

# Approaches to Arabic Name Transliteration and Matching in the DataFlux Quality Knowledge Base

**Brant N. Kay**  
 SAS Institute Inc.  
 100 SAS Campus Drive  
 Cary, NC 27513  
 brant.kay@sas.com

**Brian C. Rineer**  
 SAS Institute Inc.  
 100 SAS Campus Drive  
 Cary, NC 27513  
 brian.rineer@sas.com

## Abstract

This paper discusses a hybrid approach to transliterating and matching Arabic names, as implemented in the DataFlux Quality Knowledge Base (QKB), a knowledge base used by data management software systems from SAS Institute, Inc. The approach to transliteration relies on a lexicon of names with their corresponding transliterations as its primary method, and falls back on PERL regular expression rules to transliterate any names that do not exist in the lexicon. Transliteration in the QKB is bi-directional; the technology transliterates Arabic names written in the Arabic script to the Latin script, and transliterates Arabic names written in the Latin script to Arabic. Arabic name matching takes a similar approach and relies on a lexicon of Arabic names and their corresponding transliterations, falling back on phonetic transliteration rules to transliterate names into the Latin script. All names are ultimately rendered in the Latin script before matching takes place. Thus, the technology is capable of matching names across the Arabic and Latin scripts, as well as within the Arabic script or within the Latin script. The goal of the authors of this paper was to build a software system capable of transliterating and matching Arabic names across scripts with an accuracy deemed to be acceptable

according to internal software quality standards.

## 1 Introduction

The challenges inherent to transliterating Arabic names from the Latin script to the Arabic script lie in the fact that there are many seemingly arbitrary ways to spell Arabic names using Latin characters. Halpern (2007) attributes this arbitrariness to the fact that certain Arabic consonant sounds simply do not exist in English, so they are represented in different ways using the Latin script. He also notes that dialectical differences in vowel pronunciation contribute to the variety of Latin spellings. Because there are often several Latin variants of a single Arabic name, it is difficult to successfully transliterate them from Latin to Arabic using a rule-based approach. Take, for example, the name محمد (Latin: *Mohammed*). The single Arabic representation of this name, محمد, can be spelled in several ways using the Latin script. Alternatives include:

*Mohamad*  
*Mohamed*  
*Muhamad*  
*Muhamed*  
*Muhammet*  
*Mohammad*  
*Mohammed*  
*Muhammad*  
*Muhammed*

Given the variety of spellings in these alternatives, it becomes clear why a lexically-based approach is

necessary to transliterate such names from Latin to Arabic -- rules cannot capture the arbitrary nature of Arabic name orthography as it is rendered using Latin characters. To illustrate this assertion, let's focus on only the two variants *Muhammet* and *Muhammed*. These variants are a minimal pair differing only by their final consonant ('T' or 'D'). The sounds for both 'T' and 'D' are rendered in Arabic as د at the end of the name محمد. One might therefore deduce that a rule can be devised to transform 'T' and 'D' to د at the end of a word. However, mapping both 'T' and 'D' to the Arabic character د is not always appropriate in the word-final context. For instance, the name *Falahat* in Arabic is فلاحت. Mapping the final 'T' in *Falahat* to د would produce فلاحد, which is not a valid transliteration of *Falahat*. To allow for such idiosyncrasies, a list must be built of all known Latin variants of Arabic names, along with their accompanying Arabic transliterations.

There are similar challenges inherent to transliterating Arabic names in the opposite direction -- from the Arabic script to the Latin script. Take, for example, the name *Ruwaida* (Arabic: رويده). The single Latin representation of this name, *Ruwaida*, can be spelled in several ways using the Arabic script. Alternatives include:

رويده  
رويذا  
رويضة

Focusing specifically on the first two variants, it becomes clear why a rule-based approach will not produce the Latin transliteration *Ruwaida*. رويده and رويذا are a minimal pair differing only by their final character (ه or ا). The sounds for both ه and ا are rendered in Latin as 'A' at the end of the name *Ruwaida*. One might therefore deduce that a rule can be generated to transform ه and ا to 'A' at the end of a word. However, mapping both ه and ا to the Latin character 'A' is not always appropriate in the word-final context. For instance, the name وجيه in Latin is *Wajee*. Mapping the final ه in وجيه to 'A' would produce *Waja*, which is not a valid transliteration for the name وجيه. To allow for this orthographical idiosyncrasy, a list must be built of all known Arabic variants of Arabic names, along with their accompanying Latin transliterations.

There is yet another orthographical complication in Arabic. Arabic is written without

short vowels. Halpern (2007) refers to the omission of short vowels as the greatest challenge to achieving accuracy in transliterating Arabic to English. In the absence of information about vowel sounds, there could be several possible transliterations of a single name written in Arabic. Take, for example, فرغل (Latin: *Farghal*). Possible transliterations of this name might include:

*Ferghal*  
*Farghal*  
*Firghul*  
*Farghel*  
*Farghil*

One must have knowledge of the lexical item فرغل to know that *Farghal* is the proper way to render فرغل using Latin characters. There are no rules that would simply insert short vowels to produce the correct Latin transliteration. To illustrate this assertion we can examine the Arabic name فردوسی, which is properly transliterated to Latin as *Firdausi*. Both فرغل (Latin: *Farghal*) and فردوسی (Latin: *Firdausi*) begin with the same two Arabic letters ف (Latin: 'F') and ر (Arabic: 'R'). Yet in فرغل we would have to insert an 'A' between these two letters, whereas in فردوسی we would have to insert an 'I' between these two letters to generate each respective Latin transliteration. By definition, no vowel insertion rule can suffice. Knowledge of each lexical item as a whole is necessary for generating the correct Latin transliteration.

The fact that Arabic is not written with short vowels also presents challenges for matching names across scripts when a rule-based approach is employed. Given the absence of vowel information from input in the Arabic script, we must ignore all vowels from input in the Latin script entirely when attempting to compare names across scripts. As a result, certain false matches occur, as seen in the following cluster of names:

Cluster:

خالد  
*Khaled*  
خلود  
*Kholoud*

This cluster results from the fact that خالد is transliterated to *Khaled*, whose vowels are then removed via rules to produce the string KHL D.

Likewise, *خلود* is transliterated to *Kholoud*, whose vowels are then removed via rules to produce the string KHL D. The two Latin input strings *Khaled* and *Kholoud* likewise have their vowels removed via rules, producing the string KHL D in both cases, and all four strings match. Of course, if we consider using placeholders for vowels we could render *Khaled* and *Kholoud* as KH\*L\*D and KH\*L\*\*D, whereby preventing these two Latin renderings from falsely matching. But since Arabic does not contain short vowels, using a placeholder character prevents us from matching Arabic with Latin. There can be no placeholder in Arabic because there are no short vowels to hold on to.

A lexical-based approach would help eliminate this problem of false matches. A list of all known Latin variants and all known Arabic variants of a single name could be mapped to a single canonical Latin representation. *خالد* and *Khaled* (along with all variants of this name in both scripts) could be mapped to *Khaled*. *خلود* and *Kholoud* (along with all variants of this name in both scripts) could be mapped to *Kholoud*. The resultant match behavior would produce these two clusters:

Cluster 1:

خالد

*Khaled*

Cluster 2:

خلود

*Kholoud*

Hence the problem of false matches can be reduced by using a comprehensive list of names and their variants. A system cannot produce these separate clusters by relying solely on a rule-based approach with a step that removes vowels.

Statistical machine translation-based approaches, such as that described in Hermjakob et. al (2008), have been successful at overcoming many of these challenges. However, the software discussed in this paper relies purely on a deterministic approach to transliteration and matching. The technologies employed in a machine-learning environment were simply not available in the QKB. The QKB is part of a generic system used to analyze and transform data in many languages across different data domains. It is not built to solve any one particular language problem, such as transliterating names between two scripts. Its components are kept simple to enable business

users to customize language processing rules to solve a variety of linguistic problems. Therefore the statistical methods required for training on a particular natural language task are not built into its architecture.

## 2 Method

This section describes the development and testing procedure of the Arabic name transliteration and matching technology, as implemented in the DataFlux Quality Knowledge Base (QKB).

### 2.1 Arabic to Latin Transliteration

A lexicon of approximately 55,000 Arabic name variants written in the Arabic script, and their accompanying Latin transliterations, was compiled using data acquired from the CJK Dictionary Institute.<sup>1</sup> In addition, an Egyptian subject matter expert manually created a lexicon of approximately 10,000 Arabic name variants written in the Arabic script along with their accompanying preferred Latin transliteration. Since the technology was implemented as part of an Egyptian Arabic software localization project, precedence was given to Egyptian conventions for spelling and spacing within Arabic names written in Latin as the standard for transliterated names. The list of preferred Egyptian transliterations was applied first, followed by the general list of transliterations acquired from the CJK Dictionary Institute. Together these two lexicons served as the primary source for transliteration. Prior to the application of the transliteration lexicons, basic cleansing operations, such as punctuation and diacritics removal, were first applied. As a fall back, rules were designed after the Buckwalter Arabic transliteration scheme<sup>2</sup> to transliterate any names that were not found in either of the two lexicons. Some additional context sensitive rules were added. For example, the *◌* character transliterates to the A character at a word boundary; elsewhere it becomes H. Three other characters that do not exist in the Buckwalter scheme ( *ﺀ*, *ﻪ*, and *ﺯ* ) were added as well because they were found in the Egyptian Arabic data that were used to test the system.

<sup>1</sup> <http://www.cjk.org/cjk/index.htm>

<sup>2</sup> <http://open.xerox.com/Services/arabic-morphology/Pages/translit-chart>

A sample of 500 full Arabic names was randomly drawn from a population of approximately 9000 full Arabic names written in the Arabic script, taken from a regional banking company's customer database. The 500 names were then transliterated to the Latin script using the QKB. The results were sent to an Egyptian subject matter expert for review. Any transliteration errors were noted in the test results, and the correct transliteration was added to the Egyptian transliteration lexicon. Transliterations were judged as errors if either the lexicon or the fallback rules rendered an unacceptable transliteration according to the subject matter expert. This regression testing process was repeated until the number of errors was deemed to be acceptable according to internal software quality standards.

Example 1: Transliteration via Egyptian transliteration scheme

طارق جعفر ابوالعينين → *Tareq Jafar AboAlEnein*

Example 2: Transliteration via CJK Dictionary Institute lexicon

كاين محرج زيتون → *Kayan Muharrij Zeitoun*

Example 3: Transliteration via PERL regular expression rules

انا نستور مالاخياس → *Ana Nstur Malakhyas*

## 2.2 Latin to Arabic Transliteration

A lexicon of approximately 863,282 Arabic name variants written in the Latin script, and their accompanying Arabic transliterations, was compiled using data acquired from the CJK Dictionary Institute. Additionally, an Egyptian subject matter expert manually created a lexicon of approximately 10,000 Arabic name variants written in the Latin script along with their accompanying preferred Arabic transliteration. As stated earlier, precedence was given to Egyptian conventions for spelling and spacing, so the list of preferred Egyptian transliterations was applied before the general CJK Dictionary Institute lexicon. Prior to the application of the transliteration lexicons, basic cleansing operations, such as punctuation and diacritics removal, were applied. As a fall back, rules were put in place after the transliteration lists. These rules performed basic letter-for-letter Latin to Arabic transliteration, with some additional context

sensitive rules provided by the Egyptian subject matter expert. For example, the Latin characters 'Y' and 'I' are transliterated to the Arabic character  $\text{ي}$  at word boundaries; elsewhere they become  $\text{ي}$ . The character 'U' is transliterated to  $\text{و}$  if it occurs after 'O'; elsewhere it becomes  $\text{ع}$ .

A sample of 500 full Arabic names was randomly drawn from a population of approximately 8000 full Arabic names written in the Latin script, taken from a regional banking company's customer database. The 500 names were then transliterated to the Arabic script using the QKB. The results were sent to an Egyptian subject matter expert for review. Any transliteration errors were noted in the test results, and the correct transliteration was added to the Egyptian transliteration lexicon. Transliterations were judged as errors if either the CJK Dictionary Institute lexicon or the fallback rules rendered an unacceptable transliteration according to the subject matter expert. This regression testing process was repeated until the number of errors was deemed to be acceptable according to internal software quality standards.

Example 1: Transliteration via Egyptian transliteration scheme

*Mohamed Samir AbdElSalam* → محمد سمير عبدالسلام

Example 2: Transliteration via CJK Dictionary Institute lexicon

*Makhtouf Nesra Abd Elwakel* → مقطوف نصراء عبدالوكيل

Example 3: Transliteration via PERL regular expression rules

*Anham Enshrah Shaghata* → انهام انشراه شآغاتة

## 2.3 Matching

Matching of Arabic names in the QKB is closely related to the Arabic to Latin Transliteration method described above. All names written in the Arabic script are transliterated to Latin in order to match the same, or similar, names across the two scripts.

Prior to applying transliteration lexicons, basic cleansing operations such as punctuation and diacritics removal are applied. As a supplementary step, Arabic name particles in both scripts (ex.

*Abdel, Al, El, Abu,* (عبد, ال, ابو) are removed from the input to reduce the input string to a basic canonical representation before final matching. Names in the Arabic script are then transliterated using a lexicon of Arabic names and their Latin counterparts. A second transliteration lexicon, consisting of names in the Arabic script stripped of their particles, is applied. For example, when عبدالرازق (Latin: *AbdelRazek*) is stripped of the particle *عبدال* (Latin: *Abdel*) in the step above, the name becomes رازق (Latin: *Razek*). The second scheme then transliterates رازق to *Razek*. For any names in the Arabic script that are not in either of the two lexicons, Arabic to Latin phonetic transliteration rules are then applied on a letter-for-letter basis. These rules are similar to the Buckwalter transliterations, but are more simplified in that there are fewer Arabic-to-Latin character mappings. That is, there are more Arabic characters that map to a single Latin character in the phonetic rules than there are in the Buckwalter transliteration scheme. This allows the system to match more names that are similar in pronunciation. After the phonetic transliteration step, all Arabic input is now successfully rendered in the Latin script, and further phonetic reductions (ex. geminate consonant reduction, vowel transformations) take place before final matching.

A sample of approximately 8000 full Arabic names was randomly drawn from a population of approximately 17,000 full Arabic names, half written in Arabic, half in Latin, taken from a regional banking company's customer database. The 8000 names were sent through a cluster analysis test using the matching technology heretofore described. The results were sent to an Egyptian subject matter expert for review. Any false matches or missed matches were noted in the test results, and either the transliteration lexicon or the phonetic transcription rules were updated to yield more accurate match results. This regression testing process was repeated until the number of errors was deemed to be acceptable according to internal software quality standards.

Examples: Clusters of similar names, identified by the matching software system.

Example 1:

فاطمه عباس عبدالرازق

*Fatma Abbas Abdel Razek*

*Fatima Abas Abdel Razik*

Example 2:

*Ahmed Malawi Abdel-Aaty*

احمد معلوى عبدالعاطى

احمد معلوى عبدالعاطى

### 3 Results

This section describes the results of the testing procedure of the Arabic name transliteration and matching technology, as implemented in the DataFlux Quality Knowledge Base (QKB).

#### 3.1 Arabic to Latin Transliteration

After twelve iterations of regression testing, the QKB transliterated Arabic names written in the Arabic script to the Latin script with an accuracy of 92%. Testing was halted after twelve iterations because an 8% error rate was deemed acceptable according to internal software quality standards. Once the accuracy reached 92%, returns on further testing iterations became diminished. Customers seeking increased transliteration accuracy for their particular data have the ability to add more names to the existing transliteration schemes. Perfect accuracy was neither necessary nor expected, and thus the product was considered ready to go to market. See above for sample transliterations.

#### 3.2 Latin to Arabic Transliteration

After fourteen iterations of regression testing, the QKB transliterated Arabic names written in the Latin script to the Arabic script with an accuracy of 93.9%. Testing was halted after fourteen iterations because a 6.1% error rate was deemed acceptable according to internal software quality standards. Once the accuracy reached 93.9%, returns on further testing iterations became diminished. Customers seeking increased transliteration accuracy for their particular data have the ability to add more names to the existing transliteration schemes. Perfect accuracy was neither necessary nor expected, and thus the product was considered ready to go to market. See above for sample transliterations.

#### 3.3 Matching

After six iterations of regression testing, the QKB matched names across the Latin and Arabic scripts with an accuracy of 99.6% with respect to false

matches. That is, 0.4% of the matches generated by the QKB were false positives. The accuracy with respect to missed matches was 99.98%; a mere .025% of the data were missed matches; i.e. false negatives. Testing was halted after six iterations because the aforementioned error rates were quite acceptable according to internal software quality standards. See above for sample clusters of similar names.

#### 4 Conclusion

Transliterating and matching Arabic names presents a challenge. Transliterating from Latin to Arabic proves difficult because there are so many Latin variants of a single Arabic name. This variety cannot be readily captured using rules, so a lexicon of Latin to Arabic transliterations must supplement such rules. Transliterating from Arabic to Latin is likewise a challenge for this very same reason. The variety of known Latin transliterations for a single Arabic name means no single transliteration is canonically correct. A list of *preferred* Latin transliterations for the Arabic-speaking country or region in question determines the correct transliteration. Rules schemes such as the Buckwalter Arabic transliteration scheme cannot capture regional orthographic conventions. Finally, the absence of short vowels in the Arabic script means there can be several possible Latin transliterations of a single Arabic name if rules are used. The absence of short vowels in Arabic also accounts for the insufficiency of using rules to match names across scripts. Without vowel information in the Arabic script, we must remove all vowels from the Latin script, and certain false matches occur. The use of a comprehensive lexicon to map all Latin and Arabic variants to a single Latin representation would help solve this problem.

The hybrid approach to transliterating and matching Arabic names, as implemented in the DataFlux Quality Knowledge Base (QKB), performed well in transliterating names across scripts. It should be noted that this paper is reporting on research in progress, as the QKB is continually undergoing updates. As the transliteration lexicons are grown over time, transliteration accuracy will improve. Likewise, any additional contextual rules that may be added to the PERL regular expression rules, and/or the

phonetic transliteration rules, will likewise contribute to better transliteration accuracy in both directions. The match results were excellent, most likely due to the significant phonetic reductions, including vowel transformations, which take place after transliteration. On the other hand, we permitted a high tolerance for false positives when evaluating the test results. At the time of development of the QKB's name matching technology, the CJK Dictionary Institute lexicons were not available. In the future, matching will rely less on rules and will leverage the CJK Dictionary Institute lexicons to produce fewer false positives. Further research will involve testing the QKB on more comprehensive data from various sources, followed by subsequent improvements and updates to handle the varying conventions for data formats across different Arabic-speaking regions.

#### References

- Jack Halpern. 2007. The Challenges and Pitfalls of Arabic Romanization and Arabization. In *Proceedings of the Second Workshop on Computational Approaches to Arabic Script-based Languages*. Palo Alta, CA.
- U. Hermjakob, K. Knight, and H. Daumé III. 2008. Name Translation in Statistical Machine Translation - Learning when to Transliterate. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 389–397, Columbus, Ohio, June.