Building a Shallow Arabic Morphological Analyzer in One Day

Kareem Darwish,
University of Maryland, College Park, MD 20742
kareem@glue.umd.edu

Abstract:

The paper presents a rapid method of developing a shallow Arabic morphological analyzer. The analyzer will only be concerned with generating the possible roots of any given Arabic word. The analyzer is based on automatically derived rules and statistics. For evaluation, the analyzer is compared to a commercially available Arabic Morphological Analyzer.

1. Introduction:

Due to the morphological complexity of the Arabic language, Arabic morphology has become an integral part of many Arabic Information Retrieval (IR) systems. Some of the Arabic IR systems that use morphology include Swift [1] and electronic publishing software developed by Sakhr that contain IR components (such as the Encyclopedia of Jurisprudence) [2]. Arabic IR studies have shown that the use of Arabic roots as indexing terms substantially improves the retrieval effectiveness over the use of words as index terms [3] [4] [5].

Arabic words are divided into three types: noun, verb, and particle [6]. Nouns and verbs are derived from a closed set of around 10,000 roots [7]. The roots are commonly three or four letters and are rarely five letters. Arabic nouns and verbs are derived from the roots by applying a template to the root to generate a *stem* and then introducing prefixes and suffixes. Figure 1 shows some templates for 3 letter roots. Figure 2 shows some of the possible prefixes and suffixes and their perspective meaning. The number of unique Arabic words (or surface forms) is estimated to be 6 x 10^{10} words [8]. Figure 3 shows some of the words that maybe generated from the root "ktb". Refer to figure I in the Appendix for the mapping between the Arabic letters and their Latin representations.

Further, a word may be derived from several different roots. For example the word "AymAn" can be derived from six different roots. Figure 4 shows possible roots for the word AymAn and the meaning of the word based on each. For the purposes of this paper, a word is any Arabic surface form, a stem is a word without any prefixes or suffixes, and a root is a word without any prefixes, suffixes, or infixes. Root are the units from which words are derived. However, often irregular roots, which contain double or weak letters, lead to stems and words that have letters from the root deleted or replaced.

Building large-scale morphological analyzers is typically a laborious and time-consuming task. For example, MORPHO3 was developed by RDI in 3 man/years [8]. However, in this paper, we will present a quick method for performing shallow morphological analysis, which entails finding the roots of words. The method is based on collecting statistics from word-root pairs:

1. to build morphological rules for deriving roots from words,

2. to construct a list of prefixes and suffixes, and
3. to estimate the probability that a rule will be used or a prefix or suffix will be seen.

This analyzer will possibly be the first cross-platform freely-distributable analyzer for Arabic.

Section 2 will provide background on some of the published research in Arabic Morphology. Section 3 will provide a description of the proposed shallow morphological analyzer. Section 4 will evaluate the proposed analyzer and will address some of the shortcomings of the system.

XXX	ktb (wrote)	XXAX	ktAb (book)	XAXX	kAtb (writer)
mXXwX	mktwb (something written)	XXAXyX (middle letter doubled)	ktAtyb (Qur'an schools)	XXwX	ktwb (skilled writer)

Figure 1: Some templates to generate stems from roots with an examples from the root "ktb"

Examples of prefixes					
w	and	f	Then	Al	the
k	like	l	To	wAl	and the
Examples of suffixes					
h	his	hm	Their	hA	her
k	your (singular)	km	your (plural)	y	my

Figure 2: some examples of prefixes and suffixes and their meanings

ktb	He wrote	yktb	He is writing	Aktb	Write
kAtb	Writer	ktAb	Book	ktAbh	His book
wktAbh	And his book	ktAbhm	Their book	ktb	books

Figure 3: some of the words that can derived from the root form "ktb"

Root	Meaning	pronunciation
Amn	Peace	Eman
Aym	two poor people	Ayyiman
mAn	will he give support	Ayama'nu
ymm	will both of you seek	Ayyammani
ymn	Covenant	Ayman
ymA	will they point to	Ayama'na

Figure 4: the possible roots for the word AymAn along with the meaning and pronunciation of the word based on the different roots.

2. *Background:*

Significant work has been done in the area of the Arabic morphological analysis. The two main approaches to the problem in Arabic:

1. Use language rules which are programmed in a finite state transducer (FST). Koskenniemi proposed a two-level morphology system for language morphology which led to Antworth's two-level morphology system PC-KIMMO [9] [19]. Later, Beesley and

Buckwalter developed an Arabic morphology system ALPNET that uses a slightly enhanced implementation of PC-KIMMO [10]. Currently, this Arabic morphological analyzer is owned by Xerox and uses Xerox Finite-State Morphology tools [11]. However, this approached was criticized by Ahmed [8] for requiring excessive manual processing to state rules in an FST and for the ability only to analyze words that appear Arabic dictionaries. Kiraz summarized many variations of the FST approach in his paper "*Arabic Computation Morphology in the West*" [12]. For more information on two-level morphology and PC-Kimmo refer to the PC-KIMMO user's guide [20].

2. Use of rules in conjunction with statistics. This approach employs a list of prefixes, a list of suffixes, and templates to transform from a stem to a root. Possible prefix-suffix-template combinations are constructed for a word to derive the possible roots. RDI's system called MORPHO3 utilizes such this model [8]. Further, the Temple Project has produced a list of possible prefixes, suffixes, and verb/noun classes that would aid in the production of such a system [13]. Although such systems achieve broader morphological coverage of the Arabic language [8], manual derivation of rules is laborious, time-consuming and requires a good knowledge of Arabic orthographic (spelling rules) and morphotactic (rules governing which morphemes may each other – morphemes are meaning bearing units in a language [17]) rules. In fact, MORPHO3 was built in 3 man/years [8].

Another approach that was proposed for performing morphology in general is the use of unsupervised learning techniques. Goldsmith proposed an unsupervised learning automatic morphology tool called AutoMorphology [14]. This system is advantageous because it learns prefixes, suffixes, and patterns from a corpus or word-list in the target language without any need for human intervention. However, such a system would not be effective in Arabic morphology, because it does not address the issues of infixation, and would not detect rare prefixes and suffixes.

Large-scale morphological analyzers provide more information than just the root of word. They may provide information such as the meaning prefixes and suffixes and root disambiguation [8] [10] [11]. However, this paper is concerned with morphological analysis for the purpose of IR. Arabic IR is enhanced when the roots are used in indexing and searching [3] [4] [5].

3. System Description:

The system proposed by this paper is similar to the rule/statistical approach used by RDI's MORPHO3 [8]. However, this system does not require that rules and prefix and suffix lists be constructed manually. Instead, the system replaces the manual processing with automatic processing.

First, a system module called *build-model* utilizes a list of Arabic word-root pairs (1) to derive a list of prefixes and suffixes, (2) to construct stem templates, and (3) to compute the likelihood that a prefix, a suffix, or a template would appear. Second, another system module called *detect-root* accepts Arabic words as input, attempts to construct possible prefix-suffix-temple combinations, and outputs possible roots.

3.1 Where to Get a list of Word-Root Pairs:

The list of word-root pairs maybe constructed either manually, using a dictionary, or by using a pre-existing morphological analyzer such as ALPNET or MORPHO3 [8] [10].

1. Manual construction of word-root pair list: Building the list of several thousand pairs manually is time consuming, but feasible. Assuming that a person who knows Arabic can generate can process a word every 5 seconds, the manual process would require about 14 hours of work.
2. Automatic construction of a list using dictionary parsing: Extracting word-root pairs from an electronic dictionary is a feasible process. Since Arabic words are looked up in a dictionary using their root form, an electronic dictionary such as Lisan Al-Arab maybe parsed to generate the desired list. However, some care should be given to throw away dictionary examples and words unrelated to the root.
3. Automatic construction using a pre-existing morphological analyzer: This process is simple, but requires the availability of an analyzer.

For the purposes of this paper, method 3 was used to construct the list. Two lists of Arabic words were fed to ALPNET and then the output was parsed to generate the word-root pairs. One list was extracted from a corpus of traditional Arabic books (Ibn Al-Qayim Encyclopedia) owned by Al-Areeb Electronic Publishers [15]. The list contains 9,600 words that ALPNET was able to analyze successfully. The original list was larger, but the words that ALPNET was unable to analyze were excluded. The other was extracted from the LDC Arabic corpus (LDC2001T55) containing AFP news-wire stories [16]. This list contains 560,000 words. Of the 560,000 words, ALPNET was able to analyze 270,000 words successfully. The rest of the words (about 290,000) were also used for evaluating the new system.

3.2 “Build-Model” module:

As stated above, this module takes a word-root pair as input. By comparing the word to the root, the system determines the prefix, suffix, and stem template. For example, given the pair (wktAbhm, ktb), the system generates “w” as prefix, “hm” as suffix, and “XXAX” as stem template (X’s represent the letters in the root). The system looks-up the prefix “w” in the list of prefixes. If “w” is not found, “w” is added to the list of prefixes with a number of occurrences equal to one. If “w” is found, then the number of occurrences of the prefix is incremented by one. Suffixes and stem templates have similar lists and are accounted for in the same manner. The system takes into account the cases where there are no prefixes or suffixes and denotes either of them with the symbol “#”.

Word	w	k	t	A	b	h	m
Root		k	t		b		
Parts	Prefix	Stem (template XXAX)				Suffix	

Figure 5: The decomposition of the word “wktAbhm” with root “ktAb”

After that, the lists of prefixes, suffixes, and templates are read through to assign probabilities to items on the lists by dividing the occurrence of each item on each list by the total number of words. Another potential way of calculating the probabilities of prefixes and suffixes is that for each item (a prefix or suffix) the probability would be the chance that the item appears in the word and was actually a prefix or suffix. For example, if “w” appeared as the first letter in the word 100 times, 70 times of which it was actually a prefix, then the probability would be .70.

Notice that what the module is treating as a stem is slightly different than the standard definition of a stem given earlier. Standard stem templates may have letters added in the middle and in the beginning. For example the template “mXXwX” has “m” placed before the root and “w” placed in the middle. Both “m” and “w” are a part of the stem template. However, the module has no prior knowledge of standard stem templates. Therefore, for the template “mXXwX”, “m” is simply treated as a part of the prefix list and extracted template is “XXwX”.

3.3 “Detect-Root” module:

The detect-root module accepts an Arabic word and attempts to generate prefix-suffix-template combinations. The combinations are produced by progressively removing prefixes and suffixes and then trying matching all the produced stems to a template. For example, for the Arabic word “AymAn” the possible prefixes are:

“#”, “A”, and “Ay”

and the possible suffixes are:

“#”, “n”, and “An”.

The resulting feasible stems are:

Stem	Prefix	Template	Suffix	Root
AymAn	#	XyXAX	#	Amn
ymAn	A	XXAX	#	ymn
mAn	Ay	XXX	#	mAn
Aym	#	XXX	An	Aym
ymA	A	XXX	N	ymA

The ones that the system deemed as not feasible are “AymA” and “ym”. Although “AymA” is not feasible, “ym” is actually feasible, but the system did not know how to deal with it. The paper will address this problem in the next sub-section. The possible roots are ordered according to the product of the probability that a prefix would be observed, the probability that a suffix would be observed, and the probability that a template would be used. The probabilities of stems, suffixes, and templates are assumed to be independent. The independence assumption is made to simplify the ranking, but is not necessarily a correct assumption because prefix-suffix combinations are not allowed. Using the product requires some smoothing which will be discussed in the next sub-section.

3.4 Roots that the System Missed or Produced in Error:

As seen above, the system deemed the stem “ym” not feasible, while in actuality the stem maps to the root “ymm”. Other cases where the system failed were when the root had weak letters. Weak letters include “A”, “y”, and “w”. The weak letters are frequently substituted for each other in stem form or dropped all together. For example, the word “qAl” has the root “qwl” or “qyl” which would make the word mean ‘he said’ or ‘he napped’ respectively. Also, the word “f” has the root “wfy” where the letters “w” and “y” are missing. To compensate for these problems, two letter stems were corrected by introducing new stems that are generated by doubling the last letter (to produce “ymm” from “ym”), by adding a “w” before the stem or a “y” after the stem. As for stems with a week middle letter, new stems are introduced by substituting the middle letter with the other weak letters. For example, for the stem “qAl”, the system would introduce the stems “qwl” and “qyl”. This process over-generates potential roots. For example, from the three potential roots “qAl”, “qwl”, and “qyl”, “qyl” (to nap) and “qwl” (to say) are correct roots for the stem, but not “qAl”. To compensate for the over-generation, all the produced roots are checked against a list of dictionary roots that were extracted from *Lisan Al-Arab*. The list of roots extracted from *Lisan Al-Arab* contains about 10,000 roots.

As for smoothing the prefix and suffix probabilities, Witten-Bell discounting was used [17]. The smoothing is necessary because many prefixes and suffixes were produce in error, because the word-root pairs often had errors. Using this smoothing strategy, if a prefix or a suffix is observed once, then it will be removed from the respective list. As for the list of templates, it was reviewed by an Arabic speaker (the author of the paper) to insure the correctness of the templates. The Arabic examiner was aided by example words the system provided for each template. If a template was deemed not correct, it was removed from the list.

3.5 Particles:

Another shortcoming of the system is that it did not detect particles. To solve this problem, a list of Arabic particles was constructed with aid of *An-Nahw Ashamil* (an Arabic grammar book) [6]. If the system matched a potential stem to one of the words on the particle list, the system would indicate that the word is a particle. Note that particles are allowed to have suffixes and prefixes.

3.6 Letter Normalizations:

The system employs a letter normalization strategy in order to account for spelling variations and to ease in the deduction of roots from words. The first normalization deals with the letters “y” (ya - ي) and “Y” (alef maqsoura - ى). Both are normalized to “y”. The reason behind this normalization is that there is no one convention for spelling “y” or “Y” when either appears at the end of a word (Note that “Y” only appears at the end of a word). In the Othmani script of the Holy Qur’an for example, any “y” is written as “Y” when it appears at the end of a word [18]. As for traditional book publishing, the generally accepted convention is that “y” and “Y” are written as is. The second normalization is that of “'” (hamza - ء), “|” (alef maad – آ), “>” (alef with hamza on top - أ), “&” (hamza on w - ؤ), “<” (alef with hamza on the bottom - إ), and “}” (hamza on ya - ئ). The reason for this normalization is that all forms of hamza are represented in dictionaries as one in root form namely “'” or “>” depending on the dictionary and people often misspell different forms of alef. All are normalized to the symbol “A”.

4. Evaluation and Discussion:

To use ALPNET for evaluation, ALPNET was tested for correctness. 100 words that were analyzed by ALPNET were chosen at random for manual evaluation. All 100 words were analyzed correctly.

To evaluate the experimental system, two experiments were preformed. The first uses a large training set and the other uses a small training set.

4.1 Using a Large Training Set:

The list of 270K words was used for training the system and the list of 9,606 Arabic words was used for evaluation. Of the testing set, ALPNET analyzed all the words, while the experimental system analyzed 9,497 and failed on 112. For the generated roots, three different automatic evaluations were done:

First (Auto-Eval-1): The top generated root is compared to the roots generated by ALPNET. If the root is on the list, it is considered correct. Using this method, 8,206 roots were considered correct.

Second (Auto-Eval-2): The top two generated roots from the experimental system were compared to the list of roots that were generated by ALPNET. If either root appeared in the list then the morphological analysis was considered correct. Using this evaluation method, 8,861 roots were considered correct.

Third (Auto-Eval-*n*): All the generated roots are compared to the ones generated by ALPNET. If any match is found, the analysis is considered correct. Using this method, 9,136 roots were considered correct.

However, automatic evaluation has two flaws:

1. the number of Arabic roots in ALPNET's inventory are only 4,600 roots while the number of roots used by the system are more than 10,000. This could result in a correct root that ALPNET was unable to detect.
2. ALPNET often under-generates. For example the word "fy" could be the particle "fy" or could be a stem with the root "fyy". ALPNET only generates the particle "fy", but not the other root "fyy". This could lead to false negatives.

Therefore manual examination of reject roots was necessary. However, due to the large number of rejected roots, 100 rejected roots from the evaluation Auto-Eval-1 and Auto-Eval-2 were selected at random for examination to estimate the shortfall of the automatic evaluation. Of the 100 rejected roots:

Evaluation Method	Correct	Incorrect
Auto-Eval-1	46	54
Auto-Eval-2	38	62

Results summary:

Evaluation Method	Number of words	Words not analyzed	Number or roots deemed incorrect	Number of roots detected correctly	Number of roots estimated to be correct (manual evaluation)
Auto-Eval-1	9,609	112 (1.17%)	1,291 (13.6%)	8,206 (86.4%)	8,800 (92.7%)
Auto-Eval-2	9,609	112 (1.17%)	636 (6.7%)	8,861 (93.3%)	1,136 (96.1%)
Auto-Eval-*n*	9,609	112 (1.17%)	360 (3.8%)	9,136 (96.2%)	-

Also, the 292,216 words that ALPNET was unable to recognize were fed to the experimental system. The experimental system analyzed 128,169 words (43.9%). To verify the correctness of the system, 100 words were taken at random from the list for manual examination. Of the 100, 47 were actually analyzed correctly. Most of the failures were named-Entities. Extrapolating from the results of the manual examination, the experimental system would successfully recognize an estimated 60,000 words (20% of the original list).

Results summary:

Number of words	Number of roots detected	An estimate of the correctly detected roots
292,216	128,169 (43.9%)	60,000 (20%)

The failure of ALPNET and the low accuracy of the experimental system warrant further investigation. A quick review of the list shows a high frequency of named entities and obscure words.

4.2 Using a Small Training Set:

The 9,606 words list was used for training and the 270K words list was used for evaluation. The same automatic evaluation method mentioned above was used. Of the 270,468 words, the system was unable to analyze 84,421, and analyzed 186,047. Similar to experiment the experiment with the large training set, three automatic evaluations were used: Auto-Eval-1, Auto-Eval-2, and Auto-Eval-*n*. For Auto-Eval-1 and Auto-Eval-2, 100 of the rejected roots were manually examined to verify correctness. Of the 100 roots examined:

Evaluation Method	Correct	Incorrect
Auto-Eval-1	30	70
Auto-Eval-2	45	55

Results summary:

Evaluation Method	Number of roots	Words not analyzed	Number or roots deemed incorrect	Number of roots detected correctly	Number of roots estimated to be correct (manual evaluation)
Auto-Eval-1	270,468	84,421 (31.21%)	55,057 (29.6%)	130,990 (70.4%)	1 ,1 (79.3%)
Auto-Eval-2	270,468	84,421 (31.21%)	36,141 (19.4%)	149,906 (1 .6%)	1 ,1 (89.3%)
Auto-Eval-*n*	270,468	84,421 (31.21%)	27,020 (14.5%)	159,025 (85.5%)	-

Also, the 292,216 words that ALPNET was unable to recognize were fed to the experimental system. The experimental system analyzed 92,929 words (31.8%). To verify the correctness of the system, 100 words were taken at random from the list for manual examination. Of the 100, 55 were actually analyzed correctly. Extrapolating from the results of the manual examination, the experimental system would successfully recognize an estimated 60,000 words (20% of the original list).

Results summary:

Number of words	Number of roots detected	An estimate of the correctly detected roots
292,216	92,929 (31.8%)	1 ,000 (17%)

4.3 Success and Limitations:

The evaluation method clearly shows the effectiveness of the proposed morphological analyzer. The analyzer is often able to detect roots that were missed by a commercially available system. Also, due to the fact that rule are derived automatically, such a morphological analyzer maybe developed very rapidly. For the case of the analyzer described in the paper, it was written in less than 12 hours and with about 200 lines of Perl code [21]. Further, the analyzer is able to derive the roots of 40,000 words per minute on an Athlon 700 MHz machine with 256 MB of RAM running Linux.

Furthermore, the method used to develop this Arabic morphological analyzer can potentially be used to rapidly develop morphological analyzers for other languages. Some languages exhibit morphological properties similar to those of Arabic such as Hebrew [12].

However, the system is restricted in the following aspects:

1. Since it limits the choice of roots to a fixed set, it does stem words transliterated from other languages such as transliterated named entities. For example, the English word Britain is transliterated as “bryTAnyA”. From “bryTAnyA”, some the words that maybe generated are: “bryTAny” (British), “AlbryTAny” (the British), and “AlbryTAnyyn” (Englishmen).
2. Some words in Arabic are 1 letter long, but have 3 letter roots. For example, the word “q” means “protect yourself”. Since they are very rare, they may not appear in the training set.
3. Some individual words in Arabic constitute complete sentences. For example, the word “AnlzmkmwhA” means “will (A) we (n) forcefully bind (lzm) you (km) to it (hA)?” These also are rare and may not appear in a training set.
4. The analyzer lacks the ability to decipher which prefix-suffix combinations are legal. Although deciphering the legal combinations is feasible using statistics, the process would potentially require a huge number of examples to insure that the system would not disallow legal combinations.

5. Conclusion:

The paper presented a way to rapidly develop a shallow Arabic morphological analyzer. The analyzer is based on automatically derived rules and statistics. The analyzer is cross-platform and freely-distributable. Although some knowledge of the Arabic language was required to verify the correctness of derived rules for example, the amount of time required to build the rules is reduced to hours rather than days or weeks. Some the possible future work includes:

1. Integrating stemming with the analyzer to handle words the analyzer failed on.
2. Attempting to develop morphological analyzers for other language using the same method describe in the paper.
3. Collecting statistics on legal prefix-suffix combinations to further improve the analyzer.
4. Comparing the retrieval effectiveness when indexing is done using this analyzer compared to another commercially available analyzer such as ALPNET.
5. Examining the words for which ALPNET was unable to produce roots. This would give insight into the strength and weakness of ALPNET.

6. References:

1. RDI (Research and Development International), Cairo, Egypt. www.rdi-eg.com
2. Sakhr, Cairo, Egypt. www.sakhr.com
3. Al-Kharashi, Ibrahim and Martha Evens, "*Comparing Words, Stems, and Roots as Index Terms in an Arabic Information Retrieval*." JASIS. 45 (8): 548-560, 1994.
4. Abu-Salem, Hani, Mahmoud Al-Omari, and Martha Evens, "*Stemming Methodologies Over Individual Query Words for Arabic Information Retrieval*." JASIS. 50 (6): 524-529, 1999.
5. Hmeidi, Ismail, Ghassan Kanaan, and Martha Evens, "*Design and Implementation of Automatic Indexing for Information Retrieval with Arabic Documents*." JASIS. 48 (10): 867-881, 1997.
6. Abdul-Al-Aal, Abdul-Monem, *An-Nahw Ashamil*. Maktabat Annahda Al-Masriya, Cairo, Egypt, 1987.

   عبد المنعم عبد العال، النحو الشامل، مكتبة النهضة المصرية، القاهرة، جمهورية مصر العربية، 1987
7. Ibn Manzour, *Lisan Al-Arab*. www.muhaddith.org.
8. Ahmed, Mohamed Attia, "*A Large-Scale Computational Processor of the Arabic Morphology, and Applications*." A Master's Thesis, Faculty of Engineering, Cairo University, Cairo, Egypt, 2000.
9. Koskenniemi, Kimmo, "*Two Level Morphology: A General Computational Model for Word-form Recognition and Production.*" Publication No. 11, Dept. of General Linguistics, University of Helsinki, Helsinki, 1983.
10. Beesley, Kenneth, Tim Buckwalter, and Stuart Newton, "*Two-Level Finite-State Analysis of Arabic Morphology.*" Proceedings of the Seminar on Bilingual Computing in Arabic and English, Cambridge, England, 1989.
11. Beesley, Kenneth, "*Arabic Finite-State Morphological Analysis and Generation.*" COLING-96, 1996.
12. Kiraz, George, "*Arabic Computation Morphology in the West.*" Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing, Cambridge, 1998.
13. The Temple Project. http://crl.nmsu.edu/tide/
14. Goldsmith, John, "*Unsupervised Learning of the Morphology of a Natural Language*." http://humanities.uchicago.edu/faculty/goldsmith/, 2000.
15. Al-Areeb Electronic Publishers, LLC. 16013 Malcolm Dr., Laurel, MD 20707, USA

16. Cole, Andy, David Graff, and Kevin Walker, “*Arabic Newswire Part 1 Corpus (1-58563-190-6)*”, Linguistic Data Consortium (LDC). www.ldc.upenn.edu
17. Jurafsky, Daniel and James Martin, “*Speech and Language Processing*.” Prentice Hall, New Jersey, 2000.
18. The Holy Qur’an.
19. Antworth, Evan L, “*PC-KIMMO: a two-level processor for morphological analysis*.“ Occasional Publications in Academic Computing No. 16. Dallas, TX: Summer Institute of Linguistics, 1990.
20. Antworth, Evan L, “*PC-KIMMO User’s Guide.*” www.sil.org/pckimmo/v2/doc/guide.html, 1995
21. Perl. www.perl.com

Appendix:

Appendix figure I: Letter mappings

@	ا	alef	‘	ء	hamza	|	آ	alef maad
&	ؤ	hamza on w	>	□	alef with hamza on top	<	□	alef with hamza at bottom
}	□	hamza on y	b	□	ba	t	□	ta
p	□	ta marbouta	v	□	tha	j	□	jeem
H	□	H’a	x	□	Kha	d	□	dal
O	□	thal	r	□	raa	z	□	zain
S	□	seen	P	□	sheen	S	□	Saad
D	□	daad	T	□	T’a	Z	□	Zha
E	□	ain	j	□	jhan	f	□	fa
Q	□	qhaf	k	□	kaf	l	□	lam
M	□	meem	n	□	noon	h	□	ha
W	□	waw	y	□	ya	A	Any hamza or Alef	

List of Arabic Particles:

□ ASMA' AL-ISHARA

HOA	hOh	hOAn	hAtyn	hAlAA
AllAty	AllAAy	AllwAty	tlk	

□ DAMA'IR (Pronouns)

AnA	nHn	Ant	AntmA	Antm

Antn	hw	hy	hmA	hm
Hn				

#ASMA ASHART

MA	mn	AynmA	mty	Ayn
AyAn	lmA	AOA	klmA	mhmA
AO	Hyv	HyvmA	Any	kyfmA

#ASMA AL ISTIFHAM

Kyf	hl	mn	fym	mA
Ayn	mty	Any	km	AyAn
Bm	lm	mm	lmAOA	mAOA
AlA	Em	ElAm	AlAm	

#HOROUF Al-JARR

Emn	iiiiu	Aly	fy	mn
En	ky	w	mO	mnO
Hty	xlA	EdA	HAPA	

#AL-ATF

W	vm	Aw	Am	bl
Lkn	lA	Hty		

# NID'A

YA	AyA	hyA	Ay	

#NAFY

Ln	lm	lmA	lA	mA
An				

#TAWKEED

An	qd			

#NAHY

lA

#AL-SHART

An	AO	lw	lwlA	AmA
LmA				

#TANBEEH

HA	yA	AlA	hlA	

#MASDARIYA

An	mA	lw	ky	

#An and its siblings

An	kAn	lkn	lyt	lEl
LA	Esy			

# miscellaneous

Aly	byn	tHt	Ely	Ah
Al	Am	An	Ah	Aw
OA	AlA	ty	fy	qd
Lqd	lA	mA	mE	hl