Translating Names and Technical Terms in Arabic Text

Bonnie Glover Stalls and Kevin Knight
USC Information Sciences Institute
Marina del Rey, CA 90292
bgs@isi.edu, knight@isi.edu

Abstract

It is challenging to translate names and technical terms from English into Arabic. Translation is usually done phonetically: different alphabets and sound inventories force various compromises. For example, *Peter Streams* may come out as بيتر ستريمز `bytr strymz`. This process is called transliteration. We address here the reverse problem: given a foreign name or loanword in Arabic text, we want to recover the original in Roman script. For example, an input like بيتر ستريمز `bytr strymz` should yield an output like *Peter Streams*. Arabic presents special challenges due to unwritten vowels and phonetic-context effects. We present results and examples of use in an Arabic-to-English machine translator.

1 Introduction

Translators must deal with many problems, and one of the most frequent is translating proper names and technical terms. For language pairs like Spanish/English, this presents no great challenge: a phrase like *Antonio Gil* usually gets translated as *Antonio Gil*. However, the situation is more complicated for language pairs that employ very different alphabets and sound systems, such as Japanese/English and Arabic/English. Phonetic translation across these pairs is called transliteration.

(Knight and Graehl, 1997) present a computational treatment of Japanese/English transliteration, which we adapt here to the case in Arabic. Arabic text, like Japanese, frequently contains foreign names and technical terms that are translated phonetically. Here are some examples from newspaper text:[1]

Jim Leighton
جيم لايتون
(`jym l!ytwn`)

Wall Street
و ول ستريت
(`wwl stryt`)

Apache helicopter
هليكوبتر اباتشي
(`hlykwbtr !b!tshy`)

[1]The romanization of Arabic orthography used here consists of the following consonants: ! (*alif*), b, t, th, j, H, x, d, dh, r, z, s, sh, S, D, T, Z, @ (*@ayn*), G (*Gayn*), f, q, k, l, m, n, =h, w, y, ' (*hamza*). !, w, and y also indicate long vowels. !' and !+ indicate *hamza* over *alif* and *hamza* under *alif*, respectively.

It is not trivial to write an algorithm for turning English letter sequences into Arabic letter sequences, and indeed, two human translators will often produce different Arabic versions of the same English phrase. There are many complexity-inducing factors. Some English vowels are dropped in Arabic writing (but not all). Arabic and English vowel inventories are also quite different—Arabic has three vowel qualities (`a`, `i`, `u`) each of which has short and long variants, plus two diphthongs (`ay`, `aw`), whereas English has a much larger inventory of as many as fifteen vowels and no length contrast. Consonants like English `D` are sometimes dropped. An English `S` sound frequently turns into an Arabic `s`, but sometimes into `z`. English `P` and `B` collapse into Arabic `b`; `F` and `V` also collapse to `f`. Several English consonants have more than one possible Arabic rendering—`K` may be Arabic `k` or `q`, `t` may be Arabic `t` or `T` (`T` is pharyngealized `t`, a separate letter in Arabic). Human translators accomplish this task with relative ease, however, and spelling variations are for the most part acceptable.

In this paper, we will be concerned with a more difficult problem—given an Arabic name or term that has been translated from a foreign language, what is the transliteration source? This task challenges even good human translators:

?
مايك ماكوري
(`m!yk m!kwry`)

?
أوبك
(`!'wbk`)

?
انترنت اكسبلورر
(`!ntrnt !ksblwrr`)

(Answers appear later in this paper).

Among other things, a human or machine translator must imagine sequences of dropped English vowels and must keep an open mind about Arabic letters like b and f. We call this task *back-transliteration.* Automating it has great practical importance in Arabic-to-English machine translation, as borrowed terms are the largest source of text phrases that do not appear in bilingual dictionaries. Even if an English term is listed, all of its possible Arabic variants typically are not. Automation is also important for machine-assisted translation, in which the computer may suggest several translations that a human translator has not imagined.

2 Previous Work

(Arbabi et al., 1994) developed an algorithm at IBM for the automatic forward transliteration of Arabic personal names into the Roman alphabet. Using a hybrid neural network and knowledge-based system approach, this program first inserts the appropriate missing vowels into the Arabic name, then converts the name into a phonetic representation, and maps this representation into one or more possible Roman spellings of the name. The Roman spellings may also vary across languages (*Sharif* in English corresponds to *Chérife* in French). However, they do not deal with back-transliteration.

(Knight and Graehl, 1997) describe a back-transliteration system for Japanese. It comprises a generative model of how an English phrase becomes Japanese:

1. An English phrase is written.
2. A translator pronounces it in English.
3. The pronunciation is modified to fit the Japanese sound inventory.
4. The sounds are converted into the Japanese katakana alphabet.
5. Katakana is written.

They build statistical models for each of these five processes. A given model describes a mapping between sequences of type A and sequences of type B. The model assigns a numerical score to any particular sequence pair a and b, also called the probability of b given a, or $P(b|a)$. The result is a bi-directional translator: given a particular Japanese string, they compute the n most likely English translations.

Fortunately, there are techniques for coordinating solutions to sub-problems like the five above, and for using generative models in the reverse direction. These techniques rely on probabilities and Bayes' Rule.

For a rough idea of how this works, suppose we built an English phrase generator that produces word sequences according to some probability distribution $P(w)$. And suppose we built an English pronouncer that takes a word sequence and assigns it a set of pronunciations, again probabilistically, according to some $P(e|w)$. Given a pronunciation e, we may want to search for the word sequence w that maximizes $P(w|e)$. Bayes' Rule lets us equivalently maximize $P(w) \cdot P(e|w)$, exactly the two distributions just modeled.

Extending this notion, (Knight and Graehl, 1997) built five probability distributions:

1. $P(w)$ – generates written English word sequences.
2. $P(e|w)$ – pronounces English word sequences.
3. $P(j|e)$ – converts English sounds into Japanese sounds.
4. $P(k|j)$ – converts Japanese sounds to katakana writing.
5. $P(o|k)$ – introduces misspellings caused by optical character recognition (OCR).

Given a Japanese string o they can find the English word sequence w that maximizes the sum over all e, j, and k, of

$$P(w) \cdot P(e|w) \cdot P(j|e) \cdot P(k|j) \cdot P(o|k)$$

These models were constructed automatically from data like text corpora and dictionaries. The most interesting model is $P(j|e)$, which turns English sound sequences into Japanese sound sequences, e.g., S AH K ER (*soccer*) into s a kk a a.

Following (Pereira and Riley, 1997), $P(w)$ is implemented in a weighted finite-state acceptor (WFSA) and the other distributions in weighted finite-state transducers (WFSTs). A WFSA is a state/transition diagram with weights and symbols on the transitions, making some output sequences more likely than others. A WFST is a WFSA with a pair of symbols on each transition, one input and one output. Inputs and outputs may include the empty string. Also following (Pereira and Riley, 1997), there is a general composition algorithm for constructing an integrated model $P(x|z)$ from models $P(x|y)$ and $P(y|z)$. They use this to combine an observed Japanese string with each of the models in turn. The result is a large WFSA containing all possible English translations, the best of which can be extracted by graph-search algorithms.

3 Adapting to Arabic

There are many interesting differences between Arabic and Japanese transliteration. One is that Japanese uses a special alphabet for borrowed foreign names and borrowed terms. With Arabic, there are no such obvious clues, and it is difficult to determine even whether to attempt a back-transliteration, to say nothing of computing an accurate one. We will not address this problem directly here, but we will try to avoid inappropriate transliterations. While the Japanese system is robust—everything gets some transliteration—we will build a deliberately more brittle Arabic system, whose failure signals that transliteration may not be the correct option.

While Japanese borrows almost exclusively from English, Arabic borrows from a wider variety of languages, including many European ones. Fortunately, our pronunciation dictionary includes many non-English names, but we should expect to fail more often on transliterations from, say, French or Russian.

Japanese katakana writing seems perfectly phonetic, but there is actually some uncertainty in how phonetic sequences are rendered orthographically. Arabic is even less deterministically phonetic; short vowels do not usually appear in written text. Long vowels, which are normally written in Arabic, often but not always correspond to English stressed vowels; they are also sometimes inserted in foreign words to help disambiguate pronunciation. Because true pronunciation is hidden, we should expect that it will be harder to establish phonetic correspondences between English and Arabic.

Japanese and Arabic have similar consonant-conflation problems. A Japanese r sound may have an English r or l source, while an Arabic b may come from p or b. This is what makes back-transliteration hard. However, a striking difference is that while Japanese writing adds extra vowels, Arabic writing deletes vowels. For example:[2]

Henriette → H EH N R IY EH T (English)
→ h e n o r i e tt o (Japanese)
→ =h n r y t (Arabic)

This means potentially much more ambiguity; we have to figure out which Japanese vowels *shouldn't* be there (deletion), but we have to figure out which Arabic vowels *should* be there (addition).

For cases where Arabic has two potential mappings for one English consonant, the ambiguity does not matter. Resolving that ambiguity is bonus when going in the backwards direction—English T, for example, can be safely posited for Arabic t or T without losing any information.

4 New Model for Arabic

Fortunately, the first two models of (Knight and Graehl, 1997) deal with English only, so we can reuse them directly for Arabic/English transliteration. These are P(w), the probability of a particular English word sequence and P($e|w$), the probability of an English sound sequence given a word sequence. For example, P(*Peter*) may be 0.00035 and P(P IY T ER|*Peter*) may be 1.0 (if *Peter* has only one pronunciation).

To follow the Japanese system, we would next propose a new model P($q|e$) for generating Arabic phoneme sequences from English ones, and another model P($a|q$) for Arabic orthography. We would then attempt to find data resources for estimating these probabilities. This is hard, because true Arabic pronunciations are hidden and no databases are available for directly estimating probabilities involving them.

Instead, we will build only one new model, P($a|e$), which converts English phoneme sequences directly into Arabic writing. We might expect the model to include probabilities that look like:

P(f|F) = 1.0
P(t|T) = 0.7
P(T|T) = 0.3
P(s|S) = 0.9
P(z|S) = 0.1
P(w|AH) = 0.2
P(nothing|AH) = 0.4
P(!+|AH) = 0.4

The next problem is to estimate these numbers empirically from data. We did not have a large bilingual dictionary of names and terms for Arabic/English, so we built a small 150-word dictionary by hand. We looked up English word pronunciations in a phonetic dictionary, generating the English-phoneme-to-Arabic-writing training data shown in Figure 1.

We applied the EM learning algorithm described in (Knight and Graehl, 1997) on this data, with one variation. They required that each English sound

[2]The English phonemic representation uses the phoneme set from the online Carnegie Mellon University Pronouncing Dictionary, a machine-readable pronunciation dictionary for North American English (http://www.speech.cs.cmu.edu/cgi-bin/cmudict).

((AE N T OW N IY OW)	(! ' n T w n y w))
((AE N T AH N IY)	(! ' n T w n y))
((AA N W AA R)	(! ' n w r))
((AA R M IH T IH JH)	(! ' r m y t ! j))
((AA R N AA L D OW)	(! r n l d w))
((AE T K IH N Z)	(! ' t k y n z))
((K AO L V IY N OW)	(k ! l f y n w))
((K AE M ER AH N)	(k ! m r ! n))
((K AH M IY L)	(k m y l))
((K AA R L AH)	(k ! r l !))
((K AE R AH L)	(k ! r w l))
((K EH R AH L AY N)	(k ! r w l y n))
((K EH R AH L IH N)	(k ! r w l y n))
((K AA R V ER)	(k ! r f r))
((K AE S AH L)	(k ! s l))
((K R IH S)	(k r y s))
((K R IH S CH AH N)	(k r y s t s h n))
((K R IH S T AH F ER)	(k r y s t w f r))
((K L AO D)	(k l w d))
((K L AY D)	(k l ! y d))
((K AA K R AH N)	(k w k r ! n))
((K UH K)	(k w k))
((K AO R IH G AH N)	(k w r y G ! n))
((EH B ER HH AA R T)	(! + y b r = h ! r d))
((EH D M AH N D)	(! + d m w n))
((EH D W ER D)	(! ' d w ! r d))
((AH L AY AH S)	(! + l y ! s)
((IH L IH Z AH B AH TH)	(! + l y z ! b y t h))

Figure 1: Sample of English phoneme to Arabic writing training data.

produce at least one Japanese sound. This worked because Japanese sound sequences are always longer than English ones, due to extra Japanese vowels. Arabic letter sequences, on the other hand, may be shorter than their English counterparts, so we allow each English sound the option of producing no Arabic letters at all. This puts an extra computational strain on the learning algorithm, but is otherwise not difficult.

Initial results were satisfactory. The program learned to map English sounds onto Arabic letter sequences, e.g.: *Nicholas* onto نيكولاس `nykwl!s` and *Williams* onto وليمز `wlymz`.

We applied our three probabilistic models to previously unseen Arabic strings and obtained the top n English back-transliteration for each, e.g.,

`byfrly`	*Beverly Beverley*
`bykr`	*Baker Picker Becker*
`!'dw!r`	*Edward Edouard Eduard*
`=hdswn`	*Hudson Hadson Hodson*
`=hwknz`	*Hawkins Huggins Huckins*

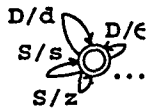
We then detected several systematic problems with our results, which we turn to next.

5 Problems Specific to Arabic

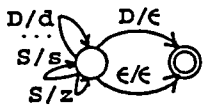
One problem was the production of many wrong English phrases, all containing the sound `D`. For example, the Arabic sequence فريمان `frym!n` yielded two possible English sources, *Freeman* and *Friedman*. The latter is incorrect. The problem proved to be that, like several vowels, an English `D` sound sometimes produces no Arabic letters. This happens in cases like أدوار *Edward* `!'dw!r` and ريموند *Raymond* `rymwn`. Inspection showed that `D` should only be dropped in word-final position, however, and not in the middle of a word like *Friedman*.

This brings into question the entire shape of our $P(a|e)$ model, which is based on a substitution of Arabic letters for an English sound, independent of that sound's context. Fortunately, we could incorporate an only-drop-final-`D` constraint by extending the model's transducer format.

The old transducer looked like this:



While the new transducer looks like this:



Whenever `D` produces no letters, the transducer finds itself in a final state with no further transitions. It can consume no further English sound input, so it has, by definition, come to the end of the word.

We noticed a similar effect with English vowels at the end of words. For example, the system suggested both *Manuel* and *Manuela* as possible sources for مانويل `m!nwyl`. *Manuela* is incorrect; we eliminated this frequent error with a technique like the one described above.

A third problem also concerned English vowels. For Arabic أوكتافيو `!'wkt!fyw`, the system produced both *Octavio* and *Octavia* as potential sources, though the latter is wrong. While it is possible for the English vowel `AH` (final in *Octavia*) to produce Arabic `w` in some contexts (e.g., روجر `rwjr`/*Roger*), it cannot do so at the end of a word. `EH` and `AA` have the same property. Furthermore, none of those three vowels can produce the letter y when in word-final position. Other vowels like `IY` may of course do so.

We pursued a general solution, replacing each in-

stance of an English vowel in our training data with one of three symbols, depending on its position in the word. For example, an AH in word-initial position was replaced by AH-S; word-final AH was replaced by AH-F; word-medial was AH. This increases our vowel sound inventory by a factor of three, and even though AH might be pronounced the same in any position, the three distinct AH- symbols can acquire different mappings to Arabic. In the case of AH, learning revealed:

$$P(\texttt{w}|\texttt{AH}) = 0.288$$
$$P(\texttt{nothing}|\texttt{AH}) = 0.269$$
$$P(\texttt{!}|\texttt{AH}) = 0.269$$
$$P(\texttt{y}|\texttt{AH}) = 0.173$$

$$P(\texttt{!}|\texttt{AH-F}) = 0.765$$
$$P(\texttt{\&}|\texttt{AH-F}) = 0.176$$
$$P(\texttt{=h}|\texttt{AH-F}) = 0.059$$

$$P(\texttt{!+}|\texttt{AH-S}) = 0.5$$
$$P(\texttt{!y}|\texttt{AH-S}) = 0.5$$
$$P(\texttt{!'}|\texttt{AH-S}) = 0.25$$

We can see that word-final AH can never be dropped. We can also see that word-initial AH can be dropped; this goes beyond the constraints we originally envisioned. Figure 2 shows the complete table of sound-letter mappings.

We introduced just enough context in our sound mappings to achieve reasonable results. We could, of course, introduce left and right context for every sound symbol, but this would fragment our data; it is difficult to learn general rules from a handful of examples. Linguistic guidance helped us overcome these problems.

6 EXAMPLE

Here we show the internal workings of the system through an example. Suppose we observe the Arabic string br!nstn. First, we pass it though the $P(a|e)$ model from Figure 2, producing the network of possible English sound sequences shown in Figure 3. Each sequence e_i could produce (or "explain") br!nstn and is scored with $P(\texttt{br!nstn}|\ e_i)$. For example, $P(\texttt{br!nstn}|\texttt{BRAENSTN}) = 0.54$.

e	a	$P(a\|e)$	a	$P(a\|e)$	a	$P(a\|e)$
AA	!	0.652	w	0.217	*	0.131
AA-S	!'	0.625	!'w	0.125	!'H	0.125
	!	0.125				
AE	!	0.889	*	0.111		
AE-S	!'	0.889	!	0.111		
AH	w	0.288	*	0.269	!	0.269
	y	0.173				
AH-F	!	0.765	&	0.176	=h	0.059
AH-S	!+	0.5	!y	0.25	!'	0.25
AO	w	0.8	y	0.1	!	0.1
AY	y	0.8	!y	0.2		
AY-F	y	1.0				
AY-S	!+	1.0				
B	b	1.0				
CH	x	0.5	tsh	0.5		
D	d	0.968	*	0.032		
EH	*	0.601	y	0.25	!	0.1
	h	0.049				
EH-S	!'	0.667	!+	0.167	!+y	0.167
ER	r	0.684	yr	0.182	wr	0.089
	!+r	0.045				
EY	y	0.444	!	0.333	*	0.111
	!@y	0.111				
EY-F	!y	0.639	y	0.361		
EY-S	!'	0.5	!	0.5		
F	f	1.0				
G	G	0.833	k	0.167		
HH	=h	0.885	=	0.113		
IH	y	0.868	*	0.079	!	0.026
	r	0.026				
IH-S	!	0.375	!+	0.25	!y	0.125
	!'	0.125	!+y	0.125		
IY	y	0.909	*	0.064	h	0.027
IY-F	y	1.0				
IY-S	!+y	1.0				
JH	j	1.0				
K	k	0.981	+q	0.019		
L	l	1.0				
M	m	1.0				
N	n	1.0				
NG	nG	1.0				
OW	w	0.714	ww	0.286		
OW-F	w	1.0				
OW-S	!'w	1.0				
P	b	1.0				
R	r	0.98	y	0.02		
S	s	0.913	z	0.087		
SH	Ty	0.333	shy	0.333	sh	0.333
T	t	0.682	T	0.273	d	0.045
TH	th	1.0				
UH	w	1.0				
UW	w	1.0				
UW-F	w	1.0				
V	f	1.0				
W	w	0.767	w!	0.121	fy	0.111
Y	y	1.0				
Z	z	0.75	s	0.25		

Figure 2: English sounds (in capitals) with probabilistic mappings to Arabic sound sequences, as learned by estimation-maximization.

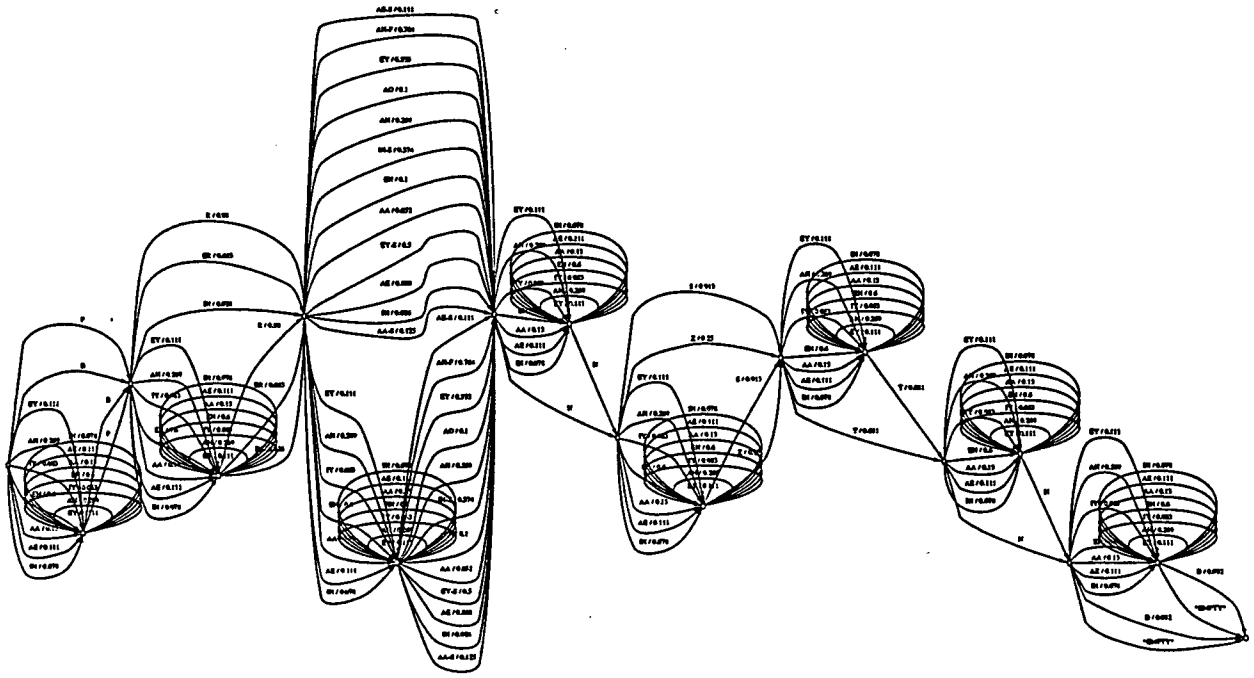


Figure 3: English sound sequences corresponding to Arabic br!nstn. Each path through the lattice represents one possible sequence.

Arabic input: b r ! n s t n

Top 20 English pronunciations	P(a|e)
B R AE N S T N	0.541074
P R AE N S T N	0.541074
B R AH-F N S T N	0.465519
P R AH-F N S T N	0.465519
B R AA N S T N	0.397275
P R AA N S T N	0.397275
B ER AE N S T N	0.377096
P ER AE N S T N	0.377096
P R AE N S EH T N	0.324645
B R AE N EH S T N	0.324645
B EH R AE N S T N	0.324645
P R AE EH N S T N	0.324645
P R AE N S T EH N	0.324645
P EH R AE N S T N	0.324645
B R AE N S T N EH	0.324645
B R EH AE N S T N	0.324645
P R AE N EH S T N	0.324645
P R AE N S T N EH	0.324645
P R EH AE N S T N	0.324645
EH B R AE N S T N	0.324645
...	

Next, we pass this network through the P(e|w) model to produce a new network of English phrases. Finally, we re-score this network with the P(w) model. This marks a preference for common English words/names over uncommon ones. Here are the top n sequences at each stage:

Top 20 word sequences	P(e|w) · P(a|e)
BRANN STEN	0.324645
BRAN STEN	0.324645
BRONN STEN	0.238365
PUR ANSE TEN	0.226257
PUR ANSE TENN	0.226257
PUR ANNE STEN	0.226257
PERE ANSE TEN	0.226257
PUR ANN STEN	0.226257
PER ANSE TEN	0.226257
PERE ANSE TENN	0.226257
PER ANSE TENN	0.226257
PERE ANNE STEN	0.226257
PER ANNE STEN	0.226257
PERE ANN STEN	0.226257
PUR AN STEN	0.226257
PER ANN STEN	0.226257
PERE AN STEN	0.226257
PERE AHN STEN	0.226257
PUR AHN STEN	0.226257
PER AN STEN	0.226257
...	

Re-scored word sequences	P(w) · P(e|w) · P(a|e)
BRONSTON	8.63004e-07
BRONSTEIN	7.29864e-08
BRAUNSTEIN	1.11942e-08

7 Results and Discussion

We supplied a list of 2800 test names in Arabic to our program and received translations for 900 of them. Those not translated were frequently not foreign names at all, so the program is right to fail in many such cases. Sample results are shown in Figure 4.

The program offers many good translations but still makes errors of omission and commission. Some of these errors show evidence of lexical or orthographic influence or of interference from other languages (such as French).

English G is incorrectly produced from its voiceless counterpart in Arabic, k. For example, كريس krys comes out correctly as *Chris* and *Kris* but also, incorrectly, as *Grace*. The source of the G-k correspondence in the training data is the English name AE L AH G Z AE N D ER *Alexander*, which is الكسندر !lksndr in our training corpus. A voiced fricative G is available in Arabic, which in other contexts corresponds to the English voiced stop G, although it, too, is only an approximation. It appears that orthographic English X is perceived to correspond to Arabic ks, perhaps due partly to French influence. Another possible influence is the existing Arabic name اسكندر !skndr (which has k), from the same Greek source as the English name *Alexander*, but with metathesis of k and s.

Sometimes an Arabic version of a foreign name is not strictly a transliteration, but rather a translation or lexicalized borrowing which has been adapted to Arabic phonological patterns. For example, the name *Edward* is found in our data as أدوارد !'dw!rd, أدوار !'dw!r, and إدوار !+dw!r. The last version, an Arabicization of the original Anglo-Saxon name, is pronounced *Idwar*. The approach taken here is flexible enought to find such matches.

Allowing the English sound D to have a zero match word-finally (also a possible French influence) proves to be too strong an assumption, leading to matches such as: !'lfr *Oliver Alfred*. A revised rule would allow the D to drop word-finally only when immediately preceded by another consonant (which consonant could further be limited to a sonorant).

Another anomaly which is the source of error is the mapping of English CH to Arabic x, which carries equal weight (0.5) to the mapping of CH to Arabic tsh (0.5). This derives from the name *Koch*, which in Arabic is كوخ kwx, as in the German pronunciation of the name, as opposed to the English pronunciation. This kind of language interference can be minimized by enlarging the training data set.

```
!'bwrzq
!'by        ABBEY ABBY ABBIE
!'byD
!'d!mz      ADAMS ADDAMS
!'drys      EDRIS
!'dw!r      EDWARD EDOUARD EDUARD
!'dw!rd     EDWARD EDOUARD EDUARD
!'dwnys
!'dyb
!'f!y&      AVERA
!'fr!H
!'fyn!sh
!'krm
!'l!n       ALAN ALLEN ALLAN
!'lbrt      ALBERT ALPERT ELBERT
!'lbrty     ALBERTI ALBERTY
!'lbyr      ALPER
!'lf!rw     ALVARO ALFARO ALVERO
!'lfr       OLIVER OLIVER ALFRED
!'lksndr    ALEXANDER ALEXANDER ALEXANDRE
!'ln        ALAN ALLEN ALLAN
!'lys       ELLIS ALICE LAS
!'lyswn     ALLISON ALISON ELLISON
!'mjd
!'mnwn
!'mrz!q&
!'mst
!'mws       AMOS AMOSS
!'mykw      AMICO AMERCO
!'myl       EMIL EMILE EMAIL
!'mym&
!'myn       AMMAN AMIN AMMEEN
!'myn&
!'myr       AMER AMIR AMOR
!'n!        ANA ANNA ANA
!'nGz       INIGUEZ
!'nTw!n     ANTOINE ANTOINE
!'nTw!nyt   ANTOINETTE ANTOINETTE
!'nTwn      ANTON ANTOON ANTOINE
!'nTwny     ANTONY ANTONI ANTONE
!'nTwny!    ANTONIA
!'nTwnyw    ANTONIO ANTONIU
!'ndrw      ANDREW ANDREU
!'ndry=h    ANDREA ANDREA ANDRIA
!'ndryyf
```

Figure 4: Sample program results for English translations of names written in Arabic.

English orthography appears to have skewed the training data in the English name *Frederick*, pronounced F R EH D R IH K. In Arabic we have فرديرك frdyrk as well as frydryk, frdyryk and frdryk for this name. The English spelling has three vowels, but the English phonemic representation lacks the middle vowel. But some Arabic versions have a (long) "vowel" (y) in the middle vowel position, leading in the training data to the incorrect mapping English R to Arabic y. This results in incorrect translations, such as *Racine* for ysyn.

As might be expected when the sound system of one language is being re-interpreted in another, Arabic transliteration is not always attuned to the subtleties of English phonotactic variation, especially when the variants are not reflected in English orthography. An example is the assimilation in voicing of English fricatives to an immediately preceding stop consonant. In *James*, pronounced JH EY M Z, the final consonant is a voiced Z although it is spelled with the voiceless variant, *s*. In this case, Arabic follows the English orthography rather than pronunciation, transliterating it جيمس jyms. Similarly, *Horowitz* is pronounced HH AO R OW IH T S in English, with a final devoiced S rather than the voiced variant *z* present in the spelling, whereas the Arabic transliteration follows the English spelling, هورو وتز =hwrwwytz. The present version of the program applies these variant correspondences indiscriminately, such that سايمون s!ymwn is translated as *Simon* or *Zyman*. Separating out these correspondences according to their positions in the word, as was done with the vowels, would help to rectify this, by reducing the probability of an S—z correspondence in less likely positions (e.g., initial position).

Some Arabic transliterations carry the imprint of English spelling even when it departs even farther from the pronunciation. For example, غراهام Gr!=h!m is an Arabic transliteration for the English name *Graham*, pronounced G R AE M. (an alternative is the Arabic غرام Gr!m). These mappings were not found by the program (even though they might be readily evident to a human). This kind of spelling-transliteration lies outside of the phonemic correspondences to Arabic orthography that the program has learned.

Vowels are still a problem, even when they are distinguished by their position in the word. In the test cases given in the Introduction, (answers are *Mike McCurry*, *OPEC*, and *Internet Explorer*), the quality of the Arabic vowels, when present, matches the English vowels fairly well. However, as can be seen from names like *Frederick*, the decision as to whether or not to insert vowel is arbitrary and somewhat dependent on English orthography, which influences the quality as well as position of the Arabic vowel. Medial English AH, for example, is normally ! (*alif* but can also be found in Arabic as w or y (e.g, English *Jeremy*, pronounced JH EH R AH M IY is written in Arabic as jyrymy). This results in incorrect translations, such as *Amman* for Arabic !'myn.

In this initial model, English vowel stress was not represented. Because long vowels in Arabic are usually stressed, one might expect that English stressed vowels would be equated with Arabic long vowels for purposes of transliteration. However, our data suggest that English stress does not have a strong correlation with Arabic long vowel placement in transliterated names. For example, *Kevin* mapped to كيفين kyfyn and كفين kfyn, but not كيفن kyfn. If stress were a factor here and were interpreted as a long vowel, كيفن kyfn would be predicted as a preferred transliteration based on the phonemic representation of *Kevin* as K EH1 V IH N (where "1" indicates primary stress). Similarly, فيكتور fyktwr and فكتور fktwr were found for *Victor* but not the expected فيكتر fyktr. كينث kynth is found, but so are كينيث kynyth, and كنيث knyth. In syllable-final position at least, it appears that stress does not outweigh other factors in Arabic vowel placement. However, the relation of English stress to Arabic vowel placement in other positions might be used to rule out unlikely translations (such as *Camille* with final stress for Arabic كامل k!ml) and deserves further study.

All of these observations point to places where the system can be improved. A larger training data set, more selected contextual mappings, and refinement of linguistic rules are all potential ways to capture these improvements within the finite-state framework, and we hope to study them in the near future.

References

Mansur Arbabi, Scott M. Fischthal, Vincent C. Cheng, and Elizabeth Bart. 1994. Algorithms for Arabic name transliteration. *IBM Journal of Research and Development*, 38(2):183–193.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 128–135. Morgan Kaufmann.

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*, pages 431–453. MIT Press.