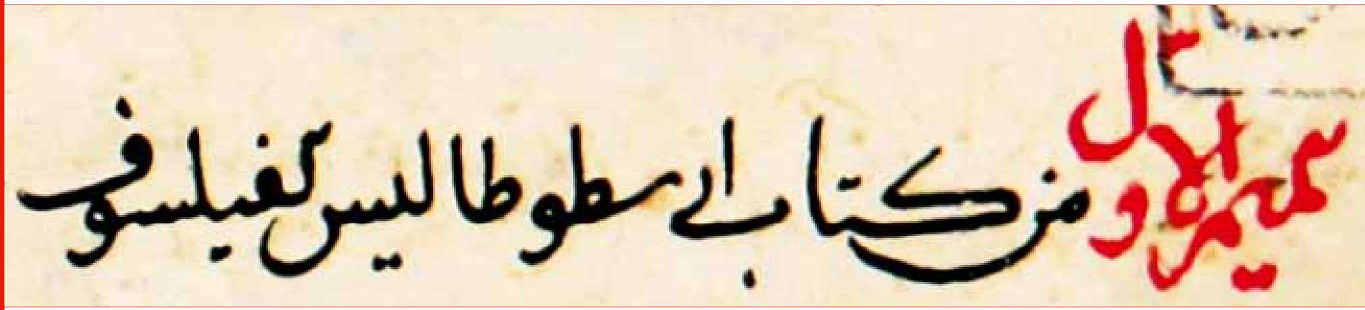
Studia graeco-arabica

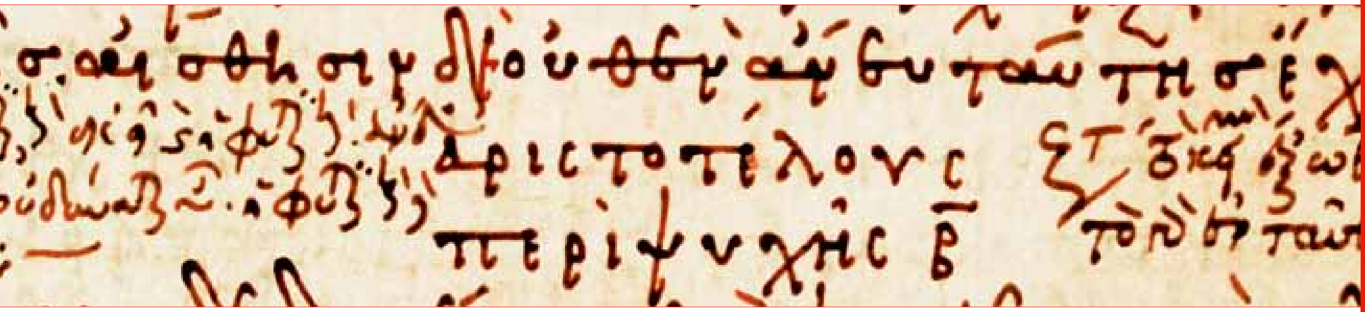


3

2013

Studia graeco-arabica

ISSN 2239-012X



With the support of the European Research Council

Pacini Editore

Studia graeco-arabica

The Journal of the Project
Greek into Arabic
Philosophical Concepts and Linguistic Bridges
European Research Council Advanced Grant 249431

3

2013

Published by
ERC Greek into Arabic
Philosophical Concepts and Linguistic Bridges
European Research Council Advanced Grant 249431

Web site: http://www.greekintoarabic.eu
Service Provider: Università di Pisa, Area Serra - Servizi di Rete di Ateneo

ISSN 2239-012X

Cover
Mašhad, Kitābḫāna-i Āsitān-i Quds-i Raḍawī 300, f. 1v
Paris, Bibliothèque Nationale de France, grec 1853, f. 186v

Publisher and Graphic Design
Pacini Editore
Via A. Gherardesca
56121 Ospedaletto (Pisa) - Italy

Printing
Industrie Grafiche Pacini

Studia graeco-arabica

3

2013

G2A Web Application

Istituto di Linguistica Computazionale “Antonio Zampolli”
Consiglio Nazionale delle Ricerche - Area della Ricerca di Pisa

Computational contributions for Arabic language processing

Part I. The automatic morphologic analysis of Arabic texts

Ouafae Nahli

Abstract
The aim of this paper is to describe our work on the project "Greek into Arabic", in which we faced some problems of ambiguity inherent to the Arabic language. Difficulties arose in the various stages of automatic processing of the Arabic version of Plotinus, the text which lies at the core of our project. Part I highlights the needs that led us to update the morphological engine AraMorph in order to optimize its morpho-syntactic analysis. Even if the engine has been optimized, a digital lexical source for better use of the system is still lacking. Part II presents a methodology exploiting the internal structure of the Arabic lexicographic encyclopaedia *Lisān al-ʿarab*, which allows automatic extraction of the roots and derived lemmas. The outcome of this work is a useful resource for morphological analysis of Arabic, either in its own right, or to enrich already existing resources.

1. Introduction

The tools for linguistic analysis build an advanced representation of the information content of the documents through text processing at different levels of complexity: morphological analysis, syntactical analysis, semantic interpretation and disambiguation. The main initial problem for computational analysis of a text is to establish the criteria aimed at identifying its basic unit, which is the word.

Words must be identified or tokenized automatically by a software for the processing of natural language. Tokenization is relatively simple for languages, like English, which use spaces to delimit words (space-delimited writing). In this case, the token can be defined simply as "any sequence of characters delimited by spaces", even though this definition leaves room for many exceptions. On the contrary, for languages with a continuous orthographic system (unsegmented writing) tokenization requires extremely complex algorithms.[1] The Arabic language is characterized by a writing system that lies between the two types. In fact, some graphic words in Arabic correspond to minimum linguistic units delimited by blank spaces. Other words result from a sequence of lexical units. Some functional words can be proclitic (e.g., conjunctions, prepositions, definite article) or enclitic (e.g., nominative pronouns or accusative/genitive pronouns). Therefore, tokenization requires additional segmentation based on morphological rules of compatibility, which permit the recognition of the various agglutinated morphemes.

[1] Ch.D. Manning - H. Schütze (eds.), *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge MA 1999, p. 125-9: "Normally, an early step of processing is to divide the input text into units called tokens where each is either a word or something else like a number or a punctuation mark. This process is referred to as tokenization (...) Many languages do not put spaces in between words at all, and so the basic word division algorithm of breaking on whitespace is of no use at all. Such languages include major East-Asian languages/scripts, such as Chinese, Japanese, and Thai. Ancient Greek was also written by ancient Greeks without word spaces. Spaces were introduced (together with accent marks, etc.) by those who came afterwards. In such languages, word segmentation is a much more major and challenging task".

2. Orthographic words in Arabic

The orthographic unit may represent a basic unit called the "minimum word", which corresponds to a lemma or to its inflected form, obtained as the result of a change in the internal vowels, and of the addition (or lack) of a prefix and/or suffix. Simple or compound clitics can be added to this minimum unit to obtain a 'maximum word'.[2] Table 1 provides an example of 'maximum word': وَبِمَكْتَبَاتِهِمْ /*wabimaktabātihim*/

Tab. 1. Example of the maximum word and its composition: وَبِمَكْتَبَاتِهِمْ /*wabimaktabātihim*/.

Maximum Word					
Composed Proclitic		Minimum Word			Enclitic
Proclitic 1	Proclitic 2	Stem	Suffix 1	Suffix 2	Enclitic
wa	*bi*	*maktab*	*āt*	*i*	*him*
Conjunction	Preposition	place of the action / *kataba*/ "to write"	Feminine plural	Genitive case	Genitive pronoun
and	in	libraries			"of" them
"and in their libraries"					

This 'maximum word' is an example to discuss the problems of tokenization in the Arabic language. There are in fact several levels of tokenization, depending on the depth of the linguistic analysis involved. At a first level, tokenization is primarily based on blank space and punctuation marks[3] to delimit the boundaries of the orthographic word ('key tokens'). The main token can be a minimum or a maximum word. After tokenization, which divides the orthographic words according to spaces and punctuation, the maximum word should be further divided into sub-tokens, whereby clitics, affixes and the stem are separated. At this stage, a software designed to extract tokens requires further morphological and syntactic information. There are, in fact, several rules that govern the combination of words with affixes and clitics. Therefore, tokenization is tightly connected with morphologic analysis[4] to:

- recognize the limits of sub-tokens within words;
- identify proclitics, minimum words and enclitics;
- identify prefixes, stems and suffixes within minimum words.

In addition to an algorithm that enables tokenization, the software needs:

- a lexicon[5] of proclitics and prefixes; a lexicon of enclitics and suffixes; a lexicon of lexical words;
- information on the morpho-syntactic relations that govern the compatibility relations between:

[2] J. Dichy, "Pour une lexicomatique de l'arabe: l'unité lexicale simple et l'inventaire fini des spécificateurs du domaine du mot", *Meta: Journal des traducteurs / Meta: Translators' Journal* 42 (1997), p. 291-306 (URI: http://id.erudit.org/iderudit/002564ar - DOI: 10.7202/002564ar).

[3] The token can also be a punctuation mark, or a multiword expression. Numbers are also considered as tokens. A list of all the punctuation marks and numbers must be reported in the system to delimit the tokens in the main text. See M.A. Attia, "Arabic Tokenization System", *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages, Common Issues and Resources*, The Association for Computational Linguistics, Stroudsburg PA 2007, p. 65-72.

[4] Attia, "Arabic Tokenization System", p. 65-72.

[5] R.L. Trask, *A Dictionary of Grammatical Terms in Linguistics*, Routledge, London 1993, p. 159, defines the lexicon as: "That part of the grammar of a language which includes the lexical entries for all the words and/or morphemes in the language and which may also include various other information, depending on the particular theory of grammar".

– prefix/stem/suffixe: for example, imperfective prefixes are compatible with verbal stem only; verbal stem is not compatible with the *tanwīn* symbol; imperfective prefixes are not compatible with the *tanwīn* symbol;

– proclitic/minimum word/enclitic: for example, prepositions are compatible with nouns but not with verbs; nouns are compatible with *tanwīn* symbol; prepositions are compatible with the *tanwīn* symbol;

- the order relations that permit the concatenation of proclitics, minimum words and enclitics, to form maximum words, for example:

interrogative particle + conjunction + future particle + [minimum verbal word] + accusative pronoun.

3. AraMorph: Buckwalter Morphologic Analyzer

The Buckwalter Arabic Morphological Analyzer, called AraMorph, was developed in 2002 by Tim Buckwalter, and is one of the most popular engines for morpho-syntactic analysis of the Arabic language. We will consider here only AraMorph version 1.0, which is available free of charge, together with its source code.[6] The components of AraMorph are the algorithm for morphological analysis and the data, which consist essentially of three lexicons and three compatibility tables used for checking the combinations among proclitics, stems and enclitics:

- Lexicon dictPrefixes,[7] containing 299 entries;
- Lexicon dictSuffixes,[8] containing 618 entries;
- Lexicon dictStems, containing 82158 entries representative of 38600 lemmas;
- Table AB to check compatibility between prefix and stem (1648 entries);
- Table BC to check compatibility between stems and suffix (1285 entries);
- Table AC to check compatibility between prefix and suffix (598 entries).

In addition to the standard encodings,[9] many researchers in Natural Language Processing use an orthographic transliteration.[10] For example, the morphological engine AraMorph is based on a transliteration table[11] made by Tim Buckwalter, after whom it is named.[12]

[6] "Buckwalter Arabic Morphological Analyzer Version 1.0" was produced by Linguistic Data Consortium (LDC), catalog number LDC2002L49 and ISBN 1-58563-257-0. The "Buckwalter Arabic Morphological Analyzer Version 1.0" is used for POS-tagging Arabic and it is free of charge as a web download distribution (http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49).

[7] Lexicon dictPrefixes is formed by concatenations of proclitics and prefixes.

[8] Lexicon dictSuffixes is formed by concatenations of suffixes and enclitics.

[9] The electronic form of digitized Arabic texts undergoes the Unicode standard for character encoding and internal representation of files; and standard TEI for encoding XML documents and for the structured representation of textual data.

[10] The transcription term denotes an orthography that characterizes the phonology or morpho-phonology of a language. The transliteration term, instead, use one-to-one orthographical symbols, mapping that customary language. See: N.Y. Habash, *Introduction to Arabic, Natural Language Processing*, Morgan & Claypool Publishers, Department of Computer Science, Toronto 2010, p. 20 (Synthesis Lectures on Human Language Technologies), DOI:10.2200/S00277ED1V01Y201008HLT010, ISBN: 9781598297966 ebook.

[11] See Habash, *Introduction to Arabic, Natural Language Processing*, p. 20-1: "The Buckwalter transliteration is a transliteration system that follows the standard encoding choices made for representing Arabic characters for computers, e.g., Unicode. The Buckwalter transliteration has been used in many NLP publications and in resources developed at the Linguistic Data Consortium (LDC). The main advantages of the Buckwalter transliteration are that it is a strict transliteration (i.e. one-to-one) and that it is written in ASCII characters, i.e., easily reproducible without special fonts".

[12] Buckwalter's transliteration table can be viewed at the Appendix 1.

The algorithm for morphologic analysis and Part-of-Speech (POS) assignment[13] is inserted in the AraMorph code and enables tokenization, word segmentation, dictionary consultation, checking of compatibility, and finally the analytic report. The code for analysis of the morphology is contained in a Perl script, mediated through Java, and for each token, it enables a listing of all the annotations considered as possible solutions, with the assignment of all the diacritical signs, boundaries of morphemes; in addition, for each token it enables POS that may match each segment.

Even though AraMorph is widely acknowledged as the best analyzer for the Arabic language, it has some weak points.[14] We have tried to identify and solve them, in order to optimize the results.

4. Limits of AraMorph and solutions adopted

a. Dilemma of the normalization of Arabic script

The problem arises because of the lack of consistency in the use of diacritics. To handle this problem, common practice in the automatic management of Arabic texts is to normalize the input text. In fact, AraMorph follows normalization procedures that permit the omission of all the diacritic symbols present in the text such as vowels, gemination symbols and the *hamza* symbol. With the justification that diacritics do not exist in Arabic texts, the analysis is done systematically on a non diacriticized form,[15] for example:

- in Arabic texts it is common to find *ʾalif-hamza* written without a symbol, hence it seems an *ʾalif*. Consequently, AraMorph proceeds with the letter *ʾalif* as if it is also an *ʾalif-hamza*;
- AraMorph does not take into account vowels and gemination symbol when they occur in the text; hence, it proposes all the possible solutions of diacriticization taking into account only the consonant body.

This approach generates erroneous segmentations which, in turn, are based on erroneous orthography. Thus, it has soon become evident that, even if the normalization improves recognition by solving input variability, the probability of ambiguity increases.[16]

Table 2, concerning analysis of the orthographic word واحد (transliterated "wAHd"), perfectly illustrates how the normalization approach adopted by AraMorph generates mistaken segmentations and wrong analyses.

In Table 2, original AraMorph does not distinguish between the *ʾalif* (transliterated "A") and the *hamza* (transliterated ">"). Hence, it proposes several solutions for the orthographic word واحد (transliterated "wAHd"), as an adjective, as a noun, and as a verb:[17] واحِد "wAHid" - وَأَحَد "wa>aHad" – وَأَحَدّ "wa>aHad~" - وَأحِدّ "wa>aHid~" - وَأحْد "wa>aHod" - وَأَحْدَ "wa>aHoda" – وَأحُدّ "wa>aHud~" - وأحِدّ "wa>aHid~" - وَأحَّدَ "wa>aH~ada" – وَأحَدَّ "wa>aHad~a".

[13] For the table of the grammar categories see the Appendix 2.

[14] M.A. Attia, *Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation* (Ph.D. Thesis), University of Manchester, Manchester 2008, p. 35-9.

[15] T. Buckwalter, "Issues in Arabic Orthography and Morphology Analysis", in A. Farghaly - K. Megerdoomian (eds.), *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, Association for Computational Linguistics, Stroudsburg PA 2004, p. 31-4.

[16] A. Farghaly - K. Shaalan, "Arabic Natural Language Processing: Challenges and Solutions", *Journal ACM Transactions on Asian Language Information Processing TALIP* 8 (2009), p. 1-22 (Doi: 10.1145/1644879.1644881).

[17] ADJ: Adjective – PVSUFF_SUBJ: Perfective Verb Suffix _Subject - 3MS: 3rd Masculine Singular.

Tab. 2. Analysis output of the orthographic word واحد transliterated “wAHd” by original AraMorph.

```
واحد wAHd   ADJ                 → wAHid=ADJ+
واحد wAHd   NOUN                → wAHid=NOUN+
واحد wAHd   NOUN                → wa=CONJ+>aHad=NOUN+
واحد wAHd   NOUN                → wa=CONJ+>aHad~=NOUN+
واحد wAHd   NOUN_PROP           → wa=CONJ+>aHad=NOUN_PROP+
واحد wAHd   VERB_IMPERFECT      → wa=CONJ+>a=IV1S+Hid~=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT      → wa=CONJ+>a=IV1S+Hod=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT      → wa=CONJ+>a=IV1S+Hoda=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT      → wa=CONJ+>a=IV1S+Hud~=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT      → wa=CONJ+>u=IV1S+Hid~=VERB_IMPERFECT+
واحد wAHd   VERB_PERFECT        → wa=CONJ+>aH~ad=VERB_PERFECT+a=PVSUFF_SUBJ:3MS+
واحد wAHd   VERB_PERFECT        → wa=CONJ+>aHad~=VERB_PERFECT+a=PVSUFF_SUBJ:3MS+
```

By contrast, the upgraded system takes vowels and other symbols into account when they occur in the text. The proposed solutions are validated by the “AraMorph Vocalic Filter” component, developed by Federico Boschetti, which (i) assesses the compatibility of the vowel structure of the proposed solution with the form actually occurring in a given text; (ii) allows the user to enable or disable the formalized rules using regular expressions in order to check the orthography of *hamza* and the digitization of other diacritics. The following table illustrates a regular expression aimed at controlling the orthography of *hamza*.

Tab. 3. Example of a regular expression.

Regular expression	Allowed	Not allowed
([>]) [>A]	*hamza* أ (transliterated “>”) can be written also as ʾ*alif* ا (transliterated “A”)	*hamza* أ “>” cannot be changed into ʾ*alif* “A”

Table 4 shows the results of analysis of the orthographic word واحد “wAHd” when the regular expression is not allowed; therefore, the engine can not replace the *hamza* with the ʾ*alif* in the proposed solutions. At the same time, these solutions are filtered by the “AraMorph Vocalic Filter” system which compares them with the form occurring in the text. Hence, all the proposed solutions containing the *hamza* are eliminated and eventually, only one solution is accepted: واحِد /*wāḥid*/ “wAHid”, that can be either a name or an adjective.

Tab. 4. Analysis output of the word واحد “wAHd” using the improved engine.

```
واحد wAHd   ADJ               @@@→   wAHid=ADJ+
واحد wAHd   NOUN              @@@→   wAHid=NOUN+
واحد wAHd   NOUN              ###→   wa=CONJ+>aHad=NOUN+
واحد wAHd   NOUN              ###→   wa=CONJ+>aHad~=NOUN+
واحد wAHd   NOUN_PROP         ###→   wa=CONJ+>aHad=NOUN_PROP+
واحد wAHd   VERB_IMPERFECT    ###→   wa=CONJ+>a=IV1S+Hid~=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT    ###→   wa=CONJ+>a=IV1S+Hod=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT     ###→    wa=CONJ+>a=IV1S+Hoda=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT     ###→    wa=CONJ+>a=IV1S+Hud~=VERB_IMPERFECT+
واحد wAHd   VERB_IMPERFECT     ###→    wa=CONJ+>u=IV1S+Hid~=VERB_IMPERFECT+
واحد wAHd   VERB_PERFECT       ###→    wa=CONJ+>aH~ad=VERB_PERFECT+a=PVSUFF_SUBJ:3MS+
واحد wAHd   VERB_PERFECT       ###→    wa=CONJ+>aHad~=VERB_PERFECT+a=PVSUFF_SUBJ:3MS+
```

b. Updating of lexicons

Other deficiencies of analysis are due to the insufficiency of AraMorph dictionaries. Analysis of the word كِتاب /*kitāb*/ (transliterated "kitAb") in Table 5 shows some of these weaknesses.

Tab. 5. Analysis output of the word كِتاب (transliterated "kitAb") by the original AraMorph.

```
كتاب   kitAb     NOUN   kitAb=NOUN+
كِتاب   kitAb     NOUN   kut~Ab=NOUN+

كِتاب   >kitAb    POS:???   VOC:???MORPH:???
```

In Table 5, original AraMorph does not take into account the vowel /i/ that appears in word كتاب (kitAb) and suggests two solutions كتاب /*kitāb*/ "book" (transliterated "kitAb") and كُتّاب /*kuttāb*/ "authors" (transliterated "kut~Ab"). Furthermore, it does not specify syntactic cases. Even the interrogative particle at the beginning of the same name أكتاب /*ʾakitāb*/ (transliterated ">akitAb") is not recognized and its analysis gives no results. In order to make up for some AraMorph weaknesses, it has been necessary to proceed with the creation of new grammatical categories (POS) and with the updating of lexicons and compatibility tables, for example:

- addition of some missing particles such as interrogative particle;
- treatment of nominal declension and verbal inflection when it is expressed by a vowel.

After this upgrade of the system, analysis of the same word in Table 6 is more complete. The system takes into account the extant vowel and proposes the correct solution كتاب /*kitāb*/ "book" with all grammatical cases.[18] The interrogative particle is recognized, the analysis of the noun (كتاب) that follows it proceeds according to compatibility charts (for example, the interrogative particle is not compatible with the genitive case).

Fig. 6. Analysis output after addition of the interrogative particle and declension cases.

```
كِتاب   kitAb NOUN      kitAb=NOUN+a=CASE_DEF_ACC+
كِتاب   kitAb NOUN      kitAb=NOUN+i=CASE_DEF_GEN+
كِتاب   kitAb NOUN      kitAb=NOUN+K=CASE_INDEF_GEN+
كِتاب   kitAb NOUN      kitAb=NOUN+N=CASE_INDEF_NOM+
كِتاب   kitAb NOUN      kitAb=NOUN+u=CASE_DEF_NOM+

كِتاب   >kitAb NOUN     >a=INTER+kitAb=NOUN+a=CASE_DEF_ACC+
كِتاب   >kitAb NOUN     >a=INTER+kitAb=NOUN+N=CASE_INDEF_NOM+
كِتاب   >kitAb NOUN     >a=INTER+kitAb=NOUN+u=CASE_DEF_NOM+
```

[18] DEF: Definite, INDEF: Indefinite, ACC: Accusative, GEN: Genitive, NOM: Nominative, INTER: Interrogative particle.

- Precision in the treatment of irregular verbs (weak verbs)

Weak verbs are verbs whose roots contain one or more letters *yāʾ* or/and *wāw*, for example, رَجَا/ *rajā*/ "he hoped" (from the root rjw), بَكَى /*bakā*/ "he planted" (from the root bky) and لقيَ /*laqiya*/ "he encountered" (from the root lqy). These verbs can lose the letter yāʾ or wāw in the conjugation, for example in jussive mood: يَلقَ /*yalqa*/ (transliterated "yaloqa"), يَبْكِ /*yabki*/ (transliterated "yaboki") and يَرْجُ /*yarju*/ (transliterated "yaroju").

In the original AraMorph, weak verbal lemmas have been grouped with the same grammatical label. Table 7 is an illustration of jussive mood analysis of these verbs by the original AraMorph.[19]

Tab. 7. In the output, the analyses specify that verbs are imperfect with no other indications.

يلقَ	ylqa	VERB_IMPERFECT	ya=IV3MS+loqa=VERB_IMPERFECT+
يبكِ	ybki	VERB_IMPERFECT	ya=IV3MS+bok=VERB_IMPERFECT+
يرجُ	yrju	VERB_IMPERFECT	ya=IV3MS+roj=VERB_IMPERFECT+

When the proposed analyses are filtered by the component "AraMorph Filter vowel", which compares them with the attested forms, some suggestions are rejected (Tab. 8).

Tab. 8. Validation of proposed analyses by "AraMorph Filter vowel".

يلقَ	ylqa	VERB_IMPERFECT	@@@→yaloqa SI:	ya=IV3MS+loqa=VERB_IMPERFECT+
يبكِ	ybki	VERB_IMPERFECT	###→ yabok NO:	ya=IV3MS+bok=VERB_IMPERFECT+
يرجُ	yrju	VERB_IMPERFECT	###→ yaroj NO:	ya=IV3MS+roj=VERB_IMPERFECT+

In Table 8, we can see that the proposed analysis of the verb يَلْقَ /yalqa/ is accepted because it contains the final vowel. Instead, for the verbs يَبْكِ /yabki/ and يَرْجُ /yarju/, the proposed analyses are rejected because they do not contain the final vowel which is missing in the dictionaries. This implies the addition of new suffixes and flexion codes, and further updating of compatibility tables. Therefore the analysis of weak verbs is better defined in Table 9.[20]

Tab. 9. Analysis output of weak verbs obtained with improved tool.

يلقَ	ylqa	VERB_IMPERFECT	@@@→yaloqa SI:	ya=IV3MS+loq=VERB_IMPERFECT+a=IVSUFF_MODD:JS+
يبكِ	ybki	VERB_IMPERFECT	@@@→yaboki SI:	ya=IV3MS+bok=VERB_IMPERFECT+i=IVSUFF_MOOD:JS+
يبكِ	ybki	VERB_IMPERFECT	@@@→yuboki SI:	yu=IV3MS+bok=VERB_IMPERFECT+i=IVSUFF_MOOD:JS+
يرجُ	yrju	VERB_IMPERFECT	@@@→yaroju SI:	ya=IV3MS+roj=VERB_IMPERFECT+u=IVSUFF_MOOD:JS+

- Optimization of the analysis which eliminates over-generation, for instance over-generation due to multiple entries for the same lemma which has the same flexion code. For example, the lemma واحِد /*wāḥid*/

[19] IV3MS: Imperfective verb, 3rd person masculine singular.

[20] IV3MS: Imperfective verb, 3rd person masculine singular, IVSUFF_MOOD:JS (Imperfective verb suffix - Jussive mood).

"one; single" can be adjective or noun and it has two entries in original AraMorph with the same code N-ap.[21] See its analysis in Table 10: the original engine offers five declination cases for both the adjective and the name.

Tab. 10. Increase of the proposed solutions caused by different entries of the same lemma.

واحد wAHd	ADJ	@@@→wAHida	wAHid=ADJ+	a=CASE_DEF_ACC+
واحد wAHd	ADJ	@@@→wAHidi	wAHid=ADJ+	i=CASE_DEF_GEN+
واحد wAHd	ADJ	@@@→wAHidK	wAHid=ADJ+	K=CASE_INDEF_GEN+
واحد wAHd	ADJ	@@@→wAHidN	wAHid=ADJ+	N=CASE_INDEF_NOM+
واحد wAHd	ADJ	@@@→wAHidu	wAHid=ADJ+	u=CASE_DEF_NOM+
واحد wAHd	NOUN	@@@→wAHida	wAHid=NOUN+	a=CASE_DEF_ACC+
واحد wAHd	NOUN	@@@→wAHidi	wAHid=NOUN+	i=CASE_DEF_GEN+
واحد wAHd	NOUN	@@@→wAHidK	wAHid=NOUN+	K=CASE_INDEF_GEN+
واحد wAHd	NOUN	@@@→wAHidN	wAHid=NOUN+	N=CASE_INDEF_NOM+
واحد wAHd	NOUN	@@@→wAHidu	wAHid=NOUN+	u=CASE_DEF_NOM+

In the updated system, for the same lemma which has the same flexion code, the multiple entries are traced back to a single entry with a unique POS, which indicates that the lemma can be either a noun or an adjective. Consequently, in Table 11, there are less analyses which are suggested. Therefore, in the first step, morphological and syntactical analysis is accomplished more quickly. The disambiguation of the word (adjective or name) will follow according to the context in a next step.

Tab. 11. The grammatical category (ADJ-NOUN) indicates that the word can be either an adjective or a noun.

واحد wAHd	ADJ-NOUN	@@@→wAHida	wAHid=ADJ-NOUN+a=CASE_DEF_ACC+
واحد wAHd	ADJ-NOUN	@@@→wAHidi	wAHid=ADJ-NOUN+i=CASE_DEF_GEN+
واحد wAHd	ADJ-NOUN	@@@→wAHidK	wAHid=ADJ-NOUN+K=CASE_INDEF_GEN+
واحد wAHd	ADJ-NOUN	@@@→wAHidN	wAHid=ADJ-NOUN+N=CASE_INDEF_NOM+
واحد wAHd	ADJ-NOUN	@@@→wAHidu	wAHid=ADJ-NOUN+u=CASE_DEF_NOM+

c. First results

Within the context of the project *Greek into Arabic*, we started to test our upgrade of AraMorph by analyzing the text edited by ʿA. Badawī (1955 and 1966), namely the Arabic version of parts of Plotinus' *Enneads*.[22] The morphological engine has found that the entire text is composed of 60253 words related to 9503 orthographic forms. AraMorph in its original version failed to recognize 832 orthographic forms (NOT FOUND) and offered 27166 analyses for the forms recognized (i.e. 8671 forms). After the upgrade, the engine recognizes all the forms represented in the entire text, but it proposes 55497 analyses. These results are the consequence of two linguistic phenomena.

First, many (if not all) words are ambiguous, since they do not contain all the vowels and consequently they can be related to various lemmas. Second, the insertion of all the cases of nominal declension and verbal inflection increases the solutions proposed by the engine. Therefore, the system generates more than one analysis for every single form.

[21] The code N-ap allows all inflexions except for the plural masculine.

[22] ʿA. Badawī, *Aflūṭīn ʿinda l-ʿArab*, Dār al-Nahḍat al-ʿarabiyya, Cairo 1955, 1966.

Let us consider the following sentence (p. 58.13-14 Badawī):

وربما كان الشيء الذي تريد الصناعة أن تأخذ رسمه وصنعته – وجدته ناقصًا

Perhaps the object whose image and skills the art wants to copy - it finds it incomplete.

The orthographic form وجدته (transliterated "wjdth") corresponds to "it finds it" in the context of the above mentioned sentence, but it is a un-vocalized form and could indicate several meanings in different contests. As shown in Table 12, in fact, the above mentioned orthographic form could be linked to different lemmas.[23] Moreover, the engine proposes cases of declination for every nominal lemma, and inflected forms for every verbal lemma.

Tab. 12. Output analyses of the orthographic form وجدته "wjdth".

```
وجدته wjdth NOUN→  wa=CONJ+jad~=NOUN+ap=NSUFF_FEM_SG+a=CASE_DEF_ACC+hu=POSS_PRON_3MS+
وجدته wjdth NOUN→  wa=CONJ+jad~=NOUN+ap=NSUFF_FEM_SG+i=CASE_DEF_GEN+hu=POSS_PRON_3MS+
وجدته wjdth NOUN→  wa=CONJ+jad~=NOUN+ap=NSUFF_FEM_SG+u=CASE_DEF_NOM+hu=POSS_PRON_3MS+

وجدته wjdth NOUN→  wa=CONJ+jid~=NOUN+ap=NSUFF_FEM_SG+a=CASE_DEF_ACC+hu=POSS_PRON_3MS+
وجدته wjdth NOUN→  wa=CONJ+jid~=NOUN+ap=NSUFF_FEM_SG+i=CASE_DEF_GEN+hu=POSS_PRON_3MS+
وجدته wjdth NOUN→  wa=CONJ+jid~=NOUN+ap=NSUFF_FEM_SG+u=CASE_DEF_NOM+hu=POSS_PRON_3MS+

وجدته wjdth VERB_PERFECT→  wa=CONJ+jad=VERB_PERFECT+ato=PVSUFF_SUBJ:3FS+hu=PVSUFF_DO:3MS+

وجدته wjdth VERB_PERFECT→   wajad=VERB_PERFECT+ato=PVSUFF_SUBJ:3FS+hu=PVSUFF_DO:3MS+
وجدته wjdth VERB_PERFECT→   wajad=VERB_PERFECT+ota=PVSUFF_SUBJ:2MS+hu=PVSUFF_DO:3MS+
وجدته wjdth VERB_PERFECT→   wajad=VERB_PERFECT+oti=PVSUFF_SUBJ:2FS+hu=PVSUFF_DO:3MS+
وجدته wjdth VERB_PERFECT→   wajad=VERB_PERFECT+otu=PVSUFF_SUBJ:1S+hu=PVSUFF_DO:3MS+

وجدته wjdth VERB_PERFECT→  wa=CONJ+jud=VERB_PERFECT+ota=PVSUFF_SUBJ:2MS+hu=PVSUFF_DO:3MS+
وجدته wjdth VERB_PERFECT→  wa=CONJ+jud=VERB_PERFECT+oti=PVSUFF_SUBJ:2FS+hu=PVSUFF_DO:3MS+
وجدته wjdth VERB_PERFECT→  wa=CONJ+jud=VERB_PERFECT+otu=PVSUFF_SUBJ:1S+hu=PVSUFF_DO:3MS+
```

Analysis of the orthographic form وجدته (transliterated "wjdth") in Table 12 illustrates the typical example of Arabic ambiguity and the difficulty of its automatic analysis. Ambiguity is an important problem that any automated analysis procedure must face. It would be possible to analyze all the proposed alternatives for the orthographic form وجدته (transliterated "wjdth") in Figure 10 (and the 55497 proposed alternatives for the entire text), but we have found more effective to proceed progressively step by step.

At a first level of analysis, the program is able to deactivate the nominal declension and the verbal inflexion when it is expressed with a vowel. Therefore the approach in Table 13 begins by establishing the exact lemma to which the orthographic form وجدته (transliterated "wjdth") is associated and

[23] CONJ: Conjunction; NSUFF_FEM_SG : Nominal suffix; feminine singular; CASE_DEF_ACC: Definite accusative; CASE_DEF_GEN: Definite genitive; CASE_DEF_NOM: Definite nominative; POSS_PRON: Personal suffix; PVSUFF_SUBJ: Perfective verb suffix _ subject; PVSUFF_DO: Perfective verb suffix_ direct object; 1S: 1st Person singular; 2MS: 2nd person masculine singular; 2FS: 2nd person feminine singular; 3FS: 3rd person feminine singular.

by eliminating the wrong lemmas proposed. Therefore the orthographic word وجدته (transliterated "wjdth") is associated to the lemma /*wajada*/ "has found".

Tab. 13. Determining the exact lemma for the attested form.

NO	وجدته	wjdth	NOUN	→wa=CONJ+jad~=NOUN+ap=NSUFF_FEM_SG+hu=POSS_PRON_3MS+
NO	وجدته	wjdth	NOUN	→wa=CONJ+jid~=NOUN+ap=NSUFF_FEM_SG+hu=POSS_PRON_3MS+
NO	وجدته	wjdth	VERB_PERFECT	→wa=CONJ+jad=VERB_PERFECT+t=PVSUFF_SUBJ+hu=PVSUFF_DO:3MS+
YES	وجدته	wjdth	VERB_PERFECT	→wajad=VERB_PERFECT+t=PVSUFF_SUBJ+hu=PVSUFF_DO:3MS+
NO	وجدته	wjdth	VERB_PERFECT	→wa=CONJ+jud=VERB_PERFECT+t=PVSUFF_SUBJ

Thus, the engine proposes for the entire text 22529 morphological analyses which show the possible lemmas for every orthographic form.[24] At this point, we narrow our focus on linking the attested form to the exact lemma. Having established the lemma corresponding to the form, the declension of that lemma is allowed, in the next step, for the syntactic study. In Table 14, the orthographic word وجدته "wjdth" is related to the lemma /*wajada*/ "has found", 3rd person singular feminine.

Tab. 14. Determining the exact inflected form for the attested form.

YES	وجدته	wjdth	VERB_PERFECT	→wajad=VERB_PERFECT+ato=PVSUFF_SUBJ:3FS+hu=PVSUFF_DO:3MS+
NO	وجدته	wjdth	VERB_PERFECT	→wajad=VERB_PERFECT+ota=PVSUFF_SUBJ:2MS+hu=PVSUFF_DO:3MS+
NO	وجدته	wjdth	VERB_PERFECT	→wajad=VERB_PERFECT+oti=PVSUFF_SUBJ:2FS+hu=PVSUFF_DO:3MS+
NO	وجدته	wjdth	VERB_PERFECT	→wajad=VERB_PERFECT+otu=PVSUFF_SUBJ:1S+hu=PVSUFF_DO:3MS+

5. Conclusion

This paper has presented an adaptation and improvement of the extant techniques of morphological analysis. Using AraMorph, which is an open-source software, we have focused our attention on sharpening lexicons and compatibility tables to solve cases of morphological ambiguity.

Even when the dictionaries of affixes (dictPrefixes and dictSuffixes) and compatibility tables are implemented to become complete, AraMorph does not support the processes of generation and derivation; therefore it does not contain any structure that allows the automatic insertion of new entries. The new lemmas must be manually entered, with all the possible stems that permit the inflection. Therefore in order to compile the dictionary of lexical words (dictStems) a time-consuming process is needed, because the dictionary is an open class. A lexical coverage as wide as possible (even if completeness cannot actually be achieved) is highly desirable, because the first task of the computer during text analysis is to refer to the dictionary; if the word is not present in it, the task fails. To work around this limitation, procedures for automatic extraction from the Arabic lexicographic encyclopaedia *Lisān al-ʿarab*[25] have been designed. These procedures are illustrated in the second part of this paper. The results of the extraction process can be used to enrich automatically the dictionary of stems (dictStems) in AraMorph.

[24] This result (i.e. 22529 analyses for 9503 attested forms in the text) is comparable with the result obtained by the original AraMorph that proceeds without taking into account the final vowels (i.e. 27166 analyses only for 8671 attested forms).

[25] The *Lisān al-ʿArab* (لسان العرب) is among the best known complete Arabic dictionaries and was compiled by Ibn Manẓūr in 1290. See Ibn Manẓūr, *Lisān al-ʿarab*, Dār al-kutub al-ʿilmiyya, Beirut 2005.

Appendix 1. Buckwalter's transliteration table

Arabic letter	phonetic transcription	Buckwalter's transliteration	Arabic letter	phonetic transcription	Buckwalter's transliteration
ء	ʾ	ʾ	ط	ṭ	T
أ	ʾ	>	ظ	ẓ	Z
إ	ʾ	<	ع	ʿ	E
ؤ	ʾ	&	غ	ġ	g
ئ	ʾ	}	ف	f	f
آ	ʾā	|	ق	q	q
ا	ā	A	ك	k	k
ى	ā	Y	ل	l	l
ب	b	b	م	m	m
ت	t	t	ن	n	n
ة	t	p	ه	h	h
ث	ṯ	v	و	w	w
ج	ǧ	j	ي	y	y
ح	ḥ	H	fatḥa	a	a
خ	ḫ	x	ḍamma	u	u
د	d	d	kasra	i	i
ذ	ḏ	*	Tanwīn fatḥa	/an/	F
ر	r	r	Tanwīn ḍamma	/un/	N
ز	z	z	Tanwīn kasra	/in/	K
س	s	s	Gemination symbol		~
ش	š	$	sukūn		o
ص	ṣ	D			
ض	ḍ	T			

Appendix 2. Grammatical categories (Parts of Speech)

Grammatical categories used by original AraMorph can be seen at:
http://www.nongnu.org/AraMorph/english/grammatical_categories.html

In addition to the grammatical categories used by the original AraMorph, we have created new categories for better analysis of the Arabic language. Some of the new categories used in this article can be seen below:

INTER	Interrogative particle
PVSUFF_SUBJ:3FS	Perfective verb suffix_Subject: 3rd person feminine singular
PVSUFF_SUBJ:2FS	Perfective verb suffix_Subject: 2nd person feminine singular
PVSUFF_SUBJ:2MS	Perfective verb suffix_Subject: 2nd person masculine singular
PVSUFF_SUBJ:1S	Perfective verb suffix_Subject: 1st person singular.
IVSUFF_MOOD:I	Imperfective Verb Suffix - Indicative mode
IVSUFF_MOOD:JS	Imperfective Verb Suffix - jussive mode
IVSUFF_MOOD:JS	Imperfective Verb Suffix - subjunctive mode
CASE_DEF_ACC	Accusative Case – Definite Noun
CASE_DEF_GEN	Genitive Case – Definite Noun
CASE_DEF_NOM	Nominative Case – Definite Noun
ACC_INDEF_ACC	Accusative Case – Indefinite Noun
CASE_INDEF_GEN	Genitive Case – Indefinite Noun
CASE_INDEF_NOM	Nominative Case – Indefinite Noun

Finito di stampare nel mese di settembre 2013
presso le Industrie Grafiche della Pacini Editore S.p.A.
Via A. Gherardesca • 56121 Ospedaletto • Pisa
Tel. 050 313011 • Fax 050 3130300
www.pacinieditore. it