Platform for Assessing and Experimenting with Complexity of Arabic Grammatical Structures

Capstone Design – Spring 2015

Wissal Lahjouji

Supervised by Dr. Violetta Cavalli-Sforza
Preface and Acknowledgements

I started thinking about my capstone project during my sophomore year. I remember not being able to sleep for about a week because I wanted to know what I would be working on beforehand. I wanted to create, to make, and to innovate. I wanted my capstone to be groundbreaking, and for people to refer to it semesters after I’d graduate. The downfall to these rather naïve and selfish aspirations was that I had no idea what to work on. I spent weeks digging aimlessly for this cutting-edge project but I remained insatiable. By the time I had to enroll in the capstone course, I still hadn’t found my revolutionary topic. I knew that I wanted to exploit my international studies, communication, and linguistics background, but I wasn’t sure how. I had previously worked with Dr. Cavalli on computational linguistics, and though I hadn’t reached any consistent results, I had gathered enough background in the field to dive into a related project with confidence.

My supervisor then suggested the topic that would prove to be one of the most challenging projects I’ve ever worked on. Completing this project was a milestone for me because I got to put both my fascination with languages and linguistics and my passion for computer science to use. I managed to keep away from the conventional capstone ideas that seemed to have little room for creativity, and gained so much knowledge in a field that I’m actually interested in. I was also introduced to the uniqueness of my own language, Arabic, which I’ve been taking for granted for most of my life, preferring to use French and English for every day conversations. And I don’t want to seem cheesy, but nowadays I make sure to speak Arabic – both the Moroccan dialect and Modern Standard Arabic – because I realized that it’s such a huge asset, yet few of us see the utility in it and tend to lose it after finishing high school.
Acknowledgements

I would like to thank my supervisor Dr. Violetta Cavalli-Sforza who trusted me with such an ambitious project, who gave me room to customize it and make it mine, and who assisted me in what was a very busy semester for her. She was available during weekends, holidays, and at very peculiar times of the day. She always answered e-mails in the span of minutes and provided me with resources and valuable pointers, along with editing my final report very thoroughly, both the content and format.

I also dedicate the completion of this project to my mother who remains my number one supporter and who manages to be interested in exactly everything I do and write. She spent a great amount of time giving me pointers and helping me make this project understandable to people with no background in NLP. She also gave me words of encouragement when I needed them most. Many other people came to my mind when writing this section, namely my sister whose infatuation with the intricacies of languages has exceeded mine and who surprises me on a continuous basis; my father who always taught me to aspire to excellence and greatness and to always aim higher; and my friends who had to deal with my continuous existential crises throughout the semester and who kept insisting that I could pull through. I also want to thank Mr. Chraibi who oriented me towards Computer Science and without whom, I probably wouldn’t be majoring in this field, Ms. Mourhir who introduced me to the world of research and scholarship, and my advisor Mr. Iraqi who once told me I was more than just a grade.
## Contents

Preface and Acknowledgements ................................................................. 1
Abstract ........................................................................................................ 5
Introduction ................................................................................................... 1
Chapter 1 – Motivation................................................................................... 3  
  1. Arabic’s Growth on the Internet ............................................................. 3  
  2. Arabic’s Popularity as a Second-Language .......................................... 5  
  3. Motivation behind the Project ............................................................... 5
Chapter 2 – Theoretical Background ........................................................... 7  
  1. Literature Review .................................................................................... 7  
  2. Computational Linguistics ...................................................................... 10  
  3. Symbolic vs Statistical Approach ............................................................ 11  
  5. Parsers and Constituency vs. Dependency ............................................... 13  
  6. Treebanks ............................................................................................... 16  
  7. Why Arabic is a Challenge to Computational Linguists ....................... 19  
  1. General Sources of Ambiguity in Modern Standard Arabic ............... 19  
  2. The Arabic Script .................................................................................. 20  
  3. Diacritics .............................................................................................. 21  
  4. Other Ambiguity Sources .................................................................... 22
Chapter 3 – Approach .................................................................................. 24  
  1. General Approach ................................................................................ 25  
  2. The Stanford Parser .............................................................................. 26  
  3. The Texts ............................................................................................. 26  
  4. The Statistics ....................................................................................... 27
Abstract

Arabic is the fifth most spoken language in the world and the fastest growing one on the Internet. It has thus gained momentum and has seen a clear increase in the research efforts dedicated to it. However, many gaps remain in the scholarship pertaining to Arabic Natural Language Processing (ANLP). One of those gaps is related to the study of its syntactic and grammatical complexity.

This capstone is an attempt at making a contribution to that field by building a platform to assess the syntactic structures of any given text in Arabic and to determine its level of complexity. It uses a statistical approach which consists of running academic texts through the Stanford Arabic parser and storing data pertaining to those parses to draw eventual patterns and come up with metrics to characterize complexity. As a research project, this capstone consists of a theoretical component, which is thoroughly addressed in this report, and of an implementation component which is in turn divided into a Web application and a Java background process runnable through the command line.
Introduction

This report is the culmination of a semester of research and implementation, in addition to months of getting acquainted with the world of computational linguistics and natural language processing. The project is an attempt at assessing the complexity of Modern Standard Arabic’s syntactic structures computationally, and more specifically at developing a set of metrics capable of analysing Arabic texts and characterizing them in terms of their complexity levels. It initially started as a component of a much larger project aimed at assessing readability of written Arabic text. But it can also be considered an addition to the burgeoning scholarship pertaining to Arabic Natural Language Processing.

Given the intricate nature of the project and its linguistics roots, this report will be structured in a way that will enable a person with no background in computational linguistics to understand the adopted process and each of the steps undertaken. For this reason, a good portion of the report will be dedicated to providing theoretical background and to explaining the motivation behind such a project.

Accordingly, the first chapter is designed to expose the reader to the larger picture for a better understanding of the motivation behind Arabic Natural Language Processing in general and this project in particular. The second chapter will be dedicated to theoretical background and will be divided into sub-chapters given the heavy technical jargon that will be encountered throughout the report. The first section will enumerate several definitions of complexity and explain which one this project is concerned with. Then it will provide a literature review of key contributions in this field of research. The second section deals with computational linguistics and NLP. The third one discusses the difference between the
symbolic and statistical approaches to computational linguistics. The fourth section describes the framework for linguistic analysis adopted in this project, and shows which levels it focuses on. The fifth section introduces the reader to parsers and describes the difference between constituency and dependency grammar. The sixth section defines Treebanks and describes the three most prominent ones for Arabic. And finally the last sub-chapter enumerates the key features of the Arabic language and exposes the reader to the peculiarities that make it such a challenge for computational linguists.

The third chapter describes the methodology and approach used in this project and explains why each decision has been made, whereas the fourth one is dedicated to describing the developed web-based platform that aids this project in collecting statistics and storing them in the database for later retrieval and analysis. The conclusion then summarizes the entire project and provides further recommendations for further research.
Chapter 1 – Motivation

Before going through the motivation behind this project, I would like to dedicate a section to the motivation behind Arabic computational linguistics in general. The reason behind this choice is largely due to the general response I got from people who heard about my project. Most of them, while feigning interest, failed to see the utility in such a task and its eventual use. And while this reaction can be attributed to my inability to fully convey the motivation behind my efforts, another reason might be that all the people I got such feedback from are Arabic-speakers who speak two or three other languages and who no longer use Modern Standard Arabic (MSA) in their daily lives. They’re people with alternatives who now take Arabic for granted – something I am also slightly guilty of, or rather was. The subsequent section will thus explain why researchers and computational linguists have taken a special interest in Arabic in the last fifteen years, and hopefully enlighten those who can’t grasp its importance.

1. Arabic’s Growth on the Internet

Being the official language of 22 countries [1], Arabic is the fifth most spoken language in the world [2] [3] with over 200 million speakers from northwest Africa to southwest Asia [1]. 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 [4]. Arabic is also the 6th most used language on Twitter with an increase of 2000% in the year of 2011[5] making it the biggest growth in the social media platform’s history, and the 9th most used one on Facebook in 2012 [6]. 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 [4]. 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 analyse 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.

Indeed, most of the Arabic present on the Internet isn’t in MSA – only 1% of all web pages are, according to the Economist [7] – but in a mixture of standard and dialectal Arabic. These tendencies foretell that most of the texts produced by social media do not conform to standard grammar and orthography rules, and are even more difficult to process using computational tools. This divergence is mostly due to the fairly recent advent of support for Arabic Script and the fact that many Arabic-speakers have gotten used to transliterating content into Roman script especially in North Africa. 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 discrepancy 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” [7].

As previously mentioned, many giants of the tech world have already integrated Arabic (Twitter, Facebook, Youtube) into their platforms, but most intricate applications such as machine translation to and from Arabic are still wobbly, and that is due to a lack of research in the syntactic particularities of the language.
2. Arabic’s Popularity as a Second-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” [8] according to an enrolment survey conducted by the Modern Language Association (MLA). Arabic enrolment rates grew by 126.5% between 2002 and 2006, and became the 8th most studied language in the US in 2010. While the interest was certainly sparked by the 9/11 events and the Arab Spring, knowing Arabic is nowadays “critical” to humanitarian efforts, state department and intelligence positions, diplomacy, and business. 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, automated tutoring systems etc..

3. Motivation behind the Project

Now that I’ve established the importance of Arabic computational linguistics, I will move to explaining the motivation behind this project in particular. Language acquisition, especially in writing is structured in terms of levels of “complexity”. In school for example, most languages are taught in stages. The first stage is becoming familiar with the script (Roman, Arabic…) and phonemes, the second with the morphology of the word, the third with the syntax of the sentence and with semantics. Texts assigned for readings get more complex and sentence structures get ‘complicated’ the more advanced the learner gets. As research for automated ways of teaching, learning, and processing natural language has
expanded, so has the field of computational complexity of natural languages, especially for second-language acquisition [9].

In the case of Arabic, there is a clear increase in the complexity of syntactic structures in the assigned readings accompanying the learner’s process of acquiring the language. However, the criteria to determine complexity hasn’t been defined for machines. A human-being fluent in the language might be able to determine how complex a text is, but a computer wouldn’t be able to do so without a clear set of metrics to base its analysis on. Automated tutoring systems, automatic homework generation, and other tasks are thus not yet possible in an optimal way, and would always require the presence of a human with expertise in the language.

This capstone project attempts to develop a set of metrics to characterize syntactic complexity of a given text in Arabic through a Hybrid approach combining statistical and symbolic approaches to syntax (these terms are further detailed in the theoretical background section). It is worth noting that Arabic is nobody’s “mother tongue” despite being spoken by over 200 million people [10]. It is taught at school like any other language, and nobody grows up speaking it with their parents, which is why this research is relevant to both people seeking to learn it as a second language and native Arabs as well, in this case children.
Chapter 2 – Theoretical Background

This chapter is very heavy in linguistics jargon. It is necessary to lay down the foundations of computational linguistics in order to fully comprehend the nature of the steps undertaken.

1. Literature Review

In the preface of “Linguistic Complexity: Second Language Acquisition, Indigenization, Contact”, Diane Larsen-Freeman describes linguistic complexity as “complex” [11]. The statement which serves as an opening for the book sums up, rather conveniently, the issue at stake. Complexity has indeed been a subject of debate among linguists and second-language acquisition scholars in particular, in which it has become a trendy topic. In 2007, a workshop at the “Winter Meeting of the Linguistic Society of America” was devoted to the topic, leading to another workshop in the same year in Leipzig “Workshop on Language Complexity as an Evolving Variable” [12]. Projects focusing on linguistic complexity burgeoned in that same decade, one of them being “The Grammatical Complexity of Natural Languages” in Helsinki which spanned the year 2003 until 2006. A more recent workshop on “Formal Linguistics and the Measurements of Grammatical Complexity” took place at the University of Washington in Seattle in 2012. Many other projects then started in order to characterize complexity, but were rather preoccupied with
comparing languages in terms of complexity, rather than focusing on the language itself. But what is linguistic complexity? And “can it be objectively measured?” [13].

It has already been established that linguists can’t really agree on a definition of linguistic complexity. However, many typologies were suggested to define the word. According to Pallotti [14], three meanings can be found in linguistic research:

“1. Structural complexity, a formal property of texts and linguistic systems having to do with the number of their elements and their relational patterns; 2. Cognitive complexity, having to do with the processing costs associated with linguistic structures; 3. Developmental complexity, i.e. the order in which linguistic structures emerge and are mastered in second (and, possibly, first) language acquisition.” (14, 118)

Another typology classifies complexity as absolute or relative. Absolute complexity is characterized as not being “related to the experiences of a particular kind of user” [15], making it only dependent on the description of the language and the theories related to it rather than on the time spent learning it. Relative complexity is thus relative to the language users. Wouters adheres to this definition of complexity and chooses to define it as “the amount of effort a generalized outsider has to make to become acquainted with the language” [14]. This view is met with skepticism by Sinnemaki who believes that “grammatical complexity should not be based on difficulty but be kept apart from it”, adding that basing complexity on cognition would make the research efforts biased towards a certain user type and would limit variation [16].

For this project, we adhere to the “absolute complexity” definition. In other words, we don’t look at the complexity of the language from the user’s perspective (e.i the time it took them to acquire the language), but at the language itself and at its specificities. In the words of
Rescher (17, 1) “Complexity is first and foremost a matter of the number and variety of an item’s constituent elements and of the elaborateness of their interrelational structure, be it organizational or operational” [17]. Accordingly, and as my supervisor Dr. Cavalli pointed out, “measures of complexity are necessarily linked to the style and or theory used for the analysis”. In the case of this project, we are looking at grammatical complexity by analysing the different morphological and syntactic structures of Arabic.

A great example of an application of “complexity research” is Coh-Metrix, a “system for computing computational cohesion and coherence metrics for written and spoken texts” [18]. This web tool allows users to assess the ‘difficulty’ and ‘complexity’ of any given text in English and Traditional Chinese. It uses different metrics to do so namely: descriptive indices, lexical diversity, syntactic complexity, word information, and seven more. Developing this system took years of research at the University of Memphis [19], and it is now used in computational linguistics to conduct research on, for example, vocabulary complexity loss when a text is translated [20], or on the quality of the writings of second-language learners [21]. The aim of the capstone is to produce a similar tool for Arabic but only for syntactic complexity given the time constraint.

I couldn’t find much literature to review on Arabic complexity research, as it is usually only brought up to make a case against a certain definition of linguistic complexity rather than being studied. It is however present in English, though the literature remains shallow. This lack of scholarship can be explained by the rather nascent nature of the field, and by the many idiosyncrasies of the Arabic language. The following section will introduce the reader to the fields of Computational Linguistics and Natural Language Processing.
2. Computational Linguistics

Based on its name, computational linguistics is an interdisciplinary field combining computing and linguistics. Digging a little deeper, the Stanford’s Encyclopaedia of philosophy defines computational linguistics as “the scientific and engineering discipline concerned with understanding written and spoken language from a computational perspective, and building artifacts that usefully process and produce language” [22]. In other words, it’s a field concerned with making computers “linguistically competent”. Thus, iPhone’s Siri, Google Translate, Voice Recognition security systems, speech recognition applications, and so on, are all applications of Computational linguistics.

A little distinction must be made between CL and Natural Language Processing (NLP) as most seem to confuse the two. Computational linguistics aims to “model aspects of the human language faculty using formal computational models” [23] – two of which will be described in the subsequent paragraph – with the purpose of studying and understanding language as a phenomenon. NLP, on the other hand, is an engineering field concerned with automating “textual and linguistic analysis, generation, representation, and acquisition”, and mostly focused “on cross-cutting techniques”. [23]

Computational linguistics started soon after the advent of computers in the 1960s following a structured approach to natural language processing most call the “symbolic” or “knowledge-based” approach [24]. That was until the advent of personal computers in the 1980 and later on of the Internet which created a paradigm shift towards a statistical or “data-driven” approach [25], given the great amount of machine-readable data available.
3. Symbolic vs Statistical Approach

The symbolic approach, which was used in the first NLP applications, relies on “the linguistic description of a language”, as it attempts to “mimic the knowledge that native speakers have when they […] understand speech or written language” [25]. It got its name from the rules for the manipulation of symbols it consists of. In other words, it’s an approach that determines the ‘grammaticality’ [26] of a sentence (whether it follows a known grammatical structure or not). It tends to “work top-down by imposing known grammatical patterns and meaning associations upon texts” [27].

The statistical approach, which became popular after the advent of the Internet, relies on a bottom-up approach which looks at texts first and looks for patterns by going through the large corpus of texts and determining which grammar is “most statistically probable” [27]. The statistical approach is the most widely used one nowadays given the great amount of data to draw patterns from. The basic assumption is that machines can learn linguistic knowledge from linguistic data through enough training [28], and it takes much less development time because it “induces the knowledge to describe what is used”, instead of what is possible.
4. Framework for Linguistic Analysis:

As an interdisciplinary field, computational linguistics follows a formal approach to language. Several levels of language are considered and they form the hierarchy presented in the figure above, and detailed below:

- Phonetics and phonology are concerned with the possible sounds in a given language and with their interactions to create and change meaning.
- Morphology 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.
- Syntax 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.
• Semantics is a “hot topic” and it deals with meaning which is very hard to characterize. This capstone is not concerned with semantics.

• Pragmatics is the top level of linguistic analysis as it deals with the use of the language in specific situations. The project doesn’t consider this level either.

This project is concerned with the second and third levels of language, namely morphology and syntax, as they form grammar. The remaining theoretical descriptions are only related to these two levels. Accordingly, given that we’re now familiar with computational linguistics and natural language processing and established a framework for our analysis, we can finally delve into the specificities of those two fields that will help the realization of the project.

So how does NLP actually work? And which of those functionalities are we concerned with? The following sections are simplified so that the reader doesn’t get caught in the details of each process but gets an overall picture that will still enable him or her to understand the adopted approach which will be described in full detail in the following chapter. I will start by introducing what we mean by a Parser and a Treebank and then determine their importance in the project.

5. Parsers and Constituency vs. Dependency

A parser, in the context of this project, is a tool developed by NLP specialists with the aim of extracting a syntactic structure from a given sentence. Parsers are also used to parse formal languages, such as C for compiling purposes, but Natural Language parsers are more complex in the sense that they “account for a broader variety of phenomena”. A parser can thus be perceived as a black box which takes a sentence as input, and produces a parse tree as
output. A parse tree is a structural description of the sequence of elements that make the sentence and it has a tree-like format.

Now, we’ve already seen how linguists love debating basic notions, and how the field of computational linguistics follows the paradigm shifts in information technology. Thus, having more than one approach to parsing seems almost natural. Phrase-structure or constituency grammar is a formal approach to syntax introduced by Noam Chomsky when he pinned the term CFG or context-free grammar. The main idea is that each sentence is composed of phrases which are in turn made of sub-phrases etc. This description of grammar thus starts with “sentence” at the top level and then divides it into Noun Phrase (NP), Verb Phrase (VP) and so on. Constituents are very important because they play a specific role in the sentence, such as the noun’s role in a Noun Phrase. A figure is provided after this section as an example of the approach.

Dependency grammar on the other hand, is less rigid on the syntactic front. The idea of dependency relations dates back to Lucien Tesniere’s work in 1959 in which he claims that:

“The sentence is an organized whole, the constituent elements of which are words. Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. The structural connections establish dependency relations between the words.” (29, 1).

The basic assumption is that syntactic structures consist of “lexical items linked by binary asymmetrical relations called dependencies”. Thus, instead of looking at the entire sentence and dividing it into sub-phrases, dependency grammar looks at components of the
The central idea is that “in each phrase one word, the head, controls the other words, the modifiers” [26]. The head word is in this sense “obligatory and determines what other constituents can occur in the phrase” [30]. For example in the phrase “a long report”, the head is ‘report’ and its modifiers/dependents are ‘a’ and ‘long’. The concept of the “head” is important in both approaches but with different purposes.

An example of a comparison between constituency and dependency grammar is presented below with the spelled out meanings of the abbreviations:

<table>
<thead>
<tr>
<th>In the Constituency Tree</th>
<th>In the Dependency Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>S: Sentence</td>
<td>V: Verb</td>
</tr>
<tr>
<td>NP: Noun Phrase</td>
<td>N: Noun</td>
</tr>
<tr>
<td>VP: Verb Phrase</td>
<td>D: Determinant</td>
</tr>
<tr>
<td>D: Determinant</td>
<td>A: Adjective</td>
</tr>
<tr>
<td>V: Verb</td>
<td></td>
</tr>
<tr>
<td>A: Adjective</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 5.1: Constituency Vs Dependency Parse Trees](image)
6. Treebanks

Treebanks are “collections of manually checked syntactic analyses of sentences” [30] 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 Natural Language Processing. For Arabic, 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” [31] and it models the English version of the Penn Treebank. It was the first major effort for Arabic, and it remains the most sought one as CATiB and PADT both used it and converted it to their “own representation” and then annotated the data. Here is an example of a PATB parse tree from Nizar Habash’s book [32].
The Prague Arabic Dependency Treebank (PADT), as its name indicates, follows the dependency approach described in previous sections. It relied on a conversion from the Penn Arabic Treebank which relies on Constituency grammar, and then annotated it accordingly to fit the dependency paradigm. The Prague Arabic Dependency Treebank was used in the CoNLL 2006 and 2007, tasks which tried to promote dependency parsing. An example of a dependency parse tree for the same sentence above is the following:
The Columbia Treebank (2008) on the other hand, 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” [32]. Statistical parsers are thus trained on these Treebanks depending on the grammar they follow and on the eventual use of the parse trees.
7. Why Arabic is a Challenge to Computational Linguists

1. General Sources of Ambiguity in Modern Standard Arabic

I previously introduced Arabic in terms of statistical data pertaining to its widespread usage and growth, and I’ve already hinted at how Arabic consists a challenge for computational linguists and NLP specialists. This section is dedicated to explaining why it is the case. Before we dive into the linguistic jargon, we should address the peculiarities of Arabic in general. As I’ve previously mentioned, Arabic is nobody’s first language as there is a clear absence of the native speaker. By Arabic, I thus refer to Modern Standard Arabic (MSA), and not to Classical Arabic or to its numerous dialects. This discrepancy results in what linguists call ‘diglossia’, a situation in which two languages or dialects exist within the same language community. For this reason, Arabic isn’t given enough importance by Arab grammarians unless it has to do with the analysis of texts from the Qur’an which is in Classical Arabic and not in MSA. The most widely used modern Arabic grammars were written by foreigners.

Arabic natural language processing started peaking the interest of computational linguists in the late 70s and early eighties [24]. 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. In the following section, I attempt to summarize this richness starting with the basics. Please note that most of the examples presented below were take or inspired from Habash [32] and Farghaly [24].
2. *The Arabic Script*

This project initially assumes that the reader is familiar with the Arabic script and will not dwell on its description. Here are the basics, however. Arabic script is mostly composed of “letter form” and “letter marks”.

Letter forms are the basic graphic backbones of Arabic letters.

![Letter Forms](image)

Figure 7.2.2: Letter Forms [31]

Letter marks are necessary to distinguish different letters. The figure features five dots/points, af, three Hamzahs and the Madda.

![Letter Marks](image)

Figure 7.2.2: Letter Marks [31]

One challenge in Arabic for computational linguists was delimiting ‘words’ as letters are not always connected. The project is not concerned with these ambiguities as we’re on level 2 and 3 (morphology and syntax). However, the main problem with the Arabic script is “Diacritics”.
3. **Diacritics**

Diacritics are symbols added to Arabic words to indicate a Vowel, Nunation, or Shadda. The issue with diacritics is that they’re optional, and that most of MSA doesn’t have them which creates ambiguity.

For example, the word can be read as (he wrote), as (it was written) and as plural of book.

In this case, the root of the word is kept and the gist is maintained. However, in some case...
cases the complete meaning of the word changes, for example can be read as (foot), and as (man). Furthermore, as underlined by Dr. Cavalli, when the root consists of “weak consonants” such as ‘w’ and ‘y’, the root becomes even harder to identify.

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. For example the word can be interpreted as:

1- wa- hamm-y as in my worry
2- wahm-y as in imaginary
3- wáhm-y as in my illusion

This problem is due to the fact that Arabic allows attaching prepositions and affixes to nouns or verbs which result in syntactic ambiguity.

4. Other Ambiguity Sources

Arabic is also a pro-drop language which means that the pronoun is dropped and it has a passive voice. This creates further ambiguity especially with the absence of diacritics as shown in the example below. The sentence: can be interpreted in three ways:

أكلت الدجاجة
- I/he/she/you ate the chicken
- The chicken ate
- The chicken was eaten

Apart from the syntactic and morphological ambiguity, Arabic has longer sentences than English which are linked through conjunctions. These conjunctions can be proclitics, which are clitics attached at the beginning. A clitic is a morpheme that operates like a word, but that is always attached to another one either preceding or following it. The issue with Arabic proclitics is that they have homonymous but have different functions. For example ‘wa’ can serve as conjunction, coordination, subordination, and so on… Enclitics, which are clitics attached at the end of another word, also cause this issue as they can designate gender and possessiveness at the same time, along with number, plurality.
I won’t go into the details of Arabic morphology because that would take the attention away from the topic, a great reference is Nizar Habash’ book “Introduction to Arabic Natural Processing” (2010). Instead of going through possible syntactic structures in Arabic which would be the symbolic approach, I would rather use the statistical one and derive them from the parses that we will run later.

**Chapter 3 – Approach**

Now that we’ve laid down the theoretical foundation, we can finally delve into the method used in this project. As mentioned in the motivation chapter, this capstone aims at assessing the complexity of any given text in Modern Standard Arabic. The optimal scenario is the following: the user inputs text, the system analyses it, then assigns a complexity level to it. I made sure to explain what the project means by complexity. In case the reader is still unclear over the use of this specific word in this report, please refer to the first section of the second chapter. Now we will talk about how to achieve that goal and scenario using NLP and computational linguistic tools.
1. General Approach

When I first started working on the project, the initial assumption was that whatever makes a parser break is likely to break the user’s understanding as well, given that parsers try to mimic the knowledge process of users. However, as I tried to determine “what makes a parser break”, several problems arose. The first one is that I’m not an expert in Arabic linguistics, and that no matter how much I read about its syntactic properties, I wouldn’t be able to go through large amounts of texts and possibly come up with correct parses to them. As time went by, it became clear that a human-being with knowledge in the field was needed so that the project could reach any tangible conclusions. This would take a great amount of time and effort that I simply can’t afford wasting.

However, the texts that my supervisor Dr. Cavalli-Sforza provided me with, which were academic texts in Arabic used for teaching the language to natives in Arab countries, proved to be valuable and central to the project, and made me shift from my initial approach. Indeed, the texts I was using already had complexity metrics embedded in them given that they belong to different “grade levels”. The goal became to extract those metrics by running the texts through a parser and then collecting statistics pertaining to those parses. This statistical approach would eliminate the need for an expert in Arabic linguistics and would simply rely on parsers and a corresponding Treebank. It would enable the system to get direct insight on the syntactic features of texts that ‘gatekeepers’ and program coordinators qualify as ‘complex’.

Of course the assumption here remains that the parses will be correct and that there won’t be any ambiguity. This assumption is shaky because no parser is capable of reaching
100% accuracy yet. For this reason, the tool will parse texts but also give the user the ability to modify the results of those parses in case he or she deems that they are ‘incorrect’. The user must thus be somewhat of an expert in linguistics (unlike myself).

2. The Stanford Parser

For this project, I chose to use the Stanford parser which is a statistical parser based on the Penn Arabic Treebank. It’s probabilistic in the sense that it uses “knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences” [33]. This parser adopts the tokenization system used in the Penn Arabic Treebank, and it now has a component that tokenizes the input accordingly. The segmentation is based on Buckwalter’s Arabic Morphological Analyzer [34], which separates all clitics from the morphemes they’re attached to, except for the determiner “Al”. The diacritics are dropped, which means we’re dealing with unvocalized Arabic. The parser also assumes the phrasal categories used in the Penn Arabic Treebank. The guidelines to those categories are provided as an appendix. It is worth noting that while the Stanford parser is based on the PATB which is constituency-based, it also produces dependency relations. This option is not yet optimal for Arabic though. I chose this parser because of its thorough documentation and because it is trained on the Penn Arabic Treebank which has the richest morphological and syntactic annotations.

3. The Texts

The texts used in this project, as previously described, are mostly of academic nature. The reason behind this choice is of course that no other texts would have better complexity indicators embedded in them, since the assumption is that texts assigned at school follow
‘complexity’ in syntactic structure among other things. The texts initially used were from the Syrian and Moroccan curriculum. The problem with the first one is that Syria went into a state of civil war, so the texts were no longer available online. However, I already had them in my possession, and given that the purpose of the project is purely research with no ulterior motive or business correlations, I kept using them. Concerning the Moroccan texts, I had to personally type them as I couldn’t find digital versions of the texts. I didn’t type many of them, given the time constraints but got around 22 texts to start working with. The third texts used in this project are excerpts from Al Kitaab [36] textbooks published by Georgetown University Press and used in the Arabic language program at Al Akhawayn University. I made sure to check with the International Studies program coordinator, Dr. Eric Ross, that using these resources was correct ethically and legally, and he gave me the green light, as long as I wasn’t reproducing the texts or receiving profit out of them, but simply using them for research. The fourth and last set of texts I am using are from the Internet, mostly from CNN’s webpage in Arabic. The reason behind this choice, is to compare the media’s standard “complex” structures of texts and those of the school.

4. The Statistics

The statistics the project is aiming to collect are mostly descriptive in nature, in the sense that the system only observes but doesn’t really process the information or makes it go through some mathematical equations. Thus, the system stores:

- The title of the text (or any indication to mark its uniqueness in the database);
- The number of sentences per text;
- The number of words per sentence;
• The parse tree for each sentence in the text, with an attribute indicating whether the parse was complete/incomplete, and correct/incorrect;

• Whether the text is fully vocalized (contains diacritics) or not (the basic assumption is that only low-level texts contain a lot of diacritics).

The system will then try to analyse this data and come up with patterns and models.

Chapter 4 – The Platform

1. Requirements Gathering

Given the nature of this project, the requirements process was iterative and changed constantly to accommodate the shifts in the approach. Therefore, the software engineering process adopted was incremental and relied on the results reached at each phase. The requirements were mostly drawn from meetings and conversations with my supervisor Dr. Cavalli based on her suggestions.

1. General Requirements

This final product should be independent and self-contained, in the sense that its implementation wouldn’t require installing third-party software or any plugins. All the
necessary libraries and extensions necessary to run the software should already be embedded in its implementation. The platform’s main goal is to be able to parse a set of sentences in Arabic and to produce correct parse trees to be stored in a database and then later retrieved and visually presented on a browser. The software should interact with the chosen parser and accompanying tools in such a way to provide reasonable parses for a given volume of texts. The software should allow the user to label a parse as correct and/or incomplete as an attempt to make up for the potential errors resulting from the occasional lack of accuracy in parsers.

2. Functional Requirements

In order to document this phase, I have chosen to follow the UML guidelines and to provide a use case diagram to show the different types of users the system supports and the actions they should be able to perform. The initial step consists of making the inventory of Actors (users) and Use Cases (all the possible scenarios carried out by users). In this project, the actors are Console User, Anonymous Web User, and Logged-In Web User. The first type is concerned with data entry and parsing Arabic text files in batches. This user doesn’t have to be an expert in Arabic NLP, and only has to provide a valid textfile name. The second type of user is somewhat passive in the sense that he or she doesn’t have the access privileges to manipulate the data but only to view it and try out the graphical tool. The last type is assumed to have background in NLP and enough expertise to be able to assert with confidence that a given parse tree is correct and/or incomplete. Here is a tentative Use Case Diagram summarizing these requirements.
3. Non-Functional Requirements

The system shall be in English, though it is targeting Arabic, because it’s the Lingua Franca of NLP and CL scholarship. It should be delivered by the end of the semester and should be robust enough as to ensure consistency in the database and avoid crashing randomly. The system shall be secure even though it’s only a prototype, and shouldn’t allow a user who isn’t logged in to label parses for example.

2. Technology Enablers

Research is constantly oriented towards open-source platforms for a better exchange of information and to remain separate from commercial products. Hence, a web application developed with Java components seems like the logical choice for this tool. The web is more accessible – assuming that the person has access to the internet (but this assumption stands for desktop applications as well given that in order to store data, one should be connected to the net). The user doesn’t have to download any piece of software and then install it on their
machine. The user also won’t be bound to his or her machine when trying to manipulate the tool, and can access all the texts he or she has inputted from any device with internet access. However, for better “batch processing” of texts, the project also consists of a simple java application that takes text files in Arabic, parses them and stores them in a database.

As mentioned earlier, I used the Java language instead of C# or other licensed languages, and the Java Enterprise framework given its layered approach which encourages consistency and more organized work. As for the IDE, I used Eclipse Luna release given that it has many APIs integrated already which facilitate consistency and debugging. I’m thus using the Maven plugins, Spring API, and Hibernate. For the server specificities, I am using Tomcat 8.0 and Apache. I initially planned on using Oracle DBMS, but due to my unfamiliarity with its framework and to the time constraints, I reverted to using MySQL which I have previously worked with. The subsequent sections will present in more detail the tools I used to develop both the Java and the web applications, along with the software engineering process.

3. Application Architecture

Given the nature of the project and for the reasons explained above, the application is actually comprised of two separate components linked by a database. The first component is the simple Java application which takes an Arabic text file as input, produces a parse tree, and stores it in a database. This component is not necessarily user-friendly because it is more concerned with processing files and not with the accuracy of the parses. The user doesn’t have to be an expert in linguistics and only has to provide a correct filename.

The second component is the web application through which a user with a certain background in linguistics can view parses, and judge whether they are correct and/or
complete. It is designed to give enough information to the visitor about the purpose of the
tool, which is why it also contains some of the sections from this report. The most important
component of the web application is the graphical tree viewer which enables the user to
visualize the parse trees in an interactive way. It’s very user friendly and was developed using
Javasctipt libraries which will be further discussed in subsequent sections. Both components
are linked through a MySQL database which stores the parses from both the java and web
applications. Below is a rough diagram of the architecture of the application.

4. Java Application

Despite not requiring a user interface, this component is the most important and vital in
this project. This is where the parsing takes place and where the system interacts with the
Stanford parser. Given that this portion interacts with both the console and the web
application, it will be described in detail. Before running any text in Arabic through the
Stanford parser, it has to be split into sentences first. The parser is capable of producing parse
trees for more than one sentence as long as they’re separated by a corresponding punctuation
mark. However, the parses become too large and cumbersome to interpret and read. For this
reason, the first step is to split the text into sentences and parse them separately. I used the
sentence splitter available through SAFAR, a platform for ANLP developed by the Ibtikarat research group in Rabat [37].

The second step is to run the sentences through the Stanford Segmenter given that the Stanford parser only uses white space to tokenize sentences. Arabic, being morphologically rich, has to be processed by a segmenter, otherwise the results of the parses would be “absurd” since the words won’t be properly separated. The output of this step can then be run through the Stanford parser, which takes sentences as input and returns a parse tree in the designated format, depending on output options. For this project, I chose the Penn Treebank output format using “oneline”, since it’s easier to encode as strings and to store in a database.

5. Web Application

Given that this project is research in nature, the web application doesn’t have to be particularly fancy as long as it is efficient and carries out the tasks it is designed for. However, given that this is a capstone project and that this is my last chance to showcase everything I’ve managed to learn during the last four years, I wanted to design an interface while keeping aesthetics in mind. Nobody likes gloomy research websites, so I experimented a little bit with CSS stylesheets and Javascript. The tool that took me most time was the graphical tree visualization tool which prompted me to learn Javascript and jQuery.

The basic assumption behind this component is that the user might want to look at the parse tree graphically rather than attempting to decipher the lisp-like sentence resulting from the parses. I thus had to convert and ‘parse’ those lisp-like sentences into a graph-like format which I could then provide as input to my two graph-drawing libraries D3 [38] and Dagre [39]. I parsed them using JSON and basic graph traversal algorithms using a stack. An example of the result is provided below.
The web tool, however, doesn’t only serve to input text and output a parse tree, but also contains all the theoretical information detailed in this report, and explains the motivation behind the project. The following screenshots provide an idea of what the website looks like.

Figure 5.1: Screenshot of the Tree Visualization Web Tool
Assessing Syntactic Complexity of Arabic

Spring 2015

Motivation

Arabic is the 5th most spoken language in the world!
This project is a contribution to the burgeoning scholarship pertaining to Arabic Natural Language Processing.

Approach

For this project, I chose to use the Stanford parser which is a statistical parser based on the Penn Arabic Treebank. It’s probabilistic in the sense that it uses “knowledge of language gained from hand-parsed sentences” to try to produce the most likely analysis of new sentences. This parser adopts the tokenization system used in the Penn Arabic Treebank and it now has a component that tokenizes the input accordingly. I don’t have to use software such as MADA to tokenize and transliterate. The segmentation is based on Buckwalter’s Arabic Morphological Analyzer, which separates all clitics from the morphemes they’re attached to, except for the determiner “AI”.

The Stanford Natural Language Processing Group

Stanford Arabic Parser FAQ

Questions

1. What tokenization of Arabic does the parser assume?
2. What character encoding do you assume?
3. What characters are encoded?
Assessing Syntactic Complexity of Arabic

Text: I

Title: For Later Retrieval

Description:

Type or Paste:

File input: Choisissez un fichier. Aucun fichier choisi.

Assessing Syntactic Complexity of Arabic

ولد الولد في مصر
6. Database Management System

The main reason a database is needed in the system is to store the parses related to each text and its sentences. Another function would be to store the user’s information so that he or she can always go back to the parses they’ve initiated. Here’s a breakdown of the design of the database.

The database has a table ‘user’ that basically stores information on the person using the system. Its columns retain information like name, age, and most importantly e-mail. Each user has an index that marks its uniqueness and which is used to link the parses to the users.

Another table is ‘text’, it is used to store information on the text as a whole, with a title and a description making it easier to retrieve later on. The way the system is designed is to keep sentences from the same text together for later retrieval and analysis. A text should have a source for better statistics, and it has attributes such as number of sentences and number of words.
The table ‘sentence’ is the most important because it’s at this level that the parse tree is stored. The latter is stored in its literal form and not the graphical one which is mostly presented for the user’s convenience. A sentence has a number of words, and it belongs to a certain text. I chose a rather easy design to the database because it doesn’t have to be intricate, otherwise it will hamper statistics collection and analysis.

Now that we’ve gone over all the components, here is a summary of the application’s structure.
Chapter 5 – Results

This platform represents a first step towards a statistically-driven analysis of Arabic grammatical and syntactic complexity. While collecting the statistics from the several texts I have parsed, I made sure to document each ‘unexpected trend’ in order to draw conclusions later. These are the most recurring observations:

- It’s easier to characterize “simple” and “very complex” texts, but it’s much more complicated to distinguish “intermediate” texts, because they’re often a mixture of both simple and complex sentences. In simple texts, which are assigned in lower grades, diacritics are often present throughout the entire file, the sentences are often very short, and don’t contain morphologically rich words. Passive voice is also avoided, and most of the syntactic structures are very basic such as Verb-Noun-Object. Very complex texts, typically assigned to the highest grades, are also easily distinguishable by the absence of diacritics, the length of the sentences, and the high frequency of subordination and coordination. For intermediate texts, there are no such clear signs, and regression should probably be modeled based on the parses stored in the database.

- The initial assumption in this project was that whatever makes the parser break is likely to break the user’s understanding and therefore should be considered a measure of complexity. However, I soon gave up on this assumption given that even when you provide the parser with an incorrect sentence, grammatically for instance, it still provided a parse because it’s statistical and relies on an existing Treebank to find the closest structure to the one provided. However, as I ran a great number of texts through the tool, I noticed that the parser did eventually break. The reason, however, was due to memory, or lack of it thereof. This often happened when the sentences
were too long, but it seemed related to processing power rather than to the parser itself. I believe that with a more performant machine, I wouldn’t have encountered this problem.

- Before using the two Javascript libraries Dagre and D3, I initially used an applet that I then embedded in the html file. The reason for this choice is that the Stanford Parser initially provides a Swing API for visualizing the trees and I thought of reusing it by embedding it in an applet instead of a frame. However, when I tried to display it in the browser, it simply wouldn’t pass the security requirements of Google Chrome and Mozilla Firefox. HTML5 made sure to label applets as obsolete and only allows applets which pass its security recommendations. I then decided to render the trees using Javascript which then required me to parse the lisp-like sentences into a graph-readable format. During the presentation, my supervisor Dr. Cavalli asked me about the possibility of this transformation being more useful in the collection of statistics. This is probably correct in the sense that the parentheses make it very hard to draw patterns and trends or to even query in the database. However, the transformation was made using JSON objects and looping through the sentence structure. Storing the parses using this method would require more than just a “String” attribute on a database.

- The next step is to use these statistics and parses to model regression or linear trends. Such a project would require extensive mathematical and algorithmic background. But before diving into it, the data stored in the database should be checked for accuracy.
Conclusion

This project was very challenging given its roots in both computer science and linguistics. Its completion took a lot of time and dedication, and taught me a lot about a field that used to be completely foreign to me: Computational Linguistics. This report was dedicated to justifying the approach adopted in the project which aims to characterize the complexity of the syntactic and grammatical structure of written Arabic, and to describing the software engineering process by which it was carried out. This report includes remarks from the first feedback I got from my supervisor Dr. Cavalli, and from my Capstone defense as well. As mentioned in the Results section, my work thus served as an initial step towards characterizing linguistic complexity in written modern standard Arabic which can be used by others who wish to work on the same topic. This was a great example of how Computer Science interacts with other fields such as Humanities and Linguistics more specifically to solve practical issues and create new technologies.
References


   <http://www.alsintl.com/blog/most-common-languages/>.


*Identification, investigation, and resolution*


[31] Habash, Nizar Y. Introduction to Arabic Natural Language Processing. Page 106


[37] "SAFAR." SAFAR. Web. 5 May 2015.


## Appendices

### Appendix A: Guidelines to the Penn TreeBank POS tag-set

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>dollar</td>
<td>$ - $ -- $ A$ C$ HK$ M$ NZ$ S$ U$ S$ USS $</td>
</tr>
<tr>
<td>&quot;</td>
<td>opening quotation mark</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>&quot;</td>
<td>closing quotation mark</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>(</td>
<td>opening parenthesis</td>
<td>( {</td>
</tr>
<tr>
<td>)</td>
<td>closing parenthesis</td>
<td>) }</td>
</tr>
<tr>
<td>,</td>
<td>comma</td>
<td>,</td>
</tr>
<tr>
<td>--</td>
<td>dash</td>
<td>--</td>
</tr>
<tr>
<td>.</td>
<td>sentence terminator</td>
<td>. ! ?</td>
</tr>
<tr>
<td>:</td>
<td>colon or ellipsis</td>
<td>: ...</td>
</tr>
<tr>
<td>CC</td>
<td>conjunction, coordinating</td>
<td>&amp; 'n and both but either et for less minus neither nor or plus so therefore times v. versus vs. whether yet</td>
</tr>
<tr>
<td>CD</td>
<td>numeral, cardinal</td>
<td>mid-1890 nine-thirty forty-two one-tenth ten million 0.5 one fortyseven 1987 twenty 79 zero two 78-degrees eighty-four IX '60s .025 fifteen 271.124 dozen quintillion DM2.000 ...</td>
</tr>
<tr>
<td>DT</td>
<td>determiner</td>
<td>all an another any both del each either every half la many much nary neither no some such that the them these this those</td>
</tr>
<tr>
<td>EX</td>
<td>existential there</td>
<td>there</td>
</tr>
<tr>
<td>FW</td>
<td>foreign word</td>
<td>gemeinschaft hund ich jeux habes Haementeria Herr K'ang-si vous lutshaw alai je joue objets salutaris fille quibusdam pas trop Monte terram fiche ou corporis ...</td>
</tr>
<tr>
<td>IN</td>
<td>preposition or conjunction, subordinating</td>
<td>astride among upon whether out inside pro despite on by throughout below within for towards near behind atop around if like until below next into if beside ...</td>
</tr>
<tr>
<td>JJ</td>
<td>adjective or numeral, ordinal</td>
<td>third ill-mannered pre-war regrettable oiled calamitous first separable ectoplastic battery-powered participatory fourth still-to-be-named multilingual multi-disciplinary ...</td>
</tr>
<tr>
<td>JJR</td>
<td>adjective, comparative</td>
<td>bleaker braver breezier briefer brighter brisker broader bumper busier calmer cheaper choosier cleaner closer colder commoner costlier cozier creamier crunchier cuter ...</td>
</tr>
<tr>
<td>JJS</td>
<td>adjective, superlative</td>
<td>calmest cheapest choicest classiest cleanest closest commonest corniest costliest crassest creepiest cruddiest cutest darkest deadliest dearest deepest dinkiest dinkiest ...</td>
</tr>
<tr>
<td>LS</td>
<td>list item marker</td>
<td>A A B B C C D E F First G H I J K One SP-44001 SP-44002 SP-44005 SP-44007 Second Third Three Two &quot; a b c d e first five four one six three two</td>
</tr>
<tr>
<td>MD</td>
<td>modal auxiliary</td>
<td>can cannot couldn't dare may might must need ought shall should shouldn't will would</td>
</tr>
<tr>
<td>NN</td>
<td>noun, common, singular or mass</td>
<td>common-carrier cabbage kmucke-duster Casino afghan shed thermostat investment slide humour tailoff slick wind hyena override subhumanity machinist ...</td>
</tr>
<tr>
<td>NNP</td>
<td>noun, proper, singular</td>
<td>Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos Oceanside Escobar Kremler Sawyer Cougar Yvette Ervin ODI Darryl CTCA Shannon A.K.C. Meltex Liverpool ...</td>
</tr>
<tr>
<td>NNPS</td>
<td>noun, proper, plural</td>
<td>Americans Americas Amharas Amityvilles Amusements Anarcho-Syndicalists Andalusiand Andes Anduses Angels Animals Anthony Antilles Antiques Apache Apaches Apocrypha ...</td>
</tr>
<tr>
<td>NNS</td>
<td>noun, common, plural</td>
<td>undergraduates Scotch a-brac products bodyguards facets coasts divestitures storehouses designs clubs fragrances averages subjectivists apprehensions muses factory-jobs ...</td>
</tr>
<tr>
<td>PDT</td>
<td>pre-determiner</td>
<td>all both half many quite such sure this</td>
</tr>
<tr>
<td>POS</td>
<td>genitive marker</td>
<td>'s</td>
</tr>
<tr>
<td>PRP</td>
<td>pronoun, personal</td>
<td>hers herself him himself hisself it itself me myself one oneself ours ourselves ownself self she thee theirs them themselves they thou thy us</td>
</tr>
<tr>
<td>PRPS</td>
<td>pronoun, possessive</td>
<td>her his mine my our ours their thy your</td>
</tr>
<tr>
<td>RB</td>
<td>adverb</td>
<td>occasionally unabasingly maddeningly adventurously professedly sturringly prominently technologically magisterially predominately swiftly fiscally pitilessly ...</td>
</tr>
<tr>
<td>RBR</td>
<td>adverb, comparative</td>
<td>further gloomier grander graver greater grimmer harder harsher healthier heavier higher however larger later leaner lengthier less-perfectly lesser lonelier longer louder lower more ...</td>
</tr>
<tr>
<td>RBS</td>
<td>adverb, superlative</td>
<td>best biggest bluest earliest farthest first furthest hardest heartiest highest largest least most nearest second tightest worst</td>
</tr>
<tr>
<td>RP</td>
<td>particle</td>
<td>aboard about across along apart around aside at away back before behind by crop down ever fast for forth from go high i.e. in into just later low more off on open out over per pie raising start teeth that through under unto up up-pp upon whole with you</td>
</tr>
<tr>
<td>SYM</td>
<td>symbol</td>
<td>% &amp; ' &quot; ( ) . + - . = &gt; @ A[f] U.S U.S.S.R. * * * * *</td>
</tr>
</tbody>
</table>
Appendix B: Growth of Arabic on Twitter