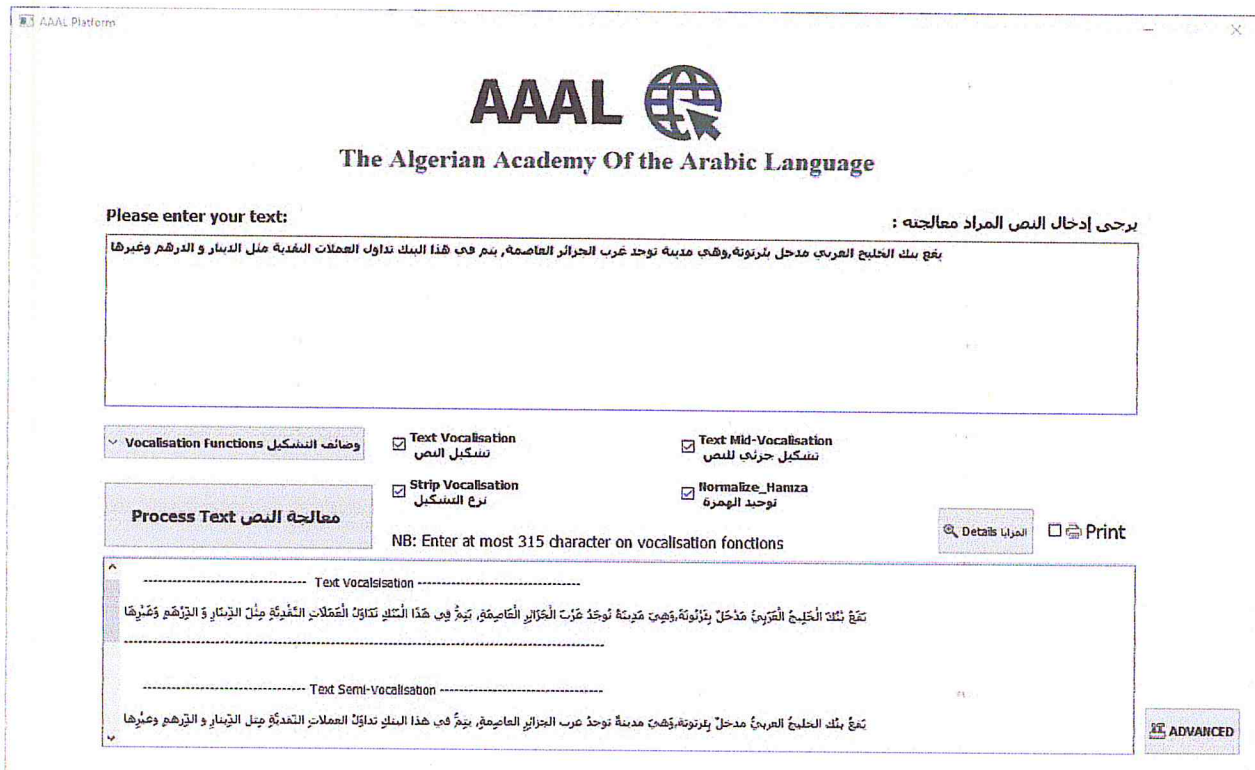


Figure 4.10: Vocalisation module.

4.5 Outputs:

The AAAL platform provides a text box for processing results. In addition, it outputs files into two formats: Txt and JSON files that will be stored in the user's computer as it shown in Figure 4.11.

The files will be stored only when the user checks the **print** box.

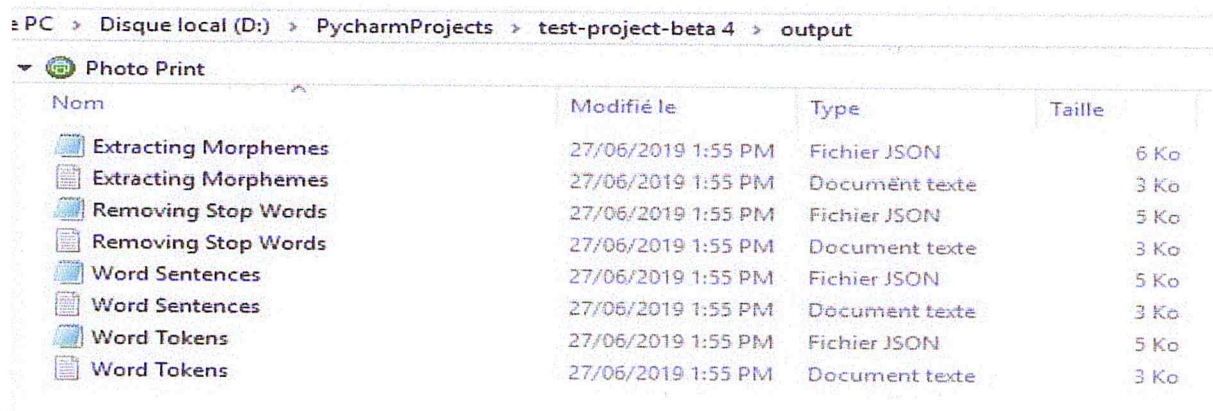


Figure 4.11: Output files.

4.6 Conclusion

Throughout this chapter, we have presented the development environment as well as the programming language used in the developing of the platform. We have also mentioned the different libraries used in each module.

Our platform role is to process several integrated modules on different types of input files. These modules were presented in a **Graphical User Interface (GUI)**. We have described the processing steps of the AAAL platform and we have illustrated them with screenshots.

CONCLUSION

Through our project, we have developed a platform, named AAAL, that amalgamate various ANLP tools. Most importantly, our platform ensures the interoperability and the reusability of these tools. We summarize below our main contributions:

- ✓ We have proposed a homogeneous and interoperable architecture that represents the used modules.
- ✓ We have improved the results of each module integrated in our platform by adding preprocessors and new entities in modules such as the Parsing, POS tagging and NER modules.
- ✓ We have also varied the inputs, which guarantees the diversity of resources (Txt, HTML, XML files) processed in our platform.

Limits and future work

One of the limitations of all the tools developed in the field of ANLP is that they are still premature due to the challenges that the developers face up in dealing with the Arabic language. Moreover, the integration of more tools and applications in our platform such as Lemmatizer, Spell checker and automatic summarizer, increases the variety of the integrated modules.

One appealing work direction consists of presenting the developed platform in a Web interface rather than a desktop application; this is due to the lack of time but it is a work that remains to be done.

Working on this project taught us that the field of NLP is a crucial part in developing many applications such as Automatic Translation, Named Entity Recognition and Discourse Analysis, in many computing science fields such as Artificial Intelligence. In case of the Arabic language, it puts a challenge ahead of every developer working in the ANLP field owing to its morphological richness, the agglutination and dispensability of vowel diacritics.

REFERENCES

- [1] C. Ronan , W. Jason , B. Léon , K. Michael , K. Koray and K. Pavel , "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, pp. 2493-2537, 2011.
- [2] P. Kishore , S. Roukos, W. Todd and Z. Wei-Jing , "BLEU: a method for automatic evaluation of machine translation," *In Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311-318, 2002.
- [3] Z. Xiang , Z. Junbo and L. Yann , "haracter-level convolutional networks for text classification," *In Advances in neural information processing systems*, pp. 649-657, 2015.
- [4] N. Ani and M. Kathleen , "Automatic summarization," *Foundations and Trends® in Information Retrieval*, vol. 5, pp. 103-233, 2011.
- [5] H. Abdelnasser, M. Ragab, R. Mohamed, A. Mohamed, B. Farouk, N. El-Makky and M. & Torki, "Al-Bayan: an Arabic question answering system for the Holy Quran," *In Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pp. 57-64, 2014.
- [6] "GATE," The University of Sheffield, 1995. [Online]. Available: <https://gate.ac.uk/>. [Accessed 7 March 2019].

- [7] "NLTK3.4.3 documentation," [Online]. Available: <https://www.nltk.org/api/nltk.tokenize.html>. [Accessed 06 Juin 2019].
- [8] "OpenNLP Tutorial," Tutorials Point India Private Limited, [Online]. Available: <https://www.tutorialspoint.com/opennlp/>. [Accessed 9 March 2019].
- [9] A. Pasha, M. Al-Badrashiny, M. Diab, A. El Kholly, R. Eskander, N. Habash, M. Pooleery, O. Rambow and R. Roth, "MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic," *LERC*, vol. 14, pp. 1094-1101, May 2014.
- [10] "ATKS," "Arabic Nlp Toolkit Service (Atks), [Online]. Available: <https://www.microsoft.com/en-us/research/project/arabic-toolkit-service-atks/>. [Accessed 10 May 2019].
- [11] K. Bouzoubaa, "SAFAR," 2013. [Online]. Available: <http://arabic.emi.ac.ma/safar/?q=node/13>. [Accessed 13 May 2019].
- [12] M. Althobaiti, U. Kruschwitz and M. & Poesio, "AraNLP: A Java-based library for the processing of Arabic text," 2014.
- [13] M. A. B. MOHAMED, S. ZRIGUI, A. ZOUAGHI and M. Zrigui., "N-scheme model: An approach towards reducing Arabic language sparseness," *5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, pp. 1-5, 2015.
- [14] N. Y. Habash, Introduction to Arabic natural language processing, Synthesis Lectures on Human Language Technologies, 2010.

- [15] "Corpus types," SKETCH ENGINE, [Online]. Available: <https://www.sketchengine.eu/corpora-and-languages/corpus-types/>. [Accessed 07 Juin 2019].
- [16] SAWALHA and M. S. Salem, "Open-source resources and standards for Arabic word structure analysis: Fine grained morphological analysis of Arabic text corpora," *University of Leeds*, 2011.
- [17] E. Kamal, "Microsoft," [Online]. Available: <https://www.microsoft.com/en-us/research/project/sarf-morphological-analyzer/>. [Accessed 05 Juin 2019].
- [18] T. Buckwalter, "Linguistic Data Consortium," University of Pennsylvania, 15 December 2004. [Online]. Available: <https://catalog ldc.upenn.edu/LDC2004L02>. [Accessed 05 Juin 2019].
- [19] N. Habash, "ALMORGEANA - Arabic Lexeme-based Morphological Generation/Analysis," Columbia University, 2005. [Online]. Available: <https://clipdemos.umiacs.umd.edu/ALMORGEANA/>. [Accessed 05 Juin 2019].
- [20] K. R. Beesley, "Xerox Arabic Home Page," [Online]. Available: <file:///C:/Users/ASUS/Downloads/home.html>. [Accessed 05 Juin 2019].
- [21] O. Smrž, V. Bielický and T. Buckwalter, "ElixirFM Online Interface," 2002. [Online]. Available: <http://quest.ms.mff.cuni.cz/cgi-bin/elixir/index.fcgi>. [Accessed 06 Juin 2019].
- [22] N. Habash and O. Rambow, "MAGEAD: a morphological analyzer and generator for the Arabic dialects," *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 681-688, 2006.

- [23] M. N. Al-Kabi, "Towards improving Khoja rule-based Arabic stemmer," *Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pp. 1-6, 2013.
- [24] k. darwish, "Software," [Online]. Available: <https://tides.umiacs.umd.edu/software.html>. [Accessed 06 Juin 2019].
- [25] T. Zerrouki, "Pypi," 2012. [Online]. Available: <https://pypi.org/project/Tashaphyne/>. [Accessed 06 Juin 2019].
- [26] S. Khoja, "APT: Arabic part-of-speech tagger," *Proceedings of the Student Workshop at NAACL*, pp. 20-25, 2001.
- [27] K. Toutanova, C. Manning and D. K. a. Y. Singer, "The Stanford Natural Language Processing Group," 2003. [Online]. Available: <https://nlp.stanford.edu/software/tagger.shtml>. [Accessed 06 Juin 2019].
- [28] M. Diab, "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*, vol. 198, p. 110, 2009.
- [29] R. MOHAMED, N. M. EL-MAKKY and K. and NAGI, "ArabRelat: Arabic Relation Extraction using Distant Supervision," *KEOD*, pp. 410-417, 2015.
- [30] "The Stanford Natural Language Processing Group," [Online]. Available: <https://nlp.stanford.edu/software/lex-parser.html>. [Accessed 06 Juin 2019].
- [31] J. Hall and J. N. a. J. Nivre, "MaltParser," Växjö University and Uppsala University, Sweden, [Online]. Available: <http://maltparser.org/>. [Accessed 06 Juin 2019].

- [32] A. Abdelali, K. Darwish, N. Durrani and H. Mubarak, "FARASA: Advanced Tools for Arabic," Qatar Computing Research Institute, 2016. [Online]. Available: <http://qatsdemo.cloudapp.net/farasa/>. [Accessed 06 Juin 2019].
- [33] T. Zerrouki, "Pyarabic," 2010. [Online]. Available: <https://pypi.python.org/pypi/pyarabic/>. [Accessed 06 Juin 2019].
- [34] M. Davis, "Polyglot," 15 02 2019. [Online]. Available: <https://polyglot.readthedocs.io/en/latest/Tokenization.html>. [Accessed 13 Juin 2019].
- [35] A. Farkiya, P. Saini, S. Sinha and S. Desai, "Natural Language Processing using NLTK and WordNet," vol. 6 (6), 2015.
- [36] "NLTK 3.4 documentation," [Online]. Available: <https://www.nltk.org/#>. [Accessed 28 February 2019].
- [37] in *Natural Language Processing with Python*, United States, 2014.
- [38] H. Cunningham, D. Maynard, K. Bontcheva and V. Tablan, "GATE: an Architecture for Development of Robust HLT," University of Sheffield, Sheffield.
- [39] N. Y. Habash, *Introduction to Arabic Natural Language Processing*, Synthesis Lectures on Human Language Technologies, 2010.
- [40] J. OLIVE, C. CHRISTIANSON and J. (. and MCCARY, *Handbook of natural language processing and machine translation: DARPA global autonomous language exploitation*, Springer Science & Business Media, 2011.
- [41] I. AlAgha, "USING LINGUISTIC ANALYSIS TO TRANSLATE ARABIC NATURAL LANGUAGE QUERIES TO SPARQL," *arXiv preprint arXiv*, 2015.

- [42] Y. Souteh and K. Bouzoubaa, "SAFAR platform and its morphological layer," *Proceeding of the Eleventh Conference on Language Engineering ESOLEC*, pp. 14-15, 2011.
- [43] Y. Jaafar and K. Bouzoubaa, "Arabic natural language processing from software engineering to complex pipeline," *First International Conference on Arabic Computational Linguistics (ACLing)*, pp. 29-36, 2015.
- [44] "The Stanford Natural Language Processing Group," 2014. [Online]. Available: <https://nlp.stanford.edu/software/segmenter.shtml>. [Accessed 21 Juin 2019].
- [45] T. Zerrouki, "Pypi," 2012. [Online]. Available: <https://pypi.org/project/Tashaphyne/>. [Accessed 19 Juin 2019].
- [46] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick and D. & Parikh, "Vqa: Visual question answering," *In Proceedings of the IEEE international conference on computer vision*, pp. 2425-2433, 2015.
- [47] C. Manning, P. Raghavan and H. & Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100-103, 2010.
- [48] R. Al-Rfou, V. Kulkarni, B. Perozzi and S. and Skiena, "POLYGLOT-NER: Massive Multilingual Named Entity Recognition," *Proceedings of the 2015 {SIAM} International Conference on Data Mining*, pp. 586-594, 2015.
- [49] T. Zerrouki, "Pyarabic," 2010. [Online]. Available: <https://pypi.python.org/pypi/pyarabic/>. [Accessed 21 Juin 2019].
- [50] Zerrouki and Barqawi, "Shakkala, Arabic text vocalization," 2017. [Online]. Available: https://ahmadai.com/shakkala/lang_en. [Accessed 21 Juin 2019].

- [51] "PyICU," [Online]. Available: <https://pypi.org/project/PyICU/>. [Accessed 1 July 2019].
- [52] D. Sites, "pycld2," 28 July 2013. [Online]. Available: <https://pypi.org/project/pycld2/>. [Accessed 1 July 2019].
- [53] "setuptools," 2013. [Online]. Available: <https://pypi.org/project/setuptools/>. [Accessed 1 July 2019].
- [54] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean and M. ... and Kudlur, "Tensorflow: A system for large-scale machine learning," *In 12th {USENIX} Symposium on Operating Systems Design and Implementation*, pp. 265-283, 2016.
- [55] F. Chollet, "Keras," 2015. [Online]. Available: <https://keras.io/>. [Accessed 1 July 2019].

APPENDIX

Keys:

1. First interface (simple mode)
2. Seconde interface (Advanced mode)
3. Text area for entered text
4. Button to get the input file
5. Text field for the file source
6. Button to switch to simple mode
7. Button to switch to advanced mode
8. ComboBox to select the processing module
9. Button to show the tasks of each module
10. CheckBoxes to choose tasks
11. CheckBoxes showing hidden details in modules
12. Button to start the processing and show the result
13. Text area for output text
14. Checkbox to get the output as files

How to use?

1. Go to (3) and write your text if you are in simple mode.
2. Click on (8) to select a module then press on (9)
3. Select the tasks you want to process in (10)
4. Press on (14) if you want to get the output as files
5. click on (12) to get results in (13)
6. Press on (7) to switch to advanced mode
7. Click on (4) to get your file
8. Repeat steps from 2 to 5 to get results
9. Press on (6) to switch to simple mode.