English to Arabic Transliteration for Information Retrieval: A Statistical Approach

Nasreen AbdulJaleel and Leah S. Larkey
Center for Intelligent Information Retrieval
Computer Science, University of Massachusetts
140 Governors Drive
Amherst, MA 01003-4610
1-413-545-
{nasreen, larkey}@cs.umass.edu

ABSTRACT

The most important query words in information retrieval are often proper names. In cross language retrieval a user issues a query in one language to search a collection in a different language. If the two languages use the same alphabet, the same proper names can be found in either language. However, if the two languages use different alphabets, the names must be transliterated or rendered in the other alphabet. In this paper, we present a method for automatically learning a transliteration model from a sample of name pairs in two languages. We evaluate the model and compare three versions for English to Arabic transliteration.

1. INTRODUCTION

In this paper, we present and evaluate a simple statistical technique for English to Arabic transliteration. This technique requires no heuristics or linguistic knowledge of either language. This technique learns translation probabilities between English and Arabic characters from a training sample of pairs of transliterated words from the two languages. Based on these probabilities, it generates Arabic transliterations for unknown English words. We compare a context-dependent version of the model with a context-independent version. This technique could be used for any language pair.

Transliteration is the representation of a word or phrase in the closest corresponding letters or characters of a different alphabet or language so that the pronunciation is as close as possible to the original word or phrase. The method of transliteration depends on the characteristics of the source and target languages (in this case English and Arabic) and on the larger purpose the transliterator is meant to serve. Transliteration techniques are useful in several areas including machine translation and information retrieval.

It is commonly observed that proper names are important query words in information retrieval. For example, in the list of queries for TREC 2001 cross language track [6], all 25 queries contained proper names. In cross language retrieval, a user queries a system in one language to find documents in a different language. There are three common approaches to this problem: machine translation of the query, dictionary-based translation and expansion of the query, and retrieval using a cross-lingual language model. For all of these approaches, if a query word is not found in the dictionary or model, one needs a way to render the unknown word. When the query language and the document language share the same alphabet it is sufficient to use the word from the query language as is. However, when the two languages have different alphabets, the best strategy is to transliterate the word from the query language alphabet to the document language alphabet.

We have been working on English-Arabic cross language retrieval, using the TREC 2001 corpus of 383,872 Arabic documents from Agence France Presse, covering the years 1994-2000. Newspapers contain many proper names that are not found in bilingual dictionaries. Here are a few examples from the TREC corpus (see the Appendix for an explanation of the phonetic notation used in the third column):

New York	نيو يورك	*(nyü yürk)*
Wall Street	وال ستريت	*(üal strEt)*
Sarah Ferguson	ساره فيرغسون	*(sar-& fErgsün)*

These examples show that transliteration is commonly used for proper names, even when the words could be translated. "Wall Street" in Arabic could be represented by translating "wall" and "street," but it is not.

In the case of English-to-Arabic transliteration, an English word can have more than one equivalent in Arabic. There are several reason for this indeterminacy:

1. The irregularity of English spelling
2. A name can have more than one correct spelling, for example "Anderson" and "Andersen", or alternative pronunciations, for example the name "Dubois" can be, *dü-'bois* or *dü-'bwä*. Each of these pronunciations has a different spelling in Arabic - "دوبوا", and "دوبويس", respectively.
3. The differences in phonetic inventories of the two languages – for example, Arabic has no *p* sound, but it has two different *t*'s.
4. Short vowels are commonly not represented in Arabic orthography. In transliterating an English word with short vowels, one approach would be to leave the short vowels out of the Arabic transliteration; another would be to write the words using Arabic long vowels. Both approaches are commonly found.

For example, we found four different versions of the name "Clinton" with little effort. Two different translations were found in the TREC2001 corpus:

Clinton	(1)	كلينتون	*(klEntün)*
	(2)	كلينتن	*(klEntn)*

Two web-based machine-translation engines [1][2] and an Arabic Proper Names list available on the web [3] each give a different (single) translation of "Clinton":

Clinton	(3)	كلنتن	*(klntn)*
	(4)	كلينتون	*(klEntün)*
	(5)	كلينطون	*(klEnTün)*

While these are all reasonable transliterations of the word "Clinton", only one of the second group, (4), matches a form of the word found in the TREC corpus. If we had chosen either of the other two resources as sources for transliterations in cross-language queries, we would have failed to find documents mentioning Clinton. All three sources would have failed to find the documents that used spelling (2).

This example makes it clear that a transliterator for information retrieval must generate multiple alternative Arabic spellings, in order to cover the variations that are found in Arabic text.

2. PREVIOUS WORK

Most prior work in Arabic-related transliteration has been for the purpose of machine translation. Arbabi et al. [4] developed a hybrid neural network and knowledge-based system to generate multiple English spellings for Arabic personal names. Knight and Graehl [8] developed a five stage statistical model to do *back transliteration*, that is, recover the original English name from its transliteration into Japanese Katakana. Stalls and Knight [10] adapted this approach for back transliteration from Arabic to English of English names. These systems are very complex, involving a great deal of human design, probably because they were dealing with a more difficult problem than that of forward transliteration. However, as the Clinton example reveals, forward transliteration for information retrieval is not as simple as the problem of forward transliteration for machine translation, in which one reasonable transliteration is good enough.

Darwish et al described a hand-crafted English to Arabic transliteration system [5]. Each English letter was mapped to the closest sounding Arabic letter or letters. These mappings were decided manually. Most English letters were given a single Arabic equivalent; a few had more than one. Darwish's system has the same purpose as ours. However, our system differs in that it learns the mappings automatically from data, one version of our system can take into account phonetic context, and we attempt to evaluate the

performance of our transliterator. Darwish presents no evaluation of his transliteration system.

3. STATISTICAL TRANSLITERATION

3.1 Monogram Transliteration Model

The transliterator is based upon a generative statistical model of how a string of English characters is converted to Arabic characters. In its simplest form, the Arabic string is generated by converting individual English characters (*monograms)*. The model is a set of probability distributions over English and Arabic characters. Each English character *e* can be transliterated to any Arabic character a_i with probability $P(a_i \mid e)$. In addition, the English and Arabic character inventories can each include NULL, which allow the generation of Arabic strings with a different length than the English strings. In practice, most of the probabilities are zero, or small enough to be considered zero. For example, the probability distribution for the character "s" in English might look like this:

$P($ س $|s) =$ .60

$P($ ز $\mid s) =$ .30

$P($ ش $\mid s) =$ .07

$P($ ص $\mid s) =$ .03

The basic issues for such a model are:

1. How to estimate these probabilities, i.e., how to train the model. There are $(m+1)$ $(n+1)$ probabilities to estimate, where m is the number of English characters ($m+1$ includes NULL) and n is the number of Arabic characters.
2. How to use the model to generate new transliterations.

Using the model generatively is very straightforward. One can either generate a single transliteration by taking the highest probability transliteration for each character, or by taking one transliteration for each character at random, according to the character transliteration probabilities. One can generate all the possible transliterations using all the mappings with nonzero probabilities. One can compute the probability for the whole transliteration as the product of individual character probabilities, rank order them, and then select a subset based on some probability threshold, or to take a fixed number of top ranking transliterations.

The first issue, of estimating the probabilities, is also fairly straightforward. In order to estimate $P(a_i \mid e)$, we need a training sample of English-Arabic pairs in which English and Arabic characters are aligned so that we can count up the number of times a_i is aligned with *e*, and divide by the total number of times *e* occurs in the aligned training sample. We discuss alignment in below in section 3.3.

A probability threshold can be applied, so that small probabilities can be set to zero, greatly reducing the number of transliterations possible for a given English character *e*.

The monogram model works well for a surprisingly large proportion of English proper nouns, such as the following two examples in which a character-by-character transliteration gives the correct answer.

New York	نيو يورك	*(nyü yürk)*
Colin	كولين	*(külEn)*

However, many other English to Arabic transliterations cannot be handled with monograms.

3.2 Bigram Transliteration Model

A more sophisticated model would have probabilities for character pairs (*bigrams*) rather than single characters.

For example, the English character "x" as in "Ajax" has no single Arabic character equivalent. It is usually transliterated as a sequence of two Arabic characters, *kaaf* and *seen*, " كس". Conversely, two English characters can be transliterated as a single Arabic character. The English bigram "sh" is usually mapped to the Arabic character *sheen*, "ش", as in:

Bush	بوش	*(büS)*
Parshalville	بارشالفيل	*(barSalfEl)*

There is also the more complex case where two English characters, e_1 and e_2, when they appear as a bigram, e_1e_2, map to a specific sequence of Arabic characters, a_1a_2. For example, "c" usually maps to

kaaf, "ك", and "h" usually maps to *ha*, "ه" or *hha*, "ح". However, when they appear as "ch", they are transliterated as "تش" (tS) as in .

Belchertown	بلتشرتاون	*(bltSrtaün)*
Smidovich	سميدوفيتش	*(smEdüfEtS)*

Bigrams can address these problems. However, such a model has many more parameters. There are potentially $(m+1)^2$ $(n+1)^2$ probabilities to estimate, though in practice, there are far fewer. This is because not all the $(m+1)^2$ $(n+1)^2$ possible bigrams occur in Arabic text.

3.3 Alignment

Training the model requires an aligned training sample so that one can count up the number of times each character *e* (or bigram) is aligned with each Arabic character a_i.

We used GIZA++ [7] to align English words with Arabic words. GIZA++ is an extension of the program GIZA, which is part of the SMT toolkit EGYPT. Its operation is described by Och and Ney [9].

GIZA++ was designed for word alignment of sentence aligned parallel corpora. We used it to do character alignment of word-aligned pairs. Alignment worked very well. A small sample of alignments was examined manually. All of the word-pairs in this sample were correctly aligned.

Here is an example of how GIZA works:

Barnes	بارنز	*(barnz)*

GIZA++ aligns these two words as:

B → ب

A →ا

R →ر

N →ن

E →NULL

S →ز

In addition to GIZA++ alignment, we also considered an "unaligned" baseline in which no alignment is performed. Details are described in the next section.

4. EXPERIMENTS

The data we used to train and test the system was a parallel list of 148,599 English and Arabic proper nouns obtained from NMSU [3]. The list contained both person names and place names.

Four training sets of size 5,000, 10,000, 50,000, and 100,000 were randomly selected from the list.

Five sets of 200 randomly selected words that did not occur in the training sets were used for testing. The results were averaged across these five sets.

Three experimental conditions were compared:

1. Monograms, unaligned
2. Monograms, aligned
3. Bigrams, aligned

4.1 Monogram Conditions

For the "monograms" conditions, the model was a probability distribution over context-independent characters. To train the model, character mappings were counted. For each English character *e*, and each Arabic character *a*, a translation probability $P(a|e)$ was obtained by counting up the number of times *a* was aligned with *e* in the training set and dividing by the total number of times *e* occurred in the training set. Probabilities below 0.05 were set to zero.

In the "unaligned" condition, the training set was restricted to pairs where the English word and its corresponding Arabic word had the same number of characters. We assumed a 1-1 correspondence, and aligned each character in the English word with the character in the same position of the Arabic word. In the unaligned condition, each English character mapped onto exactly one Arabic character in the word pairs.

In the "aligned" condition, we used the alignment tool GIZA++ for character alignment on corresponding words of the training set. From these aligned data, we could count the number of times an English character was mapped onto an Arabic character (or NULL), and derive probabilities once the matrix was filled in.

4.2 Bigram Condition

In the "bigrams" condition, English bigrams, or word pairs e_1e_2 were mapped onto Arabic characters. We did not map English bigrams onto Arabic bigrams, because we needed to control the size of the model. The bigram e_1e_2 should be thought of as e_2 in the context of e_1. The training and testing procedures were very similar to that described for monograms,

except that an extra step was carried out to represent the words as context dependent characters.

An example illustrates this step. The word "Laos" is shown broken into bigrams. Note that the symbol "#" indicates the beginning of the word, and the pair "#L" indicates "L" at the beginning of a word. Word ends were not represented.

Laos → #L, LA, AO, OS

In other words, "Laos" is made up of 4 bigrams, which can be thought of as "word initial L", "A in the context of (or following) L, "O in the context of A," and "S in the context of O."

GIZA++ is used to align these bigrams with the Arabic characters that make up the transliterations for all the word pairs in the training set. From these alignments, counts are obtained for all the $e_1e_2 \rightarrow a$ mappings and converted to probabilities as before.

5. EVALUATION

The same procedure was used to evaluate all three models. After the model was trained, it was tested by generating all possible Arabic transliterations for each word in each test set. In the case of monograms, a probability for each transliteration was obtained by multiplying the individual character mapping probabilities. The alternatives were then ranked by this probability. Test words were scored in two different ways. One method considered only the top scoring transliteration. A test item was considered correctly transliterated if the top ranked transliteration matched the "official" version from the Arabic Proper Names Dictionary. The other method considered all the transliterations. The test item was considered correct if the "official" answer occurred anywhere in the list of alternative transliterations. The second method is more realistic. In using the transliterator in an operational IR system we would actually use many alternative transliterations rather than the top ranked choice.

In the case of bigrams, in order to generate a transliteration for an unknown or test English word, the English word is first represented in bigrams, and transliterations are generated and scored for this bigram representation using the bigram probability model in exactly the same way that we did this for monograms.

The final accuracy reported for each condition was obtained by averaging the accuracy scores (percent correct) from each of five test sets.

Table 1 shows the results for all three conditions, all training set sizes, and both scoring methods. The results for one of the scoring methods, where all transliterations are considered, are shown in Figure 1.

Table 1: Transliteration Accuracy

	Scoring	**Training set size in thousands**		
	Method	5	10	50
Monograms unaligned	1	25.6	21.5	25.6
	All	30.2	26.6	30.6
Monograms aligned	1	34	35.1	36
	All	53.5	54.4	59.4
Bigrams aligned	1	40.6	41.1	43.4
	All	70.3	77.3	81.1

It can be seen clearly from the above table that the accuracy of the model increases with the size of the training set in both aligned conditions. Aligned training is more effective than unaligned training, and bigrams are more effective than monograms. Not surprisingly, the correct answer is obtained more often when all the alternative transliterations are accepted than when only the top-ranked transliteration is accepted.

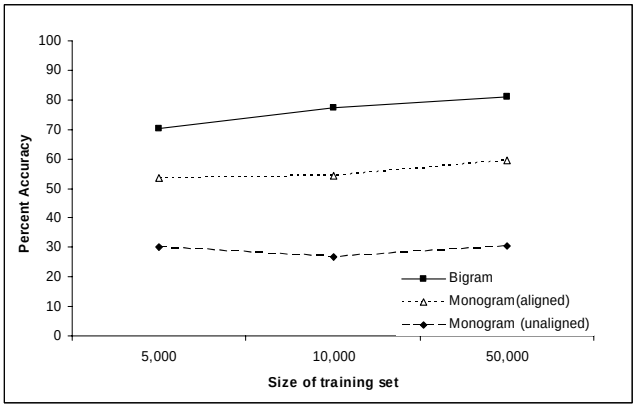

Figure 1: Transliteration accuracy as a function of training set size

6. CONCLUSIONS

We have demonstrated a simple transliteration system that works well, given large amounts of training data. Not surprisingly, context-dependent characters are better than context independent characters, and the system gets better with more training data.

We have so far evaluated this system with respect to how well it can generate correct Arabic transliterations from the Arabic Proper Names dictionary for a test set of English words, after training on a non-overlapping set of word-pairs from the same source. However, the main requirement for information retrieval is that the transliterator find the spellings that are actually used in the Arabic collection being searched.

We are currently building a test set of names that occur in the TREC 2001 corpus so we can address the following questions:

- How often does the transliterator produce Arabic spellings that match the form found in the corpus?
- Are the alternate spellings harmless? Do they not occur in the collection, or do they erroneously match other words in the documents?
- What is the number of transliterations the system should output to achieve the best tradeoff between the need to get the correct string(s) on the list, but not get strings that match other words?
- Is it better to train on multiple sources of word pairs (like those obtained from the transliterators in the online machine translation systems) rather than from a single source.

We have demonstrated a model for transliteration which is purely statistical, and uses no heuristics or hand-tuning. This approach could be used for many different language pairs.

7. APPENDIX

Phonetic notation used for examples

Symbol	Pronunciation
b, f, k, l, m, n, r, s, z	as in normal English
y	as “y” in “yellow”
ü	as “u” in “rule”
a	as “a” in “large”
S	as “s” in “sugar”
E	as “e” in “legal”
&	as “u” in “rule”
g, t, d, T	These sounds are not used in English. They are velarized forms of the English letters.

8. ACKNOWLEDGMENTS

We thank David Fisher for help in installing and using GIZA++. This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1] Ajeeb. http://tarjim.ajeeb.com/ajeeb/

[2] Almisbar. http://www.almisbar.com/salam_trans.html

[3] Arabic Proper Names Dictionary from NMSU. http://crl.nmsu.edu/~ahmed/downloads.html

[4] Arbabi, Mansur, Scott M. Fischthal, Vincent C. Cheng, and Elizabeth Bar. 1994. Algorithms for Arabic name transliteration. IBM Journal of research and Development, 38(2):183-193.

[5] Kareem Darwish, David Doermann, Ryan Jones, Douglas Oard and Mika Rautiainen. 2001. TREC-10 experiments at Maryland: CLIR and video. In *TREC 2001*. Gaithersburg: NIST. http://trec.nist.gov/pubs/trec10/t10_proceedings.html

[6] Gey, F. C. and Oard, D. W. 2001. The TREC-2001 cross-language information retrieval track:

Searching Arabic using English, French, or Arabic queries. In *TREC 2001*. Gaithersburg: NIST. http://trec.nist.gov/pubs/trec10/t10_proceedings.html

[7] GIZA++. http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html

[8] Knight, Kevin and Graehl, Jonathan. 1997. Machine transliteration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 128-135. Morgan Kaufmann.

[9] Och, Franz Josef and Hermann Ney. October 2000. Improved Statistical Alignment Models. *Proc. of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 440-447, Hongkong, China.

[10] Stalls, Bonnie Glover and Kevin Knight. 1998. Translating names and technical terms in Arabic text. http://citeseer.nj.nec.com/glover98translating.html