# Tunisian Arabic Chat Alphabet Transliteration Using Probabilistic Finite State Transducers

Nadia Karmani, Hsan Soussou, and Adel Alimi
Research Groups on Intelligent Machines, University of Sfax, Tunisia

**Abstract:** *Internet is taking more and more scale in Tunisians life, especially after the revolution in 2011. Indeed, Tunisian Internet users are increasingly using social networks, blogs, etc. In this case, they favor Tunisian Arabic chat alphabet, which is a Latin-scripted Tunisian Arabic language. However, few tools were developed for Tunisian Arabic processing in this context. In this paper, we suggest developing a Tunisian Arabic chat alphabet-Tunisian Arabic transliteration machine based on weighted finite state transducers and using a Tunisian Arabic lexicon: aebWordNet (i.e., aeb is the ISO 639-3 code of Tunisian Arabic) and a Tunisian Arabic morphological analyzer. Weighted finite state transducers allow us to follow Tunisian Internet user's transcription behavior when writing Tunisian Arabic chat alphabet texts. This last has not a standard format but respects a regular relation. Moreover, it uses aebWordNet and a Tunisian Arabic morphological analyzer to validate the generated transliterations. Our approach attempts good results compared with existing Arabic chat alphabet-Arabic transliteration tools such as EiKtub.*

**Keywords:** *Tunisian arabic chat alphabet, tunisian arabic, transliteration, aebWordNet, tunisian arabic morphological analyzer, weighted finite state transducer.*

*Received August 6, 2015; accepted April 17, 2016*

## 1. Introduction

Machine transliteration is a useful component for many multilingual applications such as information retrieval, question-answering, chat application, Internet monitoring, automatic translation, named entity recognition, etc. Many transliteration tools have been developed. Generally, they convert a word from Latin script to the native word script. These tools are used for many languages such as Hindi [14], Persian [8], Arabic [2, 3, 5, 13, 24], etc., However, few contributions are made for Tunisian Arabic Chat Alphabet transliteration (TACA) [17] i.e., a transliteration using Latin script to express Tunisian Arabic script.

Indeed, with the Tunisian political revolution, Tunisian Arabic (TA) processing is taking more and more scale. Particularly, TACA transliteration becomes very important seen its increasing use by Internet users instead of TA.

In this context, we face 4 main challenges: script specifications, missing sounds, transliteration variants and language of origin [7]. Firstly, TACA and TA have different scripts illustrated in Table 1. TACA uses Latin script (i.e., with separate characters) written from Left To Right (LTR). However, TA uses Arabic script (i.e., with intermediate characters like the character 'ڤ'/q'/[1] written as 'ڤـ' in the middle of the word) written from (RTL).

[1]Phonetic according to the International Phonetic Alphabet (IPA).

Table 1. Examples of TACA-TA graphemes alignments.

| Source language: TACA | Target language: TA | Graphemes alignment |
|---|---|---|
| tounes | *Tunisia*<br>تونِس /tu:nis/ | ت و ن ِ س<br>t ou n e s |
| Thawra | *Revolution*<br>ثَوره /θawra/ | ث َ و ر ه<br>Th a w r a |
| cha3b | *People*<br>شَعب /ʃaʔˤb/ | ش َ ع ب<br>ch a 3 b |

Secondly, some sounds are missing from TACA to TA e.g., the sound of the character 'x' /ks/, and from TA to TACA e.g., the sound of the character 'ض' /dˤ/. Thirdly, TACA-TA transliteration allows multiple variants of a source term to be valid based on the opinion of different human transliterators e.g. the TA transliterations *I* 'آنا'/ʔa:na:/ and 'أنا'/ʔana:/ are valid for the TACA word 'ana' respectively according to a TA native speaker from Tunis i.e., the capital of Tunisia and a TA native speaker from Gafsa i.e., a city in Tunisia. Finally, more than one TA character can be chosen to represent the origin of the word e.g. for the TACA word 'Ali' one could choose 'ع'/ʔˤ/ for the character 'A' to specify that the word is originally Arabic rather than the most common Arabic character 'أ'/ʔ/.

In this paper, we suggest a TACA machine transliteration based on probabilistic Weighted Finite-State Transducers (WFSTs) for automatic transliterations generation and calling aebWordNet and a TA morphological analyzer for transliterations validation. Our proposed machine transliteration adopts a hybrid transliteration approach i.e., using both spelling and phonetics [7]. We evaluated it and compared it with EiKtub using a TACA-TA testing corpus.

We decompose this paper in 6 main sections: introduction, related works, Tunisian Arabic chat alphabet, the proposed TACA machine transliteration, experimental results and conclusion.

## 2. Related Works

Transliteration has been subject to many works for many languages especially for the Arabic language. We notice the works of Arbabi *et al*. [3], stalls and Knight [23], Al-Onaizan and Knight [2], Hassan and Sorensen [5], Kashani [13], etc., the first work suggested using a hybrid algorithm based on neuronal networks and knowledge based system for named entity. The second one proposed a generative model based on pronunciation. The third one improved the last work by incorporating web counts to re-score the transliteration candidate. The fourth work used a probabilistic block based transliteration. However, the fifth work adopted hidden Markov models.

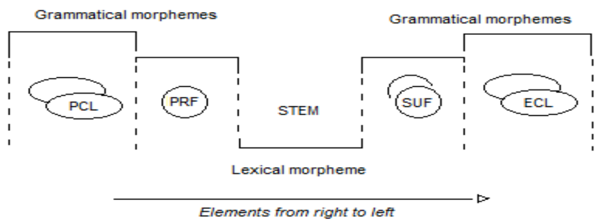
These efforts have converged to some free Arabic chat alphabet transliteration tools such as Yoolki, Yamli, Microsoft Marren, Google translator IME and EiKtub[2] [20]. The last tool is the most accurate for TA transliteration. Indeed, EiKtub adopts a phonetic one to one rule based approach that uses Bikdash Arabic transliteration rules[3], supports full vowelization and takes in charge some marginal TA consonants.

However, Tunisian Arabic is less fortunate in natural language research work and particularly in transliteration. We identified only the work in progress of Masmoudi *et al*. [17] that adopts a semi-automatic rule-based approach.

## 3. Tunisian Arabic Chat Alphabet

Many textual Internet communications are written with TACA. It is a transliteration of TA (i.e., an Arabic dialect) using Latin alphabet instead of TA alphabet based on phonemic (e.g. the character 'أ' and 'a') or graphic similarities (e.g. the character 'ق' and '9'). It does not depend on predefined rules e.g. the word *Revolutions* 'thawrat' /θawra:t/ uses the Latin morpheme 'a' to replace 'َ' then to replace 'ا'. Mainly, it is based on users practice.

TACA is a transliteration of a variant of Arabic language: TA. In fact, TA and Arabic have similar properties in transcription, lexicon and morphology. Their transcription uses Arabic script, is RTL written, is based on the Arabic consonant alphabet composed of 28 consonants and formulates vowels using Arabic diacritics. Their lexicon is composed of derived words, fixed words and exceptional words. Also, their morphology is marked by graphical words (i.e., a sequence of morphemes) which can be a unique entity or a composite unity (i.e., composed of a stem surrounded by particles such as proclitics, a prefix, suffixes and enclitics) illustrated by Figure 1.




*PCL: proclitic; PRF: prefix; SUF: suffix; ECL: enclitic*

Figure 1. The structure of graphical TA word.

However, TA varied deeply from Arabic even in transcription, lexicon and morphology. The transcription of TA uses an extended Arabic consonant alphabet composed of 31 consonants (i.e., the 28 Arabic consonants extended by three marginal TA consonants: 'ڤ' /v/, 'ڨ' /q'/ and 'پ' /P/) and generally formulates vowels using a limited set of Arabic diacritics (i.e. 6 diacritics from 9 Arabic diacritics: 'َ' /a/, 'ُ' /u/, 'ِ' /i/, 'ّ' //, 'ْ' // and 'ء' /ʔ/). In addition, its lexicon is full of exceptional words, particularly borrowed words but Arabic lexicon is rich of derived words. And its morphology is marked by TA morphemes (i.e., stem and particles) e.g. in the negation form, TA uses the enclitic 'ش' /ʃ/ at the end of the word such as *He doesn't abandon* 'مَسيلمش' /majsallamʃ/ but in Arabic, the enclitic 'ش' /ʃ/ is not used. When there is a negation, the Arabic word is preceded by 'لا' /la:/ e.g., *He doesn't abandon* 'لا يستسلم'/la: jastaslimu/.

Seen that TACA is a transcription of TA, it shares TA language's specificities but it differs in transcription. TACA uses Latin script and is LTR written e.g. *he doesn't abandon* 'mysallamch' /majsallamʃ/. It hasn't a standard alphabet (i.e., its alphabet counts Latin consonants, vowels, numbers and even symbols e.g., *People* 'cha3b', *Work* '5édma', *Loaf* 'KHob'za'). Internet users define its alphabet. In this case, we suggest building a TACA machine transliteration to define TACA alphabet and TACA-TA transliteration rules, and to generate possible TA word(s) for an inputted TACA word.

## 4. The Proposed TACA Machine Transliteration

Commonly, Machine transliteration is composed of two main parts: training and transliteration. For the proposed machine transliteration, we suggest a training part based on a manual statistical study realized by two TA native speakers and a transliteration part realized automatically using WFSTs. The last part calls aebWordNet and a TA morphological analyzer as it is shown in Figure 2.

[2]http://eiktub.com/

[3]https://en.wikipedia.org/wiki/Bikdash_Arabic_Transliteration_Rules

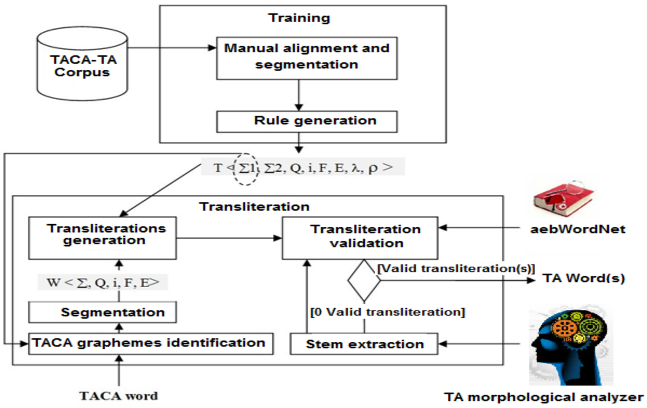


Figure 2. The proposed TACA machine transliteration structure.

## 4.1. Training

The training is based on TACA-TA training corpus detailed in section 5.1. From this corpus, we manually align graphemes and generate transliteration rules formulated as Finite State Automata (FSA).

### 4.1.1. Alignment and Segmentation

This step consists on the alignment of TACA and TA graphemes for every transliteration pair in the training corpus according to script specifications followed by the segmentation.

Generally, graphemes alignment is done automatically using existing word alignment tools like GIZA++ [21], Berkely aligner [16], Natura alignment tools[4], etc. The most used one for transliteration pairs alignment is GIZA++. However, GIZA++ considers the word in lower case, does not allow multiple to one alignment and is typically quite low for low resource language pairs [22].

Seen that TACA uses differently lower and upper case, TACA-TA transliteration contains multiple to one alignment and our training corpus is limited to 500-word pairs, we suggest manual alignment by two TA native speakers to get an efficient training model.

Indeed, a TACA word is aligned with the inversion of a TA word seen that TACA is LTR but, TA is RTL e.g., see Table 1. After alignment, the equivalent TACA-TA graphemes are manually extracted according to their position in the word, phonetic similarity e.g., ‘a’ is equivalent to ‘أ’ and graphic similarity e.g. ‘9’ is equivalent to ‘ق’. Table 2 summarizes TACA-TA equivalent graphemes extracted from the training corpus.

The alignment according to graphemes instead of characters allows us to avoid the problem of missing sounds. Indeed, missing sounds are obtained by the combination of more than one character e.g. the character ‘x’ /ks/ is transliterated as ‘كس’ /ks/ and the character ‘th’ /th/ is transliterated as ‘ض’/dˤ/.

[4]http://corpora.di.uminho.pt/natools/

Table 2. TACA-TA equivalent graphemes.

| | TACA | TA | | TACA | TA |
|---|---|---|---|---|---|
| Graphemes | a, e, A, E, é, è | ا /a:/ | Graphemes | 3 | ع /ʔˤ/ |
| | a, e, o, A | أ /ʔ/ | | g, 4, gh | غ /ɣ/ |
| | e, i, E, I | إ /ʔ/ | | f, F | ف /f/ |
| | a, e | آ /ʔa:/ | | k, 9 | ق /q/ |
| | b, B | ب /b/ | | c, k, q, C, ck | ك /k/ |
| | t, T | ت /t/ | | l, L | ل /l/ |
| | t | ة /t/ | | m, M | م /m/ |
| | th | ث /θ/ | | n, N | ن /n/ |
| | g, j | ج /ʒ/ | | a, h, H | ه /h/ |
| | h, H, 7 | ح /ħ/ | | u, w, U, W, ou | و /w/, و/u:/ |
| | 5, kh | خ /x/ | | e, i, y, I, W, Y | ي/j/,ي /i:/ |
| | d, D | د /d/ | | a, é | ى /a:/ |
| | dh, th | ذ /ð/ | | g | ڨ /q’/ |
| | r, R | ر /r/ | | p | پ /P/, پ /b’/ |
| | s, z, Z | ز /z/ | | v, V | ڥ /v/ |
| | s, S, Z | س /s/ | | e | ء // |
| | ch, ck, CH | ش /ʃ/ | | a, e, i, A, E, è | ◌َ /a/ |
| | s, S | ص/sˤ/ | | e, o, O, ou | ◌ُ /u/ |
| | th | ض /dˤ/ | | e, i, u, I | ◌ِ /i/ |
| | t | ط /tˤ/ | | ’ | ◌ْ // |
| | | ظ/dˤ/ | | x | كس /ks/ |

### 4.1.2. Rule Generation

With the absence of standard in the case of TACA-TA transliteration, we consider TACA-TA equivalent graphemes identified in the section above as transliteration rules. That allows us to cover frequent transliteration variants and the language of origin. We suggest formulating transliteration rules with a Finite State Transducer (FST). A FST is a FSA whose state transitions are labelled with both input and output symbols. Therefore, a path through the transducer encodes a mapping from an input symbol sequence to an output symbol sequence [19].

We define TACA transliteration FST T as T <∑1, ∑2, Q, i, F, E >

∑1={a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, H, I, L, M, N, O, R, S, T, U, V, W, Y, Z, 3, 4, 5, 7, 9, é, è, ’, ch, ck, dh, gh, kh, th, ou,CH}

∑2 = {ء, ا, أ, إ, آ, ب, پ, ت, ة, ث, ج, ح, خ, د, ذ, ر, ز, س, ش, ص, ض, ط, ع, غ, ف, ڥ, ق, ڨ, ك, ل, م, ن, ه, و, ي, ى, ◌َ, ◌ُ, ◌ِ, ◌ْ, كس}

Q= {0}

i=0

F= {0}

**E**={(0,a:ا,0), (0,a:أ,0), (0,a:آ,0), (0,a:ى,0), (0,a:ه,0), (0,a: ◌َ,0), (0, b:ب,0), (0,c:ك,0), (0,d:د,0), (0,e:ء,0), (0,e:ا,0), (0,e:أ,0), (0,e:إ,0), (0,e:آ,0), (0,e:ي,0), (0,e: ◌َ,0), (0,e:◌ُ,0), (0,e:◌ِ,0), (0,f:ف,0), (0,g:ج,0), (0,g:غ,0), (0,g:ڨ,0), (0,h:ح,0), (0,h:ه,0), (0,i:إ,0), (0,i:ي,0), (0,i:◌َ,0), (0,i:◌ِ,0), (0,j:ج,0), (0,k:ق,0), (0,k:ك,0), (0,l:ل,0), (0,m:م,0), (0,n:ن,0), 0,o:أ,0), (0,o:◌ُ,0), (0,p:پ,0), (0,q:ك,0), (0,r:ر,0), (0,s:ز,0), (0,s:س,0), (0,s:ص,0), (0,t:ت,0), (0,t:ة,0), (0,t:ط,0), (0,u:و,0), (0,u:◌ِ,0), (0,v:ڥ,0), (0,w:و,0), (0,x:كس,0), (0,y:ي,0), (0,z:ز,0), (0,A:أ,0), (0,A:ا,0), (0,A:◌َ,0), (0,B:ب,0), (0,C:ك,0), (0,D:د,0), (0,E:إ,0), (0,E:ا,0), (0,E:◌َ,0), (0,F:ف,0), (0,H:ح,0), (0,H:ه,0), (0,I:إ,0), (0,I:ي,0), (0,I:◌ِ,0), (0,L:ل,0), (0,M:م,0), (0,N:ن,0), (0,O:◌ُ,0), (0,R:ر,0), (0,S:س,0), (0,S:ص,0), (0,T:ت,0), (0,U:و,0), (0,V:ڥ,0), (0,W:و,0), (0,W:ي,0), (0,Y:ي,0), (0,Z:ز,0), (0,Z:س,0), (0,3:ع,0), (0,4:غ,0), (0,5:خ,0), (0,7:ح,0), (0,9:ق,0), (0,é:ا,0), (0,é:ى,0), (0,è:ا,0), (0,è:◌َ,0), (0,’:◌ْ,0), (0,ch:ش,0), (0,ck:ش,0), (0,ck:ك,0), (0,dh:ذ,0), (0,gh:غ,0), (0,kh:خ,0), (0,th:ث,0), (0,th:ذ,0), (0,th:ض,0), (0,ou:و,0), (0,ou:◌ُ,0), (0,CH:ش,0)}

In fact, Σ1 is the set of input alphabet (i.e., TACA graphemes in Table 1), Σ2 is the set of output alphabet (i.e., TA graphemes in Table 1), Q is the set of states, i is the initial state, F is the set of final states and E is the transitions set.

We notice that T is not a deterministic finite state i.e., for the same state and the same input TACA grapheme there is many outputs in T, e.g., the state 0 and the input grapheme 'a' has many output graphemes as 'ا', 'أ','آ','ى' and '◌َ'. So, using T, every TACA word may have more than one TA transcription. For every word $w=l_1 l_2 l_3 \dots l_n$, we find P possible TA transliterations as detailed in Equation (1) (i.e., $a_i$ is the number of transitions for $l_i$).

$$P=\prod_{i=1}^{n} a_i \quad (1)$$

Consequently, the transliteration process is ambiguous.

To avoid the ambiguity, we suggest using statistics to weight T. This allows us to distinguish transitions with the same state and input graphemes using weights. In this case, we propose to take the training corpus as a sample. From this sample, we count frequency f (i.e., number of occurrences) of every transition $e_i$ in E and calculate relative frequency Rf (i.e., empirical probability) for every element l in Σ1. Relative frequency is detailed in Equation (2).

$$\text{For } l,\ Rf(l, e_i)= f(l,e_i)/\Sigma_i f(l, e_i). \quad (2)$$

E.g., for l='a', we count frequency in Table 3 and we calculate relative frequency in Table 4.

Table 3. Frequency table for the graphem 'a' in the training corpus.

| Association | Number of instances |
|---|---|
| (a,ا) | 168 |
| (a,أ) | 12 |
| (a,آ) | 1 |
| (a,ى) | 6 |
| (a,ه) | 2 |
| (a,◌َ) | 204 |
| Total | 393 |

Table 4. Relative frequency table for the graphem 'a' in the training corpus.

| Association | Relative frequency |
|---|---|
| (a,ا) | 0.427 |
| (a,أ) | 0.030 |
| (a,آ) | 0.002 |
| (a,ى) | 0.015 |
| (a,ه) | 0.005 |
| (a,◌َ) | 0.519 |
| Total | 1 |

The statistical study of T allows us to define a WFST for TACA transliteration over the probability semiring (ℜ+, +, ×, 0, 1). A WFST puts weights on transitions in addition to the input and output symbols. Weights may encode probabilities, durations, penalties or any other quantity that accumulates along paths to compute the overall weight of mapping an input sequence to an output sequence [18]. In our work, we use relative frequencies as weights for T transitions. Consequently, we get the probabilistic weighted finite state T as T <Σ1, Σ2, Q, i, F, E, λ, ρ>

∑1={a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, H, I, L, M, N, O, R, S, T, U, V, W, Y, Z, 3, 4, 5, 7, 9, é, è, ', ch, ck, dh, gh, kh, th, ou, CH}

∑2 = {ء,ا , أ, إ, آ, ب, پ, ت, ة, ث, ج, ح, خ, د, ذ, ر, ز, س, ش, ص, ض, ط, ع, غ, ف, ڥ, ق, ڨ, ك, ل, م, ن, ه, و, ي, ى, ◌َ, ◌ُ, ◌ِ, ◌ْ, كس}

Q= {0}

i=0

F= {0}

E={(0, a:ا, 0.427, 0), (0, a:أ, 0.030, 0), (0, a:آ, 0.002, 0), (0, a:ى, 0.015, 0), (0, a:ه, 0.005, 0), (0, a: ◌َ, 0.519, 0), (0, b:ب, 1,0), (0,c:ك, 1,0), (0,d:د, 1,0), (0,e:ء, 0.006,0), (0,e:ا, 0.245, 0), (0, e:أ, 0.006, 0), (0, e:إ, 0.072, 0), (0, e:أ, 0.033, 0), (0, e:ي, 0.006, 0), (0, e: ◌َ, 0.072, 0),(0, e:◌ُ, 0.013, 0), (0, e:◌ِ, 0.543, 0), (0, f:ف, 1, 0), (0, g:ج, 0.285, 0), (0, g:غ, 0.142, 0), (0, g:ڨ, 0.571, 0), (0, h:ح, 0.157, 0), (0, h:ه, 0.842, 0), (0, i:إ, 0.028, 0), (0, i:ي, 0.706, 0), (0, i:◌َ, 0.005, 0), (0, i:◌ِ, 0.259, 0), (0, j:ج, 1, 0), (0, k:ق, 0.086, 0), (0, k:ك, 0.913, 0),(0, l:ل, 1, 0), (0, m:م, 1, 0), (0, n:ن, 1, 0), (0, o:أ, 0.038, 0), (0, o:◌ُ, 0.961, 0), (0, p:پ, 1, 0),(0, q:ك, 1, 0), (0, r:ر, 1, 0), (0, s:ز, 0.016, 0), (0, s:س, 0.516, 0), (0, s:ص, 0.467, 0), (0, t:ت, 0.864, 0), (0, t:ط, 0.135, 0), (0, u:و, 0.984, 0), (0, u:◌ِ, 0.015, 0), (0, v:ڥ, 1, 0), (0, w:و, 1, 0), (0, x:كس, 1, 0), (0, y:ي, 1, 0), (0, z:ز, 1, 0), (0, A:أ, 0.125, 0), (0, A:ا, 0.375, 0), (0, A:◌َ, 0.500, 0), (0, B:ب, 1, 0), (0, C:ك, 1, 0), (0, D:د, 1, 0), (0, E:إ, 0.333, 0), (0, E:ا, 0.500, 0), (0, E:◌ِ, 0.166, 0), (0, F:ف, 1, 0), (0, H:ح, 0.250, 0),(0, H:ه, 0.750, 0), (0, I:إ, 0.222, 0), (0, I:ي, 0.444, 0), (0, I:◌ِ, 0.333, 0), (0, L:ل, 1, 0),(0, M:م, 1, 0), (0, N:ن, 1, 0), (0, O:◌ُ, 1, 0),(0, R:ر, 1, 0), (0, S:س, 0.800, 0), (0, S:ص, 0.200, 0), (0, T:ت, 1, 0),(0, U:و, 1, 0), (0, V:ڥ, 1, 0), (0, W:و, 0.666, 0), (0, W:ي, 0.333, 0), (0, Y:ي, 1, 0), (0, Z:ز, 0.500, 0), (0, Z:س, 0.500, 0), (0, 3:ع, 1, 0), (0, 4:غ, 1, 0), (0, 5:خ, 1, 0), (0, 7:ح, 1, 0), (0, 9:ق, 1, 0), (0, é:ا, 0.833, 0), (0, é:ى, 0.166, 0), (0, è:ا, 0.800, 0), (0, è:◌َ, 0.200, 0), (0,':◌ْ, 1, 0), (0, ch:ش, 1, 0), (0, ck:ش, 0.900, 0), (0, ck:ك, 0.100, 0), (0, dh:ذ, 1, 0), (0, gh:غ, 1, 0), (0, kh:خ, 1, 0), (0, th:ث, 0.125, 0), (0, th:ذ, 0.625, 0), (0, th:ض, 0.250, 0), (0, ou:و, 0.500, 0), (0, ou:◌ُ, 0.500, 0), (0, CH:ش, 1, 0)}

## 4.2. TACA Transliteration Process

The proposed TACA machine transliteration detailed in Figure 2 takes as input a TACA word and generates as output TA word(s). The inputted TACA word is processed over four main steps. The first step consists on TACA graphemes identification. The second step uses these graphemes to formulate the input word segmentation as a FSA. The third step represents transliteration generation based on FSTs. Finally, the fourth step is the validation step that calls aebWordNet and if necessary, TA morphological analyzer to identify acceptable outputted TA word(s).

### 4.2.1. TACA Graphemes Identification

In this step, we retain every graphem in Σ1 (i.e., the inputted alphabet of T) included in the TACA word. Therefore, we obtain a set of possible graphemes that will be used for segmentation e.g., in Table 5.

Table 5. Examples of TACA word graphemes identification.

| **TACA input word** | **Possible graphemes** |
|---|---|
| tounes | G={t, o, u, n, e, s, ou} |
| Thawra | G={T, h, a, w, r, Th} |
| cha3b | G={c, h, a, 3, b, ch} |

### 4.2.2. Segmentation

A TACA word can be represented as a list of graphemes. However, more than one list of graphemes is possible for the same word from the set of graphemes identified by the previous step e.g. For the TACA word *Tunisia* 'tounes', the lists L1= t→ o → u →n → e → s and L2= t→ ou → n→ e → s are acceptable.

In this case, we suggest using a FSA noted W to represent all possible graphemes succession. In fact, the TACA word is formulated as: W <Σ, Q, i, F, E> where Σ is the set of input alphabet (i.e., possible TACA graphemes in the word identified in the step above), Q is the set of states (i.e., they represent grapheme positions), i is the initial state, F is the set of final states and E is the set of transitions (i.e., graphemes succession in the word).

To define E, we use the inputted word and Σ (i.e., we note Σ= {$g_i$}). It consists on the extraction of the transitions (state1, $g_1$, state2) and (state2, $g_2$, state3) such that $g_1$ is the predecessor of $g_2$ and $g_2$ is the successor of $g_1$ in the inputted word e.g., for the TACA word *Tunisia* 'tounes', we get the FSA tounes <Σ, Q, i, F, E >, where Σ= {t, o, u, n, e, s, ou}, Q={0, 1, 2, 3, 4, 5, 6} and E= {(0,t,1), (1,ou,3), (1,o,2), (2,u,3), (3,n,4), (4,e,5), (5,s,6)}.

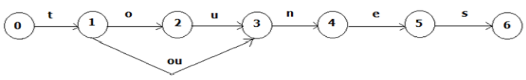



Figure 3. The FSA 'tounes'.

The FSA tounes is detailed in Figure 3 and its adjacency matrix is illustrated in Table 6.

Table 6. The adjacency matrix of 'tounes'.

| × | t | o | u | n | e | s | ou |
|---|---|---|---|---|---|---|---|
| **t** | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| **o** | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **u** | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **n** | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **e** | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **s** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ou** | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

### 4.2.3. Transliterations Generation

This step aims to generate possible transliterations for an inputted word. For this purpose, we calculate the transliteration WFST, then we optimize it using morphological rules and finally we deduce possible transliterations.

To get the transliteration WFST noted WT, we use the FSA W, considered as WFST with the same input and output that all transitions are weighted with 1, and the WFST T. Indeed, the composition i.e., W º T [18], between W and T generates WT e.g. from the FSA tounes and the WFST T, we get tounesT (see Figure 4) as tounesT <Σ1, Σ2, Q, i, F, E >, where Σ1= {t, o, u, ou, n, e, s}, Σ2= {ء, ا, أ, إ, آ, ت, ط, ن, و, ي, ◌َ, ◌ُ, ◌ِ, ز, س, ص }, Q={0, 1, 2, 3, 4, 5, 6} and E= {(0, t:ت, 0.864, 1), (0, t:ط, 0.135,1), (1, o:◌ُ, 0.961, 2), (1, o:أ, 0.038, 2), (1, ou:◌ُ, 0.500, 3), (1, ou:و, 0.500, 3), (2, u:و, 0.984, 3), (2, u:◌ِ, 0.015, 3), (3, n:ن, 1, 4), (4, e:ء, 0.006, 5), (4, e:ا, 0.245, 5), (4, e:أ, 0.006, 5), (4, e:إ, 0.072, 5), (4, e:آ, 0.033, 5), (4, e:ي, 0.006, 5), (4, e:◌َ, 0.072, 5), (4, e:◌ُ, 0.013, 5), (4, e:◌ِ, 0.543, 5), (5, s:ز, 0.016, 6), (5, s:س, 0.516, 6), (5, s :ص, 0.467, 6)}.

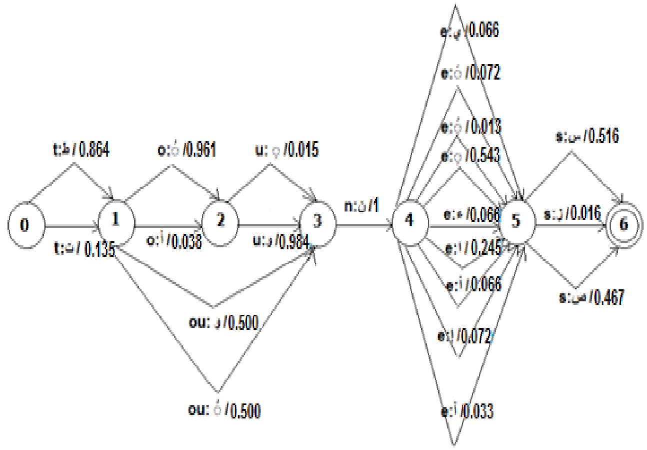



Figure 4. The WFST tounesT.

The WFST WT can be optimized based on TA morphological rules detailed in Table 7. In fact, we valid every transition in E by the verification of morphological rules. Two main optimization cases exist:

1. Identification of one wrong transition.
2. Identification of two wrong successive transitions.

Table 7. TA morphological rules.

| **N°** | **Morphological rule** |
|---|---|
| **1** | A TA word does not begin by a diacritic. |
| **2** | A TA word does not begin by the long vowel 'ى'. |
| **3** | A TA word may contain 'ى' or 'ة' or 'ء' only at the end. |
| **4** | A TA consonant takes at most two diacritics: '◌ّ' or 'ء' with one of the other diacritics. |
| **5** | A TA diacritic does not succeed the long vowels 'ا' and 'ى'. |
| **6** | The diacritic '◌ّ' does not succeed another diacritic. |
| **7** | The diacritic '◌ِ' does not succeed the graphem 'أ'. |
| **8** | The diacritics '◌َ', '◌ُ' and '◌ْ' do not succeed the graphem 'إ'. |

In the first case, we delete the wrong transition from E. However, the second case is more complicated. Let a state s with I= {$i_0 \ldots i_n$} the set of its previous transitions and O= {$o_0 \ldots o_m$} the set of its following transitions.

In this case, we notice four possibilities:

1. Every transition in I cannot precede all transitions in O. Therefore, we exclude the sets I and O from E and the state s from Q.

2. Every transition in I cannot precede a transition $o_i$ in O. Therefore, we exclude the transition $o_i$ from E.
3. One transition $i_i$ in I cannot precede all transitions in O. Therefore, we exclude the transition $i_i$ from E.
4. One transition $i_i$ in I cannot precede one transition $o_i$ in O. Therefore, we add a new state s2' in Q, we replace the transition $i_i$ (s1, g, w, s2) by a new transition $i_i$' (s1, g, w, s2') in E and we add in E a copy of transitions in O excluding $o_i$ where the first state is replaced by s2'.

We suggest using these rules for the correction and simplification of WT e.g., to optimize tounesT, we apply morphological rules one by one. Only two rules i.e. rule n°4 and rule n°7 intervene. In fact, for the state 2 in tounesT every transition in I cannot precede the transition $o_i$ (1, u:ُ,2). Consequently, the set E of tounesT composed of 21 transitions becomes E' with 20 transitions. The optimization of tounesT: tounesT' is showed in Figure 5.

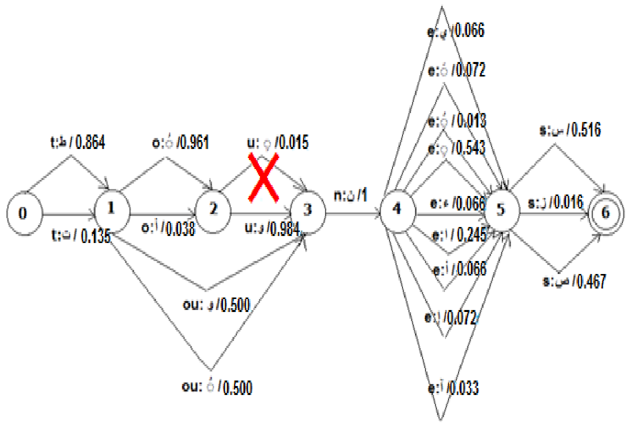



Figure 5. Morphological optimization of the WFST tounesT.

After morphological optimization, we generate transliterations by the traverse the WFST WT to search output paths $\prod = e_0 e_1 \ldots e_n$ (i.e., the concatenation of the output alphabet graphemes $e_i$) from initial state to final state representing possible transliterations and we sort them according to the path weight representing the transliteration probability. Indeed, for every path ∏ a weight $w[\prod]$ is defined by Equation (3) [18].

$$w[\prod] = w[e_0] \otimes w[e_1] \otimes \ldots \otimes w[e_n] \quad (3)$$

Seen that WT is defined over the probability semiring, $w[\prod]$ is calculated as in Equation (4).

$$w[\prod] = w[e_0] \times w[e_1] \times \ldots \times w[e_n] \quad (4)$$

E.g., for the path ∏= تونس /tunis/ (i.e., the concatenation of ت←و←ن←س), the path weight $w[\prod]$=0.135×0.961×0.984×1×0.543×0.516=0.035.

#### 4.2.4. Transliterations Validation

All transliterations generated in the step before need to be validated. In the first time, we use aebWordNet. In fact, a transliteration takes the valid state if we find it in aebWordNet Lemmas or in aebWordNet wordForms e.g., for the inputted word 'tounes' we get 'تونس' /tunis/ (P=0.035) as valid transliteration.

In this way, we can validate basically simple TA words and some graphical TA words. However, graphical TA words are not covered by aebWordNet wordForms. So, if none of the transliterations is validated, we call a TA morphological analyzer. The last one is used to extract the stems of the transliterations representing graphical TA words. The stem is validated by aebWordNet instead of the transliteration. Therefore, the validation of a transliteration stem implies the validation of the concerned transliteration.

### 4.3. The Lexicon aebWordNet

WordNet (i.e., a semantic lexicon), firstly developed for English, covers nowadays many other languages like Arabic and even dialects such as TA.

We use the standardized aebWordNet [10] according to ISO 24613 [6], that adopts an extended WordNet-LMF [23] model. This WordNet represents simple TA words as Lemmas and graphical TA words as WordForms [11]. It covers many simple TA words (i.e., verbs, nouns, adjectives and adverbs) and some graphical words like verbs in imperative tense, feminine nouns, plural nouns etc., The lexicon aebWordNet currently counts 8,279 different lemmas (i.e., 3,530 verbs, 3,010 nouns, 1,267 adjectives and 472 adverbs) and 12,152 word forms [12]. The version of aebWordNet, used for validation, covers all lexical stems of TACA testing corpus.

### 4.4. Tunisian Arabic Morphological Analyzer

Graphical TA words are considered canonically as words. However, morphologically and lexically, it is a set of lexical unities. The proposed morphological analyzer allows us to extract lexical unities and to establish lexical and grammatical labeling based on lexical characteristics of the stem, the proclitics, the prefix, the suffixes and the enclitics. This morphological analyzer is an adapted version of Arabic intelligent morphological analyzer described in [9] to TA language that uses a lexical TA dictionary, aebWordNet and an expert system. It uses a filtering approach to identify possible lexical unities combinations for an inputted TA word. Then, it calls the lexical dictionary (i.e., containing labeled TA proclitics, prefixes, suffixes and enclitics) for combinations generation and labeling i.e., every combination adopts the common lexical characteristics of its unities. Finally, the labeled combinations are submitted to an expert system that excludes wrong combinations based on labels incoherence.

# 5. Experimental Results

We evaluate our machine transliteration and EiKtub i.e., Arabic chat alphabet-Arabic transliteration tool, with standards metrics using TACA-TA testing corpus. Indeed, Seen the similarity between Arabic and TA in one hand and the common use of Latin script for the transcription of Arabic chat alphabet and TACA in the other hand, Arabic chat alphabet-Arabic transliteration tools seem accurate for TACA-TA transliteration. However, Yoolki and Yamli are available only as a Web page. Microsoft Maren and Google translator IME are available as applications, but they ignore diacritics and specific TA consonants. While EiKtub takes into consideration diacritics and two specific TA consonants i.e., 'ڨ' /q'/ and 'پ' /P/. Consequently, it is the most adapted tool for TA.

We suggest its evaluation in a TA context i.e., using TACA-TA testing corpus, to compare it with our proposed machine transliteration.

## 5.1. TACA-TA Corpus

Seen the lack of standard TACA-TA corpus, we suggest building a specialized bilingual corpus listing 1,000 different word pairs. It counts 1,000 TACA words extracted from many Internet sources i.e., forums, blogs, Facebook, etc. We transliterate these words manually by two TA native speakers to get the bilingual corpus.

This corpus is divided on a training corpus counting 500 words and testing corpus counting 500 words (see Appendix A).

We use the training corpus to identify TACA alphabet, to define TACA-TA transliteration rules and to establish a statistical study. However, the testing corpus is used for transliteration tools evaluation.

## 5.2. Evaluation Metrics

To evaluate machine transliteration, we use standard transliteration metrics: word accuracy and character accuracy [7]. The first metric, known as word accuracy, transliteration accuracy or precision A, measures the proportion of transliterations that are correct as in Equation (5).

$$A = \frac{Number-of-correct-transliterations}{Total-number-of-test-words} \quad (5)$$

The second metrics called character accuracy is based on the edit distance or Levenshtein distance between the transliterated word and the expected transliteration. The edit distance measures the number of character insertions, deletions, and substitutions that are required to transform one word into another [15]. Character accuracy CA, checks for the percentage of matched characters for each word pair as mentioned in Equation (6).

$$CA = \frac{len(W) - ED(w, L(Wi))}{len(W)} \quad (6)$$

Where, *len*(*W*) is the length of the expected target word *W* ; *L*(*Wi*) is the suggested transliteration of the system at rank i, and *ED* is the edit distance between two strings [4]. When *CA* is used to evaluate a system, an average over all the test pairs is reported.

## 5.3. Results

We implement the proposed machine transliteration using OpenFst[5][1] Then we evaluated it and we compare it to EiKtub using TACA-TA testing corpus. We get results detailed in Table 8.

Table 8. Experimental results on TACA-TA machine transliteration and EiKtub.

| Standard metrics | TACA machine transliteration | EiKtub |
|---|---|---|
| Word accuracy | 82.8% | 14.2% |
| Character accuracy | 81.99% | 79.85% |

We notice that EiKtub is not accurate for TACA-TA transliteration. Despite that EiKtub attemps 79.85 percent as character accuracy, it gets only 14.2 percent as word accuracy. While our machine transliteration gets 81.99 percent as character accuracy and 82.8 percent as word accuracy. Consequently, our machine transliteration attempts good results.

In fact, we study the TACA-TA machine transliterations excluded by the TA native speakers in word accuracy and we notice that about the half of them i.e., 48.83 percent, share the same stem or root e.g., (Table 9).

When they share the root, the TACA-TA machine transliteration form represents inflected or derived form of the manual transliteration. If we accept these transliterations, the word accuracy attempts 91.2 percent. These results are very encouraging compared with EiKtub results.

Table 9. Examples of excluded transliterations.

| TACA word | Manuel transliteration | TACA machine transliteration | Shared stem/root | form |
|---|---|---|---|---|
| sidi | سديي | سديِ | Stem سديِ | - |
| winou | ووني | وين | Stem وين | - |
| khobza | خبزه | خبز | Root خبز | Inflexion |
| 7yout | حويط | حيط | Root حوط | Inflexion |
| 9a3ed | قاعد | قعد | Root قعد | Derivation |
| jme3a | جامعه | جمع | Root جمع | Derivation |

# 6. Conclusions

The proposed TACA transliteration machine adopts a hybrid transliteration approach. It is based on probabilistic WFSTs deduced from a statistical study of Internet user transliteration practice through the training corpus. It respects TACA-TA transliteration specificities such as scripts specifications, missing sound, transliteration variant and language of origin,

[5]http://www.openfst.org

and allows us to follow Tunisian Internet user's transliteration behavior. Its evaluation, using TACA-TA testing corpus, attempts good results (i.e., word accuracy of 82.8 percent and a character accuracy of 81.99 percent) compared with EiKtub which is mainly an Arabic transliteration tool.

Our machine transliteration is very useful for TA processing as semantic analysis, clustering, information retrieval, etc which is taking more and more scale, especially after Tunisian politic revolution. In fact, TA processing tools and particularly machine transliteration are taking a main part in the Tunisian Internet monitoring in many fields such as political, economic, commercial etc. Thus, our transliteration machine can help and support the stability establishment in varied Tunisian domains.

## Acknowledgments

This work is supported by the General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

## References

[1] Allauzen C., Riley M., Schalkwyk J., Skut W., and Mohri M., "OpenFst: A General and Efficient Weighted Finite-State Transducer," *in Proceedings of the 12th International Conference on Implementation and Application of Automata*, Prague, pp. 11-23, 2007.

[2] Al-Onaizan Y. and Knight K., "Translating Named Entities using Monolingual and Bilingual Resources," *in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, pp. 400-408, 2002.

[3] Arbabi M., Fischthal S., Cheng V., and Bart E., "Algorithms for Arabic Name Transliteration," *IBM Journal of Research and Development*, vol. 38, no. 2, pp. 183-194, 1994.

[4] Hall P. and Dowling G., "Approximate String Matching," *ACM Computing Surv*eys, vol. 12, no. 4, pp. 381-402, 1980.

[5] Hassan H. and Sorensen J., "An Integrated Approach for Arabic-English Named Entity Translation," *in Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages/Association for Computational Linguistics*, Ann Arbor, pp. 87-93, 2005.

[6] ISO 24613, *Language Resource Management – Lexical Markup Framework*, ISO. Geneva, 2008.

[7] Karimi S., Scholer F., and Turpin A., "Machine Transliterations Survey," *ACM Computing Surveys*, vol. 43, no. 3, 2011.

[8] Karimi S., Scholer F., and Turpin A., "Collapsed Consonant and Vowel Models: New Approaches for English-Persian Transliteration and Back-Transliteration," *in Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics/Association for Computational Linguistics*, Czech Republic, pp. 648-655, 2007.

[9] Karmani N. and Souilem D., "Préanalyse Du Mot Arabe Basée Sur Une Approche De Filtrage Pour Une Analyse Morphologique," *in Proceedings of 16th Congrés INFormatique des ORganisations et Systèmes d'Information et de Décision/Workshop of the Arabic Information System*, Hammamet, 2006.

[10] Karmani N., Soussou H., and Alimi A., "Building a Standardized Wordnet in the ISO LMF for Tunisian Arabic Language," *in Proceedings of 7th Global Wordnet Conference*, Tartu Estonia, 2014.

[11] Karmani N., "Construction d'un Wordnet Standard Pour l'Arabe Tunisien," *in Proceedings of the 2nd Colloque Pour Les Étudiants Chercheurs en Traitement Automatique du Langage Naturel ET ses Applications*, Sousse, 2015.

[12] Karmani N., Soussou H., and Alimi A., "Tunisian Arabic aebWordNet: Current state and future extensions," *in Proceedings of the 1st International Conference on Arabic Computational Linguistics*, Cairo, pp. 3-8, 2015.

[13] Kashani M., Automatic Transliteration from Arabic to English and its Impact on Machine Translation, Theses, Simon Fraser University, 2007.

[14] Kaur V., Kaur A., and Singh J., "Hybrid Approach for Hindi to English Transliteration System for Proper Nouns," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 5, pp. 6361-6366, 2014.

[15] Levenshtein V., "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Doklady Akademi Nauk*, vol. 163, no. 4, pp. 845-848, 1965.

[16] Liang P., Taskar B., and Klein D., "Alignment by Agreement," *in Proceedings of the 5th of Human Language Technology Conference-North American Chapter of the Association for Computational Linguistics Annual Meeting*, New York, pp. 104-111, 2006.

[17] Masmoudi A., Habash N., Ellouze M., and Esteve Y., "Arabic Transliteration of Romanized Tunisian Dialect Text: Preliminary Investigation," *in Proceedings of the 16th International Conference on Intelligent Text processing and Computational Linguistics*, Cairo, pp. 608-619, 2015.

[18] Mohri M., "Weighted Finite-State Transducer Algorithms: An Overview," *Formal Languages and Applications*, Heidelberg, pp. 551-536, 2004.

[19] Mohri M., Pereira F., and Riley M., "Weighted Finite-State Transducers in Speech Recognition,"

*Computer Speech and Language*, pp. 1-26, 2001.

[20] Mostafa L., "A survey of Automated Tools for Translating Arab Chat Alphabet into Arabic Language," *American Academic and Scholarly Research Journal*, vol. 4, no. 3, 2012.

[21] Och F. and Ney H., "The Alignment Template Approach to Statistical Machine Translation," *Computational Linguistics*, vol. 30, no. 4, pp. 417-449, 2004.

[22] Pal S., Kumar Naskar S., and Bandyopadhyay S., "A Hybrid Word Alignment Model for Phrase-Based Statistical Machine Translation," *in Proceedings of the 2nd Workshop on Hybrid Approaches to Translation/ Association for Computational Linguistics*, Sofia, pp 94-101, 2013.

[23] Soria C. and Monachini M., Kyoto-LMF Wordnet Representation Format, KYOTO Working Paper, 2008.

[24] Stalls B. and knight K., "Translating Names and Technical Terms in Arabic Texts," *in Proceedings of the 17th International Conference on Computational Linguistics COLING/ACL Workshop on Computational Approach to Semitic Languages*, Montreal, pp. 34-41, 1998.

## Appendix A

Table 10. Examples from the training corpus.

| corpusN° | TACA word | Manual transliteration 1 | Manuel transliteration 2 |
|---|---|---|---|
| 1 | 5alal | خَلَل | خَلَل |
| 2 | akhaw | أكهاو | أكهو |
| 3 | ittasalt | إتَّصَلت | إتَّصَلت |
| 4 | alihom | عليهُم | عليهُم |
| 5 | aumourek | أومُورِك | أمُورِك |
| 6 | menha | مِنها | مِنها |
| 7 | orang | أرَنج | أورونج |

**Nadia Karmani** (IEEE Student Member since 2012) She is graduated in Information systems and new technologies in 2007. She obtained a PhD in Information Systems Engineering in 2017. She is a member of the research REGIM-Lab. on intelligent Machines.

**Hsan Soussou** He obtained a PhD in Information Systems Engineering in 2011.He is the manager of the electronic journal "Tunisie numérique".He is founder and manager of MDSoft society since 2005.

**Adel Alimi** (IEEE Student Member'91, Member'96, Senior Member'00). He graduated in Electrical Engineering in 1990. He obtained a PhD and then an HDR both in Electrical & Computer Engineering in 1995 and 2000 respectively. He is full Professor in Electrical Engineering at the University of Sfax, ENIS, since 2006. He is founder and director of the research REGIM-Lab. on intelligent Machines.