Automatic Transliteration of Romanized Dialectal Arabic

Mohamed Al-Badrashiny[†], **Ramy Eskander, Nizar Habash and Owen Rambow**
[†]Department of Computer Science, The George Washington University, Washington, DC
[†]badrashiny@gwu.edu
Center for Computational Learning Systems, Columbia University, NYC, NY
{reskander,habash,rambow}@ccls.columbia.edu

Abstract

In this paper, we address the problem of converting Dialectal Arabic (DA) text that is written in the Latin script (called Arabizi) into Arabic script following the CODA convention for DA orthography. The presented system uses a finite state transducer trained at the character level to generate all possible transliterations for the input Arabizi words. We then filter the generated list using a DA morphological analyzer. After that we pick the best choice for each input word using a language model. We achieve an accuracy of 69.4% on an unseen test set compared to 63.1% using a system which represents a previously proposed approach.

1 Introduction

The Arabic language is a collection of varieties: Modern Standard Arabic (MSA), which is used in formal settings and has a standard orthography, and different forms of Dialectal Arabic (DA), which are commonly used informally and with increasing presence on the web, but which do not have standard orthographies. While both MSA and DA are commonly written in the Arabic script, DA (and less so MSA) is sometimes written in the Latin script. This happens when using an Arabic keyboard is dispreferred or impossible, for example when communicating from a mobile phone that has no Arabic script support. Arabic written in the Latin script is often referred to as "Arabizi". Arabizi is not a letter-based transliteration from the Arabic script as is, for example, the Buckwalter transliteration (Buckwalter, 2004). Instead, roughly speaking, writers use sound-to-letter rules inspired by those of English[1] as well as informally established conventions to render the sounds of the DA sentence. Because the sound-to-letter rules of English are very different from those of Arabic, we obtain complex mappings between the two writing systems. This issue is compounded by the underlying problem that DA itself does not have any standard orthography in the Arabic script. Table 1 shows different plausible ways of writing an Egyptian Arabic (EGY) sentence in Arabizi and in Arabic script.

Arabizi poses a problem for natural language processing (NLP). While some tools have recently become available for processing EGY input, e.g., (Habash et al., 2012b; Habash et al., 2013; Pasha et al., 2014), they expect Arabic script input (or a Buckwalter transliteration). They cannot process Arabizi. We therefore need a tool that converts from Arabizi to Arabic script. However, the lack of standard orthography in EGY compounds the problem: what should we convert Arabizi into? Our answer to this question is to use CODA, a conventional orthography created for the purpose of supporting NLP tools (Habash et al., 2012a). The goal of CODA is to reduce the data sparseness that comes from the same word form appearing in many spontaneous orthographies in data (be it annotated or unannotated). CODA has been defined for EGY as well as Tunisian Arabic (Zribi et al., 2014), and it has been used as part of different approaches for modeling DA morphology (Habash et al., 2012b), tagging (Habash et al., 2013; Pasha et al., 2014) and spelling correction (Eskander et al., 2013; Farra et al., 2014).

This paper makes two main contributions. First, we clearly define the computational problem of transforming Arabizi to CODA. This improves over previous work by unambiguously fixing the

[1]In different parts of the Arab World, the basis for the Latin script rendering of DA may come from different languages that natively uses the Latin script, such as English or French. In this paper, we concentrate on Egyptian Arabic, which uses English as its main source of sound-to-letter rules.

Proceedings of the Eighteenth Conference on Computational Language Learning, pages 30–38,
Baltimore, Maryland USA, June 26-27 2014. ©2014 Association for Computational Linguistics

target representation for the transformation. Second, we perform experiments using different components in a transformation pipeline, and show that a combination of character-based transduction, filtering using a morphological analyzer, and using a language model outperforms other architectures, including the state-of-the-art system described in Darwish (2013). Darwish (2013) presented a conversion tool, but did not discuss conversion into a conventionalized orthography, and did not investigate different architectures. We show in this paper that our proposed architecture, which includes an EGY morphological analyzer, improves over Darwish's architecture.

This paper is structured as follows. We start out by presenting relevant linguistic facts (Section 2) and then we discuss related work. We present our approach in Section 4 and our experiments and results in Section 5.

2 Linguistic Facts

2.1 EGY Spontaneous Orthography

An orthography is a specification of how to use a particular writing system (script) to write the words of a particular language. In cases where there is no standard orthography, people use a spontaneous orthography that is based on different criteria. The main criterion is phonology: how to render a word pronunciation in the given writing system. This mainly depends on language-specific assumptions about the grapheme-to-phoneme mapping. Another criterion is to use cognates in a related language (similar language or a language variant), where two words represent a cognate if they are related etymologically and have the same meaning. Additionally, a spontaneous orthography may be affected by speech effects, which are the lengthening of specific syllables to show emphasis or other effects (such as كتيييير *ktyyyyr* [2] 'veeeery').

EGY has no standard orthography. Instead, it has a spontaneous orthography that is related to the standard orthography of Modern Standard Arabic. Table 1 shows an example of writing a sentence in EGY spontaneous orthography in different variants.

[2] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007): (in alphabetical order) *AbtθjHxdðrzsšSDTĎςγfqklmnhwy* and the additional symbols: ' ء, Â أ, Ă إ, Ā آ, ŵ ؤ, ŷ ئ, ى ý, ة ħ.

2.2 Arabizi

Arabizi is a spontaneous orthography used to write DA using the Latin script, the so-called Arabic numerals, and other symbols commonly found on various input devices such as punctuation. Arabizi is commonly used by Arabic speakers to write in social media and SMS and chat applications.

The orthography decisions made for writing in Arabizi mainly depend on a phoneme-to-grapheme mapping between the Arabic pronunciation and the Latin script. This is largely based on the phoneme-to-grapheme mapping used in English. Crucially, Arabizi is *not* a simple transliteration of Arabic, under which each Arabic letter in some orthography is replaced by a Latin letter (as is the case in the Buckwalter transliteration used widely in natural language processing but nowhere else). As a result, it is not straightforward to convert Arabizi to Arabic. We discuss some specific aspects of Arabizi.

Vowels While EGY orthography omits vocalic diacritics representing short vowels, Arabizi uses the Latin script symbols for vowels (a, e, i, o, u, y) to represent EGY's short and long vowels, making them ambiguous. In some cases, Arabizi words omit short vowels altogether as is done in Arabic orthography.

Consonants Another source of ambiguity is the use of a single Latin letter to refer to multiple Arabic phonemes. For example, the Latin letter "d" is used to represent the sounds of the Arabic letters د *d* and ض *D*. Additionally, some pairs of Arabizi letters can ambiguously map to a single Arabic letter or pairs of letters: "sh" can be use to represent ش š or سه *sh*. Arabizi also uses digits to represent some Arabic letters. For example, the digits 2, 3, 5, 6, 7 and 9 are used to represent the Hamza (glottal stop), and the sounds of the letters ع ʕ, خ *x*, ط *T*, ح *H* and ص *S*, respectively. However, when followed by "'", the digits 3, 6, 7 and 9 change their interpretations to the dotted version of the Arabic letter: غ γ, ظ *Ď*, خ *x* and ض *D*, respectively. Moreover, "'" (as well as "q") may also refer to the glottal stop.

Foreign Words Arabizi contains a large number of foreign words, that are either borrowings such as *mobile* or instances of code switching such as *I love you*.

Abbreviations Arabizi may also include some abbreviations such as *isa* which means إن شاء الله *Ăn šA' Allh* 'God willing'.

Orthography	Example
CODA	ما شفتش صحابي من امبارح *mA šftš SHAby mn AmbArH*
Non-CODA Arabic Script	ماشوفتش صوحابى من امبارح *mAšwftš SwHAbý mn AmbArH* مشفتش صحابى من إنبارح *mšftš SHAbý mn ĂnbArH* ما شفتيش صحابي من إمبارح *mA šftyš SHAby mn ĂmbArH*
Arabizi	*mashoftesh sohaby men embare7* *ma shftesh swhabi mn imbareh* *mshwftish swhaby min ambare7*

Table 1: The different spelling variants in EGY and Arabizi for writing the sentence *"I have not seen my friends since yesterday"* versus its corresponding CODA form.

2.3 CODA

CODA is a conventionalized orthography for Dialectal Arabic (Habash et al., 2012a). In CODA, every word has a single orthographic representation. CODA has five key properties (Eskander et al., 2013). First, CODA is an internally consistent and coherent convention for writing DA. Second, CODA is primarily created for computational purposes, but is easy to learn and recognize by educated Arabic speakers. Third, CODA uses the Arabic script as used for MSA, with no extra symbols from, for example, Persian or Urdu. Fourth, CODA is intended as a unified framework for writing all dialects. CODA has been defined for EGY (Habash et al., 2012a) as well as Tunisian Arabic (Zribi et al., 2014). Finally, CODA aims to maintain a level of dialectal uniqueness while using conventions based on similarities between MSA and the dialects. For a full presentation of CODA and a justification and explanation of its choices, see (Habash et al., 2012a).

CODA has been used as part of different approaches for modeling DA morphology (Habash et al., 2012b), tagging (Habash et al., 2013; Pasha et al., 2014) and spelling correction (Eskander et al., 2013; Farra et al., 2014). Converting Dialectal Arabic (written using a spontaneous Arabic orthography or Arabizi) to CODA is beneficial to NLP applications that better perform on standardized data with less sparsity (Eskander et al., 2013).

Table 1 shows the CODA form corresponding to spontaneously written Arabic.

3 Related Work

Our proposed work has some similarities to Darwish (2013). His work is divided into two sections: language identification and transliteration. He used word and sequence-level features to identify Arabizi that is mixed with English. For Arabic words, he modeled transliteration from Arabizi to Arabic script, and then applied language modeling on the transliterated text. This is similar to our proposed work in terms of transliteration and language modeling. However, Darwish (2013) does not target a conventionalized orthography, while our system targets CODA. Additionally, Darwish (2013) transliterates Arabic words only after filtering out non-Arabic words, while we transliterate the whole input Arabizi. Finally, he does not use any morphological information, while we introduce the use of a morphological analyzer to support the transliteration pipeline.

Chalabi and Gerges (2012) presented a hybrid approach for Arabizi transliteration. Their work relies on the use of character transformation rules that are either handcrafted by a linguist or automatically generated from training data. They also employ word-based and character-based language models for the final transliteration choice. Like Darwish (2013), the work done by Chalabi and Gerges (2012) is similar to ours except that it does not target a conventionalized orthography, and does not use deep morphological information, while our system does.

There are three commercial products that con-

vert Arabizi to Arabic, namely: Microsoft Maren,[3] Google Ta3reeb[4] and Yamli.[5] However, since these products are for commercial purposes, there is not enough information about their approaches. But given their output, it is clear that they do not follow a well-defined standardized orthography like we do. Furthermore, these tools are primarily intended as input method support, not full text transliteration. As a result, their users' goal is to produce Arabic script text not Arabizi text. We expect, for instance, that users of these input method support systems will use less or no code switching to English, and they may employ character sequences that help them arrive at the target Arabic script form, which otherwise they would not write if they are targeting Arabizi.

Eskander et al. (2013) introduced a system to convert spontaneous EGY to CODA, called CODAFY. The difference between CODAFY and our proposed system is that CODAFY works on spontaneous text written in Arabic script, while our system works on Arabizi, which involves a higher degree of ambiguity. However, we use CODAFY as a black-box module in our preprocessing.

Additionally, there is some work on converting from dialectal Arabic to MSA, which is similar to our work in terms of processing a dialectal input. However, our final output is in EGY and not MSA. Shaalan et al. (2007) introduced a rule-based approach to convert EGY to MSA. Al-Gaphari and Al-Yadoumi (2010) also used a rule-based method to transform from Sanaani dialect to MSA. Sawaf (2010), Salloum and Habash (2011) and Salloum and Habash (2013) used morphological analysis and morphosyntactic transformation rules for processing EGY and Levantine Arabic.

There has been some work on machine transliteration by Knight and Graehl (1997). Al-Onaizan and Knight (2002) introduced an approach for machine transliteration of Arabic names. Freeman et al. (2006) also introduced a system for name matching between English and Arabic, which Habash (2008) employed as part of generating English transliterations from Arabic words in the context of machine translation. This work is similar to ours in terms of text transliteration. However, our work is not restricted to names.

[3]http://www.getmaren.com

[4]http://www.google.com/ta3reeb

[5]http://www.yamli.com/

4 Approach

4.1 Defining the Task

Our task is as follows: for each Arabizi word in the input, we choose the Arabic script word which is the correct CODA spelling of the input word and which carries the intended meaning (as determined in the context of the entire available text).

We do not merge two or more input words into a single Arabic script word. If CODA requires two consecutive input Arabizi words to be merged, we indicate this by attaching a plus to the end of the first word. On the other hand, if CODA requires an input Arabizi word to be broken into two or more Arabic script words, we indicate this by inserting a dash between the words. We do this to maintain the bijection between input and output words, i.e., to allow easy tracing of the Arabic script back to the Arabizi input.

4.2 Transliteration Pipeline

The proposed system in this paper is called 3ARRIB.[6] Using the context of an input Arabizi word, 3ARRIB produces the word's best Arabic script CODA transliteration. Figure 1 illustrates the different components of 3ARRIB in both the training and processing phases. We summarize the full transliteration process as follows. Each Arabizi sentence input to 3ARRIB goes through a preprocessing step of lowercasing (de-capitalization), speech effects handling, and punctuation splitting. 3ARRIB then generates a list of all possible transliterations for each word in the input sentence using a finite-state transducer that is trained on character-level alignment from Arabizi to Arabic script. We then experiment with different combinations of the following two components:

Morphological Analyzer We use CALIMA (Habash et al., 2012b), a morphological analyzer for EGY. For each input word, CALIMA provides all possible morphological analyses, including the CODA spelling for each analysis. All generated candidates are passed through CALIMA. If CALIMA has no analysis for a candidate, then that candidate gets filtered out; otherwise, the CODA spellings of the analyses from CALIMA become the new candidates in the rest of the transliteration pipeline. For some words, CALIMA may suggest multiple CODA spellings that reflect different analyses of the word.

[6]3ARRIB (pronounced /ar-rib/) means "Arabize!".

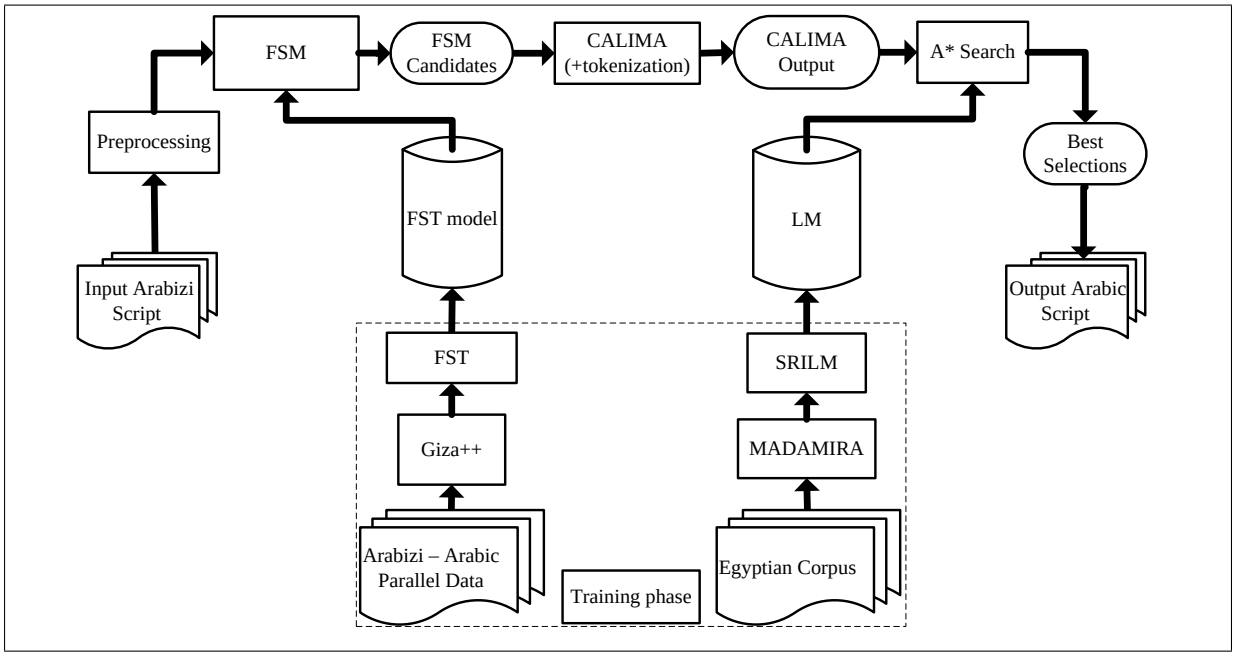

Figure 1: An illustration of the different components of the 3ARRIB system in both the training and processing phases. FST: finite-state Transducer; LM: Language Model; CALIMA: Morphological Analyzer for Dialectal Arabic; MADAMIRA: Morphological Tagger for Arabic.

Language Model We disambiguate among the possibilities for all input words (which constitute a "sausage" lattice) using an n-gram language model.

4.3 Preprocessing

We apply the following preprocessing steps to the input Arabizi text:

- We separate all attached emoticons such as (:D, :p, etc.) and punctuation from the words. We only keep the apostrophe because it is used in Arabizi to distinguish between different sounds. 3ARRIB keeps track of any word offset change, so that it can reconstruct the same number of tokens at the end of the pipeline.
- We tag emoticons and punctuation to protect them from any change through the pipeline.
- We lowercase all letters.
- We handle speech effects by replacing any sequence of the same letter whose length is greater than two by a sequence of exactly length two; for example, *iiiii* becomes *ii*.

4.4 Character-Based Transduction

We use a parallel corpus of Arabizi-Arabic words to learn a character-based transduction model. The parallel data consists of two sources. First, we use 2,200 Arabizi-to-Arabic script pairs from the training data used by (Darwish, 2013). We manually revised the Arabic side to be CODA-compliant. Second, we use about 6,300 pairs of proper names in Arabic and English from the Buckwalter Arabic Morphological Analyzer (Buckwalter, 2004). Since proper names are typically transliterated, we expect them to be a rich source for learning transliteration mappings.

The words in the parallel data are turned into space-separated character tokens, which we align using Giza++ (Och and Ney, 2003). We then use the phrase extraction utility in the Moses statistical machine translation system (Koehn et al., 2007) to extract a *phrase table* which operates over characters. The phrase table is then used to build a finite-state transducer (FST) that maps sequences of Arabizi characters into sequences of Arabic script characters. We use the negative logarithmic conditional probabilities of the Arabizi-to-Arabic pairs in the phrase tables as costs inside the FST. We use the FST to transduce an input Arabizi word to one or more words in Arabic script, where every resulting word in Arabic script is given a probabilistic score.

As part of the preprocessing of the parallel data, we associate all Arabizi letters with their word location information (beginning, middle and ending letters). This is necessary since some Arabizi

mapping phenomena happen only at specific locations. For example, the Arabizi letter "o" is likely to be transliterated into أ *Â* in Arabic if it appears at the beginning of the word, but almost never so if it appears in the middle of the word.

For some special Arabizi cases, we directly transliterate input words to their correct Arabic form using a table, without going through the FST. For example, *isa* is mapped to إن شاء الله *Ăn šA' Allh* 'God willing'. There are currently 32 entries in this table.

4.5 Morphological Analyzer

For every word in the Arabizi input, all the candidates generated by the character-based transduction are passed through the CALIMA morphological analyzer. For every candidate, CALIMA produces a list of all the possible morphological analyses. The CODA for these analyses need not be the same. For example, if the output from the character based transducer is *Aly*, then CALIMA produces the following CODA-compliant spellings: إلى *Ălý* 'to', إلي *Ălý* 'to me' and آلي *Ālý* 'automatic' or 'my family'. All of these CODA spellings are the output of CALIMA for that particular input word. The output from CALIMA then becomes the set of final candidates of the input Arabizi in the rest of the transliteration pipeline. If a word is not recognized by CALIMA, it gets filtered out from the transliteration pipeline. However, if all the candidates of some word are not recognized by CALIMA, then we retain them all since there should be an output for every input word.

We additionally run a tokenization step that makes use of the generated CALIMA morphological analysis. The tokenization scheme we target is D3, which separates all clitics associated with the word (Habash, 2010). For every word, we keep a list of the possible tokenized and untokenized CODA-compliant pairs. We use the tokenized or untokenized forms as inputs to either a tokenized or untokenized language model, respectively, as described in the next subsection. The untokenized form is necessary to retain the surface form at the end of the transliteration process.

Standalone clitics are sometimes found in Arabizi such as *lel ragel* (which corresponds to لل+ راجل *ll+ rAjl* 'for the man'). Since CALIMA does not handle most standalone clitics, we keep a lookup table that associates them with their tokenization information.

4.6 Language Model

We then use an EGY language model that is trained on CODA-compliant text. We investigate two options: a language model that has standard CODA white-space word tokenization conventions ("untokenized"), and a language model that has a D3 tokenized form of CODA in which all clitics are separated ("tokenized"). The output of the morphological analyzer (which is the input to the LM component) is processed to match the tokenization used in the LM.

The language models are built from a large corpus of 392M EGY words.[7] The corpus is first processed using CODAFY (Eskander et al., 2013), a system for spontaneous text conventionalization into CODA. This is necessary so that our system remains CODA-compliant across the whole transliteration pipeline. Eskander et al. (2013) states that the best conventionalization results are obtained by running the MLE component of CODAFY followed by an EGY morphological tagger, MADA-ARZ (Habash et al., 2013). In the work reported here, we use the newer version of MADA-ARZ, named MADAMIRA (Pasha et al., 2014). For the tokenized language model, we run a D3 tokenization step on top of the processed text by MADAMIRA. The processed data is used to build a language model with Kneser-Ney smoothing using the SRILM toolkit (Stolcke, 2002).

We use A* search to pick the best transliteration for each word given its context. The probability of any path in the A* search space combines the FST probability of the words with the probability from the language model. Thus, for any certain path of selected Arabic possibilities $A_{0,i} = \{a_0, a_1, ...a_i\}$ given the corresponding input Arabizi sequence $W_{0,i} = \{w_0, w_1, ...w_i\}$, the transliteration probability can be defined by equation (1).

$$P(A_{0,i}|W_{0,i}) = \prod_{j=0}^{i} (P(a_j|w_j) * P(a_j|a_{j-N+1,j-1})) \quad (1)$$

Where, N is the maximum affordable n-gram length in the LM, $P(a_j|w_j)$ is the FST probability of transliterating the Arabizi word w_j into the Arabic word a_j, and $P(a_j|a_{j-N+1,j-1})$ is the LM probability of the sequence $\{a_{j-N+1}, a_{j-N+2}, ...a_j\}$.

[7] All of the resources we use are available from the Linguistic Data Consortium: www.ldc.upenn.edu.

5 Experiments and Results

5.1 Data

We use two in-house data sets for development (Dev; 502 words) and blind testing (Test; 1004 words). The data contains EGY Arabizi SMS conversations that are mapped to Arabic script in CODA by a CODA-trained EGY native speaker.

5.2 Experiments

We conducted a suite of experiments to evaluate the performance of our approach and identify optimal settings on the Dev set. The optimal result and the baseline are then applied to the blind Test set. During development, the following settings were explored:

- **INV-Selection:** The training data of the finite state transducer is used to generate the list of possibilities for each input Arabizi word. If the input word cannot be found in the FST training data, the word is kept in Arabizi.
- **FST-ONLY:** Pick the top choice from the list generated by the finite state transducer.
- **FST-CALIMA:** Pick the top choice from the list after the CALIMA filtering.
- **FST-CALIMA-Tokenized-LM-5:** Run the full pipeline of 3ARRIB with a 5-gram tokenized LM.[8]
- **FST-CALIMA-Tokenized-LM-5-MLE:** The same as FST-CALIMA-Tokenized-LM-5, but for an Arabizi word that appears in training, force its most frequently seen mapping directly instead of running the transliteration pipeline for that word.
- **FST-CALIMA-Untokenized-LM-5:** Run the full pipeline of 3ARRIB with a 5-gram untokenized LM.
- **FST-Untokenized-LM-5:** Run the full pipeline of 3ARRIB minus the CALIMA filtering with a 5-gram untokenized LM. This setup is analogous to the transliteration approach proposed by (Darwish, 2013). Thus we use it as our baseline.

Each of the above experiments is evaluated with exact match, and with Alif/Ya normalization (El Kholy and Habash, 2010; Habash, 2010).

[8] 3, 5, and 7-gram LMs have been tested. The 3 and 5-gram LMs give the same performance while the 7-gram LM is the worst.

5.3 Results

Table 2 summarizes the results on the Dev set. Our best performing setup is FST-CALIMA-Tokenized-LM-5 which has 77.5% accuracy and 79.1% accuracy with normalization. The baseline system, FST-Untokenized-LM-5, gives 74.1% accuracy and 74.9 % accuracy with normalization. This highlights the value of morphological filtering as well as sparsity-reducing tokenization.

Table 3 shows how we do (best system and best baseline) on a blind Test set. Although the accuracy drops overall, the gap between the best system and the baseline increases.

5.4 Error Analysis

We conducted two error analyses for the best performing transliteration setting on the Dev set. We first analyze in which component the Dev set errors occur. About 29% of the errors are cases where the FST does not generate the correct answer. An additional 15% of the errors happen because the correct answer is not covered by CALIMA. The language model does not include the correct answer in an additional 8% of the errors. The rest of the errors (48%) are cases where the correct answer is available in all components but does not get selected.

Motivated by the value of Arabizi transliteration for machine translation into English, we distinguish between two types of words: words that remain the same when translated into English, such as English words, proper nouns, laughs, emoticons, punctuations and digits (EN-SET) versus EGY-only words (EGY-SET). Examples of words in EN-SET are: *love you very much* (code switching), *Peter* (proper noun), *haha* (laugh), *:D* (emoticon), *!* (punctuation) and *123* (digits).

While the overall performance of our best settings is 77.5%, the accuracy of the EGY-SET by itself is 84.6% as opposed to 46.2% for EN-SET. This large difference reflects the fact that we do not target English word transliteration into Arabic script explicitly.

We now perform a second error analysis only on the errors in the EGY-SET, in which we categorize the errors by their linguistic type. About 25% of the errors are non-CODA-compliant system output, where the answer is a plausible non-CODA form, i.e., a form that may be written or read easily by a native speaker who is not aware of CODA. For example, the system generates the non-CODA

System	Exact-Matching	A/Y-normalization
INV-Selection	37.1	40.6
FST-ONLY (pick top choice)	63.1	65.1
FST-CALIMA (pick top choice)	66.1	68.9
FST-CALIMA-Tokenized-LM-5	**77.5**	**79.1**
FST-CALIMA-Tokenized-LM-5-MLE	68.7	73.5
FST-CALIMA-Untokenized-LM-5	77.3	78.9
FST-Untokenized-LM-5	74.1	74.9

Table 2: Results on the Dev set in terms of accuracy (%).

System	Exact-Matching	A/Y-normalization
FST-CALIMA-Tokenized-LM-5	**69.4**	**73.9**
FST-Untokenized-LM-5	63.1	65.4

Table 3: Results on the blind Test set in terms of accuracy (%).

form مينفعش *mynfς̌* instead of the correct CODA form ما ينفعش *mA ynfς̌* 'it doesn't work'. Ignoring the CODA-related errors increases the overall accuracy by about 3.0% to become 80.5%. The accuracy of the EGY-SET rises to 88.3% as opposed to 84.6% when considering CODA compliance.

Ambiguous Arabizi input contributes to an additional 27% of the errors, where the system assigns a plausible answer that is incorrect in context. For example, the word *matar* in the input Arabizi *fel matar* 'at the airport' has two plausible out-of-context solutions: مطار *mTAr* 'airport' (contextually correct) and مطر *mTr* 'rain' (contextually incorrect).

In about 2% of the errors, the Arabizi input contains a typo making it impossible to produce the gold reference. For example, the input Arabizi *ba7bet* contains a typo where the final *t* should turn into *k*, so that it means باحبك *bAHbk* 'I love you [2fs]'.

In the rest of the errors (about 46%), the system fails to come up with the correct answer. Instead, it assigns a completely different word or even an impossible word. For example, the correct answer for the input Arabizi *sora* 'picture' is صورة *Swrħ*, while the system produces the word سور *swr* 'wall'. Another example is the input Arabizi *talabt* 'I asked for', where the output from the system is طالبة *TAlbħ* 'student', while the correct answer is طلبت *tlbt* 'I asked for, ordered' instead.

6 Conclusion and Future Work

We presented a method for converting dialectal Arabic (specifically, EGY) written in Arabizi to Arabic script following the CODA convention for DA orthography. We achieve a 17% error reduction over our implementation of a previously published work (Darwish, 2013) on a blind test set.

In the future, we plan to improve several aspects of our models, particularly FST character mapping, the morphological analyzer coverage, and language models. We also plan to work on the problem of automatic identification of non-Arabic words. We will extend the system to work on other Arabic dialects. We also plan to make the 3ARRIB system publicly available.

Acknowledgement

This paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0014. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

G. Al-Gaphari and M. Al-Yadoumi. 2010. A method to convert Sana'ani accent to Modern Standard Arabic. *International Journal of Information Science and Management*, pages 39–49.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in arabic text. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Achraf Chalabi and Hany Gerges. 2012. Romanized Arabic Transliteration. In *Proceedings of the Second Workshop on Advances in Text Input Methods (WTIM 2012)*.

Kareem Darwish. 2013. Arabizi Detection and Conversion to Arabic. *CoRR*.

Ahmed El Kholy and Nizar Habash. 2010. Techniques for Arabic Morphological Detokenization and Orthographic Denormalization. In *Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Noura Farra, Nadi Tomeh, Alla Rozovskaya, and Nizar Habash. 2014. Generalized Character-Level Spelling Error Correction. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, Baltimore, Maryland, USA.

Andrew Freeman, Sherri Condon, and Christopher Ackerman. 2006. Cross linguistic name matching in English and Arabic. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 471–478, New York City, USA.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Mona Diab, and Owen Rabmow. 2012a. Conventional Orthography for Dialectal Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Istanbul.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012b. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Nizar Habash. 2008. Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation. In *Proceedings of ACL-08: HLT, Short Papers*, pages 57–60, Columbus, Ohio.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the European chapter of the Association for Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics*, Prague, Czech Republic.

Franz Joseph Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M. Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

Wael Salloum and Nizar Habash. 2011. Dialectal to Standard Arabic Paraphrasing to Improve Arabic-English Statistical Machine Translation. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, pages 10–21, Edinburgh, Scotland.

Wael Salloum and Nizar Habash. 2013. Dialectal Arabic to English Machine Translation: Pivoting through Modern Standard Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Hassan Sawaf. 2010. Arabic dialect handling in hybrid machine translation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, Denver, Colorado.

Khaled Shaalan, Hitham Abo Bakr, and Ibrahim Ziedan. 2007. Transferring Egyptian Colloquial into Modern Standard Arabic. In *International Conference on Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO.

Ines Zribi, Rahma Boujelbane, Abir Masmoudi, Mariem Ellouze, Lamia Belguith, and Nizar Habash. 2014. A Conventional Orthography for Tunisian Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.