The Challenges and Pitfalls of Arabic Romanization and Arabization

Jack Halpern (春遍雀來)
The CJK Dictionary Institute, Inc. (日中韓辭典研究所)
34-14, 2-chome, Tohoku, Niiza-shi, Saitama 352-0001, Japan
jack@cjk.org

Abstract

The high level of ambiguity of the Arabic script poses special challenges to developers of NLP tools in areas such as morphological analysis, named entity extraction and machine translation. These difficulties are exacerbated by the lack of comprehensive lexical resources, such as proper noun databases, and the multiplicity of ambiguous transcription schemes. This paper focuses on some of the linguistic issues encountered in two subdisciplines that play an increasingly important role in Arabic information processing: the *romanization* of Arabic names and the *arabization* of non-Arabic names. The basic premise is that linguistic knowledge in the form of linguistic rules is essential for achieving high accuracy.

1 Introduction

The process of automatically transcribing Arabic to a Roman script representation, called **romanization**, is a tough computational task to which there is no definitive solution. The opposite operation of transcribing a non-Arabic script into Arabic, called **arabization**, is also difficult but for different reasons.

This paper briefly describes the algorithms and major linguistic issues encountered in the course of developing two automatic transcription systems: (1) Automatic Romanizer of Arabic Names (ARAN), which romanizes unvocalized Arabic names into various romanizations systems, and (2) Non-Arabic Name Arabizer (NANA), which arabizes non-Arabic names written in the Roman and CJK scripts.

A novel feature of these systems is that they are fine tuned to transcribing personal names and placenames to and from Arabic, with special focus on the linguistic knowledge and rules required for transcribing CJK names written in their native script directly into Arabic, something probably never attempted. These systems are part of our ongoing efforts to develop Arabic resources for automatic transcription, machine translation and named entity extraction.

The following typographic conventions are used in this paper:

1. Phonemic transcriptions are indicated by slashes (قــابوس > /qaabuus/) .
2. Phonetic transcriptions are indicated by square brackets (قــابوس > [qɑːbuːs]) .
3. Graphemic transliterations are indicated by back slashes (قــابوس > \qAbws\) .
4. Popular transcriptions are indicated by italics (قــابوس > *Qaboos*).

2 Motivation and Previous Work

Arabic transcription technology is playing an increasingly important role in a variety of practical applications such as named entity recognition, machine translation, cross-language information retrieval and various security applications such as anti-money laundering and terrorist watch lists. Despite the importance of these applications, Arabic transcription has not been the subject of sufficient studies that examine the linguistic issues. This paper attempts to fill that gap.

Several companies and researchers have developed automatic diacriticization software. Vergyri and Kirchhoff (2004) report the high error rate of these products. Gal used a HMM bigram model and achieved a 14% error rate, while AbdulJaleel and Larkey (2003) developed an n-gram based statistical system for arabizing English, with an error rate of 10%-20%. Elshafei et al. (2006) report a 5.5% error rate using an HMM approach, while Arbabi et al (1994). developed a diacriticizer that combines a knowledge base with neural networks to achieve a low error rate of 3.1% but which rejects 55% of the names as unprocessable.

We have not used sophisticated statistical approaches. Our basic strategy has been to use conventional linguistic knowledge because we believe that ultimately statistical methods by themselves are inadequate. Kay (2004) argues that "statistics are a surrogate for knowledge of the world" and that "this is an alarming trend that computational linguists ... should resist with great determination." This was reinforced by Farghaly (2004) when he wrote "It is becoming increasingly evident that statistical and corpus-based approaches...are not sufficient..."

Our policy is that linguistic rules, based on deep analysis of the source and target scripts, are indispensable. To rephrase, many contemporary statistical methods involve brute-force mathematical techniques that exploit vast amounts of data, whereas a rule-based approach captures aspects of human intelligence because it is based on linguistic knowledge. We have combined linguistic rules with statistically derived mapping tables to build a flexible system that can be extended to other Arabic script based languages.

3 Basic Concepts

Much confusion surrounds the terms *transliteration* and *transcription*, with the former often misleadingly used in the sense of the latter even in academic papers (AbdulJaleel and Larkey, 2003). To discuss these concepts in an unambiguous manner it is necessary to understand these and related terms correctly.

Romanization is the representation of a language written in a non-Roman script using the Roman alphabet. This includes both transliteration and transcription, e.g. محمد is transliterated as \mHmd\ and transcribed as *Mohammed, Muhammad*, or *Mohamad*, among many others.

Transliteration is a representation of the script of a source language by using the characters of another script. Ideally, it unambiguously represents the graphemes, rather than the phonemes, of the source language. For example, محمد is transliterated as \mHmd\, in which each Arabic letter is unambiguously represented by one Roman letter, enabling round-trip conversion.

Transcription is a representation of the source script of a language in the target script in a manner that reflects the pronunciation of the original, often ignoring graphemic correspondence. This includes the following subcategories:

1. A *phonetic transcription* represents the actual speech sounds, including allophones. The best known of these is IPA. For example, محمد is transcribed as [muħɛ̈mmɛ̈d].
2. A *phonemic transcription* represents the phonemes of the source language (ignoring allophones), ideally on a one-to-one basis. For example, محمد is transcribed as /muHammad/, in which a represents the phoneme /a/, rather than the phone [ɛ̈].
3. A *popular transcription* is a conventionalized orthography that roughly represents pronunciation. For example, محمد is transcribed in some 200 different ways, such as *Mohammed, Muhammad, Moohammad, Moohamad, Mohammad, Mohamad,* etc.

Diacriticization is the process of adding vowel signs (called vocalization) and other diacritics. For example, محمد \mHmd\ is converted to the vocalized مُحَمَّد \muHam~ad\. Note the four diacritics that were added.

Arabization is the reverse of romanization; that is, the representation of a non-Arabic script, such as the Roman and CJK scripts, using the Arabic alphabet, e.g., *Muhammad* → محمد, *Clinton* → كلينتـــــون 埼玉, *Saitama* → ســـــايتاما.

4 Why is Arabic ambiguous?

A distinguishing feature of abjads in general, and of Arabic in particular, is that words are written as a string of consonants with little or no indication of vowels, referred to as *unvocalized Arabic.* Though diacritics can be used to indicate short vowels, they are used sparingly, while the use of consonants to indicate long vowels is ambiguous. On the whole, unvocalized Arabic is highly ambiguous and poses major challenges to Arabic information processing applications.

4.1 Morphological Ambiguity

Arabic is a highly inflected language. Inflection is indicated by changing the vowel patterns as well as by adding various suffixes, prefixes, and clitics. A full paradigm for كَاتِب /kaatib/ 'writer' that we created (for a comprehensive Arabic-English dictionary project) reaches a staggering total of 3487 valid forms, including affixes and clitics as well as inflectional syncretisms. For example, كاتـب can represent any of the following seven wordforms: كَاتِبْ /kaatib/, كَاتَبَ /kaataba/, كَاتِبٍ /kaatibin/, كَاتِبٌ /kaatibun/,كَاتِبَ /kaatiba/, كَاتِبِ /kaatibi/, كَاتِبُ /kaatibu/.

4.2 Orthographical Ambiguity

On the orthographic level, Arabic is also highly ambiguous. For example, the string مو can theoretically represent 40 consonant-vowel permutations, such as *mawa, mawwa, mawi, mawwi, mawu, mawwu, maw, maww, miwa, miwwa....* etc., though in practice some may never be used. Humans can normally disambiguate this by context, but for a program the task is formidable.

Conventional wisdom has it that the Arabic script is ambiguous "due to non-representation of short vowels," while other features are often lightly passed over. In fact, a whole gamut of factors contribute to orthographical ambiguity.

The list of factors below is not intended to serve as a detailed treatment of Arabic orthographic ambiguity, but to demonstrate the principal linguistic issues that need to be addressed to achieve accurate transcription.

1. The greatest challenge is the omission of short vowels; e.g., the unvocalized كاتـب \kAtb\ can represent seven wordforms such as كَاتِبْ /kaatib/ and كَاتِبَ /kaatiba/.
2. In contrast, some short vowels actually *are* represented. For example, *taa' marbuuTa* often indicates a short /a/, as in جامعة /jaami`a/, while in foreign names short and long vowels are normally written identically by adding ا ,ي or و, as in روسـيا \rwsyA\ 'Russia'.
3. Long /aa/ can be expressed in multiple ways, e.g., by *'alif Tawiila* (ا) as in سـوريا, by (2) *'alif maduuda* (آ) as in آسـيا, and by (3) *'alif maqSuura* (ى) as in آسـيا الوسـطى.
4. Long vowels are sometimes omitted too, as in هذا/haadha/. In this case, the *'alif qaSiira* ("dagger alif") is omitted.
5. Not all bare alifs represent long /a/. Some are silent (next item), while some are nunated; e.g., را in شـكرا represents /ran/, رًا, *not* رَا /raa/.
6. *'alif alfaaSila* (otiose alif), added to the third person masculine plural forms of the past tense, is a mere orthographic convention and is not pronounced. Though it must be transliterated, it must not be transcribed, e.g., كتبـوا is transliterated as \ktbwA\, with *'alif* at the end, but transcribed as /katabuu/, omitting the *'alif*.
7. The diacritic *shadda* indicating consonant gemination is normally omitted, e.g., the unvocalized محمد *Muhammad* (vocalized مُحَمَّد) provides no clues that the [m] should be doubled.
8. Another source of ambiguity is the omission of *tanwiin* diacritics for case endings, e.g., in شـكرا \$ukrAF\ (vocalized شُكْرًا), the *fatHatayn* is not written.
9. The rules for determining the hamza seat are of notorious complexity. In transcribing to Arabic, it is difficult to determine the hamza seat as well as the short vowel that follows; e.g., hamzated *waaw* (ؤ) could represent /'a/, /'u/ or even /'/ (no vowel).
10. In arabization, determining the hamza seat requires the application of complex rules based on the phonological environment, which is further complicated by the frequent omission and inconsistent use of hamza in foreign names (see Section 7).
11. Phonological alternation processes such as assimilation that modify the phonetic realization. For example, the unvocalized الرجل الطويـل 'the tall man' is realized as /'arrajulu-TTawiilu/ (اَلرَّجُلُ ٱلطَّوِيل), in which the ال is assimilated into طّ /TTa/, not as /'alrajulu alTawiilu/.
12. Vowel shortening is sometimes lexically determined and thus cannot be predicted from the orthography; e.g., فـي القاهرة 'in Cairo' is pronounced /fi-lqaahira/, not /fii-lqaahira. That is, /fii/ is shortened to /fi/.

5 Automatic Romanizer of Arabic Names

5.1 Overview

The Automatic Romanizer of Arabic Names (ARAN) consists of multiple modules for the transcription and transliteration of Arabic and related tasks such as variant generation and vocalization. The core problem that ARAN addresses is making an intelligent guess at determining the vowels of unvocalized Arabic names and generating romanized candidates based on statistically motivated linguistic rules derived from an in-depth analysis of Arabic orthography. The principal components of ARAN are:

1. ATAN: Automatic Transcriber of Arabic Names
2. AXAN: Automatic Transliterator of Arabic Names

3. APAN: Automatic Phoneticizer of Arabic Names
4. ADAN: Automatic Diacriticizer of Arabic Names
5. AVAN: Automatic Variant Generator for Arabic Names

Table 1 shows examples of how each module processes a string of unvocalized Arabic:

Table 1. Output from Various ARAN modules

Unvocalized	**Vocalized**	**Phonemic**	**Graphemic**	**Phonetic**	**Popular**
(input)	(ADAN)	(ATAN)	(AXAN)	(APAN)	(AVAN)*
محمد	مُحَمَّد	muHammad	mHmd	muħɛ̈mmɛ̈d	Muhammad
قـــابوس	قَابُوس	qaabuus	qAbws	qɑːbuːs	Qaboos
جمال	جَمَال	jamaal	jmAl	dʒɛ̈mɛ̈ːl	Jamal
مكة	مَـكّة	makka	mkp	mɛ̈kkɛ	Mecca

*Only one popular variant is shown, but in reality there could be dozens. For example, for قـــابوس AVAN generates *Qabuus, Qabus, Qabous, Qabooss,* … and many more.

5.2 Romanization Algorithm

The romanization algorithm accepts an Arabic string as input and generates a list of romanized candidates by combining lookup in the Database of Arabic Names (DAN), a database of about 180,000 romanized Arabic name variants and their variants, with a knowledge base of rules. ARAN can generate candidates in pure algorithmic mode, or it can access DAN to find explicit entries before resorting to algorithmic generation. Roughly, the algorithm works as follows:

1. Get an Arabic string from the input file.
2. Transliterate to Buckwalter for internal processing using the AXAN module.
3. Attempt to find an exact match in DAN.
4. If that fails, perform a fuzzy match to retrieve from DAN.
5. If that fails, generate romanization candidates algorithmically.
6. Output a list of romanized candidates.

For example, إبـــراهيم is first transliterated to \<brAhym\ and looked up in DAN. If the parameters are set to return popular readings and their variants, the output will be *Ibrahiim, Ibrahim, Ebraheem, Ebrahiim....* If the parameters are set to return purely generated candidates the result will be *ibraahiim, ibaraahiim, ibiraahiim, iburaahiim*, one of which is correct. These candidates can be further expanded by AVAN to generate variants such as *Ibrahim* and *Ibraahim*.

Fuzzy matching, such as ignoring hamza and collapsing *'alif* with *'alif maqSuura,* is a bit risky but might improve the match rate because fuzzily matched names could often be correct, whereas generated names could have incorrect short vowels. The user can set parameters to output any desired combination of three modes: exact match, fuzzy match or algorithmic generation.

5.3 Rules Knowledge Base

ARAN uses a knowledge base module for generating romanized strings from the Arabic input string. This is the central component of the algorithm but is independent of it for maximum flexibility. The rules can be modified by the user to further refine the accuracy or to adjust them to other Arabic-script based languages.

The knowledge base was created by in-depth analysis of the Arabic orthography using the results of statistical analysis of a large name corpus based on a bilingually aligned phone directory. A regular-expression-like mini-language for writing vocalization and romanization rules was developed in which LHS (left-hand side) and RHS (right-hand side) style rules are defined as declarative statements on a high level of abstraction independent of specific computer languages. These are then implemented by the appropriate functions in the romanization algorithm module. For example, the rule "I:C1(?=[^Awyp]):&c[aiu]" (colons are field separators) means as follows:

> *an initial consonant (indicated by "I"1 in the first field) in the C1 consonant subset not followed by a long vowel 'alif, waaw, yaa' or taa' marbuuTa (regex back reference), is converted to the corresponding consonant in question (defined in a mapping table) followed by one of the romanized short vowels 'a', 'i' or 'u'.*

6 Non-Arabic Name Arabizer

6.1 Overview

The **Non-Arabic Name Arabizer** (NANA) is designed to arabize non-Arabic names. This includes Roman-script names such as *Bill Clinton* to بيـــل كلينتـــون, as well as a technology probably never attempted before: transcribing CJK names directly into Arabic. We have developed language-dependent rules, mapping tables and algorithms for transcribing CJK names written in their native scripts. For example, the Japanese placename 埼玉 /saitama/ is arabized as ســـــايتاما, the Chinese name 杨海洋 /yang haiyang/ as يـــانغ هاييـــانغ, and the Korean city 부산 /busan/ as بوســـان.

Various papers, such as AbdulJaleel and Larkey (2003), describe systems for transcribing Roman-script names into Arabic. Although NANA also has this capability, it is beyond the scope of this paper. The issues for Chinese and Korean, the subject of a future paper, are similar in nature but require a different set of language-specific rules.

6.2 Arabization Policy

A fundamental problem in arabizing CJK names is that there are significant differences between the Arabic and CJK phonological systems and the lack of detalied transcription standards. Since these languages are not well known in the Arab-speaking world, CJK names are often arabized on the basis of their romanized transcriptions, rather than the native script, and it is sometimes erroneously assumed that the Roman letters are pronounced as in English. This is further complicated by the plethora of CJK romanization standards. We have established an arabization policy for Japanese based on a number of sometimes conflicting criteria:

1. How names are actually spelled on the Arabic web, atlases, maps and books.
2. Ensuring that same source syllables are spelled consistently taking into account phonological changes.
3. Treating Japanese names as a sequence of syllables, rather than of morae, since that is how they are commonly transcribed.
4. Using hamza to represent vowel sequences only in those cases where dipthongization is not possible or awkward (see Section 6.3).
5. Generating hamzated variants, such as ســـــائيتاما for the more common ســـــايتاما (埼玉 /saitama/), and other kinds of variants, such as كاناجـــاوا for the more common كاناغـــاوا (神奈川 /kanagawa/).

6.3 Vowel Sequence Ambiguity

Vowels sequences are difficult to transcribe because they could represent diphthongs, monophthongs (distinct vowels), or long vowels. Representing Japanese vowels accurately in Arabic is not possible. In cases where vowel sequences represent monophthongs, hamza is sometimes used and sometimes omitted.

Table 2. Diphthong Ambiguity for 福井 /fu-ku-i/

No.	Arabic	Google hits	Buckwalter
1	فوكوئـــــي	468	fwkw}y
2	فوكـــوئ	9	fwkw}
3	فوكـــوي	1950	Fwkwy
4	فوكويـــــي	335	Fwkwyy

Table 2 shows some of the variation to expect in Japanese name Arabization. Though phonologically (2) is the most accurate, it is the least used. As expected, the diphthongized (3) is the most common form because of the tendency to avoid hamza in foreign names. Some important vowel sequence issues are:

1. There is a strong tendency not to use non-initial hamza, as in (1) and (2) above, in foreign names. One reason for this is insufficient knowledge of the phonology of the source language, especially of such "exotic" languages as Japanese.
2. Japanese is especially problematic because it is moraic. Some Japanese mora sequences, such as あい /ai/ or うい /ui/, are often diphthongized in Arabic, though ideally the second vowel should be treated as a monophthong represented by hamza. That is, 福井 /fu-ku-i/ should be written as (1) فوكوئـــــي or (2) فوكـــوئ, rather than the more common (3) فوكـــوي.
3. In theory, a vowel sequence like /ai/ as in さい /sa-i/ can be written in five ways: ساي ســائي ســايي سائ سي. To accurately transcribe a name like Saitama (埼玉) it is necessary to know that it consists of four morae (/sa-i-ta-ma/ さいたま), rather than three syllables (/sai-ta-ma/). Ideally it should be transcribed as ســـــائيتاما, rather than the much more common ســـــايتاما. That is, since /sa-i/ is a bimoraic syllable, the hamza

over *yaa'* should be used to represent /i/ as a distinct monophthong, as in سائي. In reality, Saitama is normally spelled ســـــايتاما, so that /sa-i/ is diphthongized as ساي /say/.

4. In names like 福岡 /fu-ku-o-ka/ the sequence /ku-o/ represents distinct sounds that cannot be diphthongized. Following hamza rules, this should be written فوكوؤوكـــا, but in fact it is commonly spelled فوكوأوكـــا, in which أو, rather than ؤو, represents /u/. Omitting the hamza here would make little sense.

6.4 Long and Short Vowels

The treatment of Japanese vowels is complex and may have hamzated variants.

Table 3. Long and Short Vowels

No.	Kanji	Kana	Phonemic	Arab1	Arab2	Arab3
1	太田	おおた	oota	أوتـا		
2	風馬	ふうま	fuuma	فومـا		
3	敬子	けいこ	keiko	كييكــــو	كيكـــو	
4	空野	くうの	kuuno	كونـــو		
5	久野	くの	kuno	كونـــو		
6	日枝	ひえだ	hieda	هييـــدا	هيئـــدا	هيئيـــدا
7	芳江	よしえ	yoshie	يوشـــــيي	يوشـــــيئه	يوشـــــيئي

1. Japanese long vowels are expressed in various ways, such as by repeating the vowel as in (2) ふう /fuu/, or by adding う /u/ after /o/ as in (1). えい /ei/ is special because the ي may be repeated, as in (3).
2. Since short vowels are omitted in Arabic, short vowels in foreign names are normally transcribed as if they were long; that is, by adding ا for /a/, ي for /i/ and و for /u/. Thus both (4) and (5) are written identically as كونـــو and there is no way to distinguish vowel length.
3. Normally the vowel /e/ is not distinguished from /i/ and both are represented by ي. An extra complication is that at word end /e/ is sometimes expressed by ه, so that in transcribing such names as (5) and (6) it is necessary to consider hamza rules, whether to diphthongize, the position of the syllable in the word, and how these interact.

6.5 Arabization Algorithm

The arabization algorithm accepts a CJK string as input and generates a list of romanized candidates by combining lookup in the Japanese-English Proper Noun Database (JEP), a database of about 600,000 Japanese personal and place names using a knowledge base of rules and mapping tables fine tuned to the Japanese and Arabic phonological systems. Roughly, the algorithm works as follows:

1. Get a string from the input file.
2. Determine if the string is Japanese.
3. Convert kanji to hiragana reading by looking up in JEP.
4. Convert hiragana to romanized Japanese by looking up in JEP.
5. If (3) fails, convert to hiragana algorithmically (difficult due to extreme ambiguity).
6. If (3) returns multiple strings, use criteria like frequency and semantic codes to eliminate unlikely candidates.
7. Determine whether to diphthongize or to use hamza by considering both the hiragana and the romanized Japanese.
8. Use the rules knowledge base, which is embedded in a multi-option comprehensive hiragana-to-Arabic mapping tables to convert to Arabic script.
9. The AVAN module generates variants if requested by user parameters.
10. Output arabized name (with or without variants as necessary).

We have not yet performed formal error rate testing, but our preliminary experiments indicate that the above algorithm can arabize a CJK name to its correct or legitimate variant form with a success rate of nearly 100%. This is because the algorithm is based on a thorough understanding of the Arabic and Japanese (as well as Chinese and Korean, though not discussed here) phonological systems, and a comprehensive mapping table designed to cover almost all possible Japanese-to-Arabic mappings, including positional variants and phonological changes resulting from liaison.

7 Arabic Orthographic Variants

The number of personal names and their variants in the world is in the billions. Identifying names and their variants (named entity recognition) is a hot topic in computational linguistics. To enhance this technology, we added a variant generation module (AVAN) to both the ARAN and NANA systems, which is supported by comprehensive databases of CJK proper nouns.

7.1 Romanization Variants

The many popular transcriptions of Arabic names result in a large number of variants. One reason for this is that several Arabic consonants, such as ع [ʔˤ], ض [dˤ], ط [tˤ] and ظ [ðˤ], do not exist in European languages. These sounds are difficult to pronounce and are rendered in different ways when romanized. Another factor is the bewildering variety of ways in which Arabic vowels are transcribed, partially due to dialectical influences. For example, the أ /'u/ in أسامة is transcribed in various ways as seen in *Usama, Ousama, Osama* and *Oosama*, while معمر \mEmr\ is spelled as *Moammar, Muammar, Mu'ammar, Mo'ammar, Moammar, Moamer, Moamar*, and others.

7.2 Arabic Variants

Both Arab and foreign names have orthographic variants in Arabic. These are of two kinds:

1. Orthographic variants are non-standard ways to spell a specific variant of a name, like ابـو ظـبي instead of أبـو ظـبي for Abu Dhabi, in which the hamza is omitted.
2. Orthographic errors are frequently occurring, systematic spelling mistakes, like yaa' in ابـو ظـبي (Abu Dhabi) being replaced by *'alif maqSuura* in ابـو ظـبى.

these cannot be rigorously defined, they are both of frequent occurrence based on statistical and linguistic analysis of MSA orthography. It should also be noted that "standard form," though linguistically correct, is not necessarily the most common form (we are gathering statistics for the occurrence of each form).

There are often many more variants than those shown above. For example, Alexandria can be written in about a dozen ways, the most frequent ones according to Google being الاســـــكندرية with 2,930,000, الإســـــكندرية with 690,000, and الاســـــكندريه with 89,200 occurrences respectively.

8 System Modules and Future Work

The principal components of ARAN (some of which are in progress) are briefly described below,

1. The **Automatic Transcriber of Arabic Names** (ATAN) is ARAN's core module for generating phonemic and popular transcriptions of Arabic personal names. Because of the inconsistent nature of the various popular Arabic romanization systems, there are often many, sometimes dozens or even hundreds, of romanizations for the same name.
   ATAN supports most of the commonly used systems, and has a flexible architecture that enables the user to configure the system to support user-defined systems. For example, شـــولوخ, which is first transliterated to \$wlwx\ by the AXAN module, can then be transcribed as /shwlwkh/ in the ALC-LC system, as /šūlūḫ/ in the DIN system, as *Shoulokh* as a possible English spelling, etc. The AVAN module can then be used to return many popular variants.

Table 4. Orthographic Variation in Arabic Names

Standard	Buckwalter	English	Variant	Error	Remarks
أبـو ظـبي	>bw Zby	Abu Dhabi	ابـو ظـبي	أبـو ظـبى ابـو ظـبى	V: omit hamza E: *'alif maqsura* replaces *yaa'*
الإســـــكندرية	Al<skndryp	Alexandria	الاســـــكندرية	إســـــكندريهال	V: omit *hamza* E: *haa'* replaces *taa' marbuuTa*
بـــالو ألتـــو	bAlw >ltw	Palo Alto	بـــالو التـــو بـــالو آلتـــو		V1: omit hamza V2: *madda* replaces hamza
طوكيـــو	Twkyw	Tokyo		توكيـــو	E: *taa'* replaces *Taa'*

Table 4 shows examples of variants ("V") and errors ("E"). Though the difference between

2. The **Automatic Transliterator of Arabic Names** (AXAN) generates transliterations of

Arabic names or any other Arabic text. There are few strict transliteration systems that use unique symbols for each letter and allow for round-trip conversion. The excellent and widely used Buckwalter transliteration system is not only supported by AXAN, but is also used for internal processing in all ARAN databases and algorithms. AXAN can be configured to support other transliteration systems, including *Cyrillization*, by adding custom mapping tables .

3. The **Automatic Phoneticizer of Arabic Names** (APAN) generates phonetic transcriptions of Arabic names in IPA. This represents the actual pronunciation in MSA, including distinctions between the major allophones. For example, the name قـــابوس *Qaboos* is transcribed as [qɑːbuːs]. Note that the *phonemic* transcription /qaabuus/ only indicates the vowel length (/aa/), whereas the *phonetic* transcription also indicates the quality of the vowel (ɑː), distinguishing it from its more common realization [ɛ̈ː]. APAN can be configured to transcribe in various MSA flavors. This refers to regional variations in MSA pronunciation, *not* to Arabic dialects per se. For example, for جمال /jammal/ APAN generates [dʒɛ̈mɛ̈ːl] for Gulf MSA, [gɛ̈mɛːl] for Egyptian MSA , and [ʒɛmɛ̈ːl] for Levantine MSA.
4. The **Automatic Generator of Variants for Arabic Names** (AVAN) supports the ARAN and NANA system by generating a large number of variants and variant candidates both algorithmically and by retrieving from hardcoded databases, whose occurrences are then validated in Arabic corpora and the web. See Section 7 for details.
5. The **Automatic Diacriticizer of Arabic Names** (ADAN) automatically *diacriticizes*, or adds vowels and other diacritics (like *fatha* and *shadda*) to unvocalized or semi-vocalized Arabic. For example, محمد \mHmd\ and الرياض \AlryAD\ are converted to the vocalized مُحَمَّد and الرِّيَاض respectively. This is related to, but distinct from, the equally difficult task of phonemic transcription. See Table 1 for examples.
6. There are dozens of non-Arabic languages that are or have been written in the Arabic script, referred to as Arabic Script Based Languages (ASBL). The most important of these are Farsi (official language of Iran), Pashto (western Pakistan and official language of Afghanistan), Dari (Afghan dialect of Farsi, official language of Afghanistan), Urdu (official language of Pakistan) and Kurdish (Turkey, Iraq, Iran, Syria, Armenia, Lebanon). Others include Shamukhi (Pakistani version of Punjabi), Kashmiri (India and Pakistan), and Uyghur (northwest China). ARAN will eventually be expanded to (1) romanize to/from the major ASBL languages, (2) automatically identify the language, (3) automatically detect legacy encodings and convert to Unicode.

9 Conclusion

As we have seen, the high level of ambiguity in the Arabic script makes it challenging to build automatic transcription systems that produce reliable results. In particular, we have seen the difficulties in arabizing CJK names due to the lack of standards and to the major phonological differences between the languages. We have also seen how important linguistic knowledge is in such areas as Japanese-to-Arabic transcription, resulting in a very high accuracy rate. Since Arabic transcription is playing an increasingly important role in a variety of practical applications, it is necessary to pursue efforts to develop more language-specific transcription systems based on linguistic knowledge.

References

Nasreen Abduljaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic Cross Language Information Retrieval. CIKM 2003: 139-146

M. Arbabi, S.M. Fischthal, V.C. Cheng and E. Bart. 1994. Algorithms for Arabic name transliteration. *IBM J. Res. Develop.,* 38(2)

Moustafa Elshafei, Husni Al-Muhtaseb, Mansour Al-Ghamdi. 2006. Machine Generation of Arabic Diacritical Marks. MLMTA 2006: 128-133

Ali Farghaly. 2004 Computer Processing of Arabic Script-based Languages: Current State and Future Directions. COLING 2004

Martin Kay Stanford University. 2004. Arabic Script-based Languages deserve to be studied linguistically. COLING 2004.

D. Vergyri and K. Kirchhoff. 2004. Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. COLING Workshop on Arabic-script Based Languages, Geneva, Switzerland, 2004.