Processing Spontaneous Orthography

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh
Center for Computational Learning Systems
Columbia University
{reskander,habash,rambow,nadi}@ccls.columbia.edu

Abstract

In cases in which there is no standard orthography for a language or language variant, written texts will display a variety of orthographic choices. This is problematic for natural language processing (NLP) because it creates spurious data sparseness. We study the transformation of spontaneously spelled Egyptian Arabic into a conventionalized orthography which we have previously proposed for NLP purposes. We show that a two-stage process can reduce divergences from this standard by 69%, making subsequent processing of Egyptian Arabic easier.

1 Introduction

In areas with **diglossia**, vernacular spoken variants ("low") of a language family co-exist with a largely written variant ("high"), which is often not spoken as a native language. Traditionally, the low variants have not been written: written language is reserved for formal occasions and in those formal occasions only the high variant is used. Prototypical examples of diglossia are the German speaking parts of Switzerland, and the Arab world. The advent of the internet has changed linguistic behavior: it is now common to find written informal conversations, in the form of email exchanges, text messages, Twitter exchanges, and interactions on blogs and in web forums. These written conversations are typically written in the low variants (or in a mixture of low and high), since conversations in the high variant seem unnatural to the discourse participants. For natural language processing (NLP), this poses many challenges, one of which is the fact that the low variants have not been written much in the past and do not have a standard orthography which is generally agreed on by the linguistic community (and perhaps sanctioned by an authoritative institution). Instead, each discourse participant devises a **spontaneous orthography**, in which she chooses among conventions from the high variant to render the spoken language. We are thus faced with a large number of ways to spell the same word, none of which can be assumed as "standard" since there is no standard. As a result, the increased data sparseness adds to the challenges of NLP tasks such as machine translation, compared to languages for which orthography is standardized.

In this paper, we work on Egyptian Arabic (EGY). We follow the conventions which we have previously proposed for the normalized orthography for EGY (Habash et al., 2012), called CODA (Conventional Orthography for Dialectal Arabic). In this paper, we investigate how easy it is to convert spontaneous orthography of EGY written in Arabic script into CODA orthography automatically. We will refer to this process as "normalization" or "codafication". We present a freely available system called CODAFY, which we propose as a preprocessor for NLP modules for EGY. We show that a "do nothing" baseline achieves a normalization performance of 75.5%, and CODAFY achieves a normalization performance of 92.4%, an error reduction of 69.2% over this baseline on an unseen test set.

The paper is structured as follows. We first review relevant linguistic facts in Section 2 and then present the conventionalized orthography we use in this paper. After reviewing related work in Section 4, we present our data (Section 5), our approach (Section 6), and our results (Section 7). We conclude with a discussion of future work.

2 Linguistic Facts

2.1 Writing without a Standard Orthography

An **orthography** is a specification of how the words of a language are mapped to and from a particular script (in our case, the Arabic script). In cases when a standard orthography is absent, writers make decisions about spontaneous orthography based on various criteria. Most prominent among them is **phonology**: how can my pronunciation of the word be rendered in the chosen writing system, given some (language-specific) assumptions about grapheme-to-phoneme mapping? Often, these assumptions come from the "high" variant of the language, or some related language. Another criterion for choosing orthography is a cognate in a related language or language variant (Modern Standard Arabic or MSA, the high variant for EGY), where a cognate pair is a pair of words (or morphemes) in two languages or language variants which are related by **etymology** (in some unspecified manner) and which have roughly the same meaning. Finally, the chosen spontaneous orthography can be altered to reflect **speech effects**, notably the lengthening of syllables to represent emphasis or other effects (such as كتييييير *ktyyyyr*[1] 'very').

It is important to distinguish typos from spontaneous orthography. We define **spontaneous orthography** to be an intentional choice of graphemes to render the words in a language or language variant. We define a **typographical error (typo)** to be an unintended sequence of graphemes. For example, كدا *kdA* and كده *kdh* can be intended spellings for EGY /kida/ 'like this', while كرا *krA* is not a plausible intentional spelling since it neither relates /kida/ to an MSA cognate, nor does the sequence of graphemes represent the phonology of EGY using standard assumptions. Instead, we can explain the spelling by assuming that the writer accidentally substituted the grapheme ر for the grapheme د, which look somewhat alike and are near each other on some Arabic keyboard layouts. Of course, when we encounter a specific spelling in a corpus, it can be, in certain cases, difficult to determine whether it is a conscious choice or a typo.

2.2 Relevant Differences between EGY and MSA

Lexical Variations Lexically, the number of differences is quite significant. For example, EGY مرات *mrAt* 'wife [of]' corresponds to MSA زوجة *zwjħ*. In such cases of lexical difference, no cognate spelling is available.

Phonological Variations There is an extensive literature on the phonology of Arabic dialects (Watson, 2002; Holes, 2004; Habash, 2010). Several phonological differences exist between EGY and MSA which relate to orthography. Here, we discuss one representative difference. The MSA consonant ث /θ/ is pronounced as /t/ in EGY (or /s/ in more recent borrowings from MSA). For example, MSA يكثر *ykθr* 'increase (imperfective)' is pronounced /yakθur/ in MSA versus /yiktar/ in EGY, giving rise to the EGY phonological spelling يكتر *yktr*.

Morphological Variations There are a lot of morphological differences between MSA and EGY. For orthography, two differences are most relevant. The MSA future proclitic /sa/+ (spelled +س *s*+) appears in EGY as /ha/+ or /Ha/+. The two forms appear in free variation, and we have not been able to find a variable that predicts which form is used when. This variation is not a general phonological variation between /h/ and /H/, we find it only in this morpheme. Predictably, this leads to two spellings in EGY: +ح *H*+ and +هـ *h*+. Negation in EGY is realized as the circum-clitic /mā/+ . . . +/š/. The principal orthographic question is whether the prefix is a separate word or is part of the main word; both variants are found.

3 CODA

CODA is a conventionalized orthography for Arabic dialects (Habash et al., 2012). In this section, we summarize CODA so that the reader can understand the goals of this paper. CODA has five key properties.

1. CODA is an internally consistent and coherent convention for writing DA: every word has a

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

single orthographic rendering.

2. CODA is created for computational purposes.
3. CODA uses the Arabic script as used for MSA, with no extra symbols from, for example, Persian or Urdu.
4. CODA is intended as a unified framework for writing all dialects. In this paper, we only discuss the instantiation of CODA for EGY.
5. CODA aims to maintain a level of dialectal uniqueness while using conventions based on similarities between MSA and the dialects.

We list some features of CODA relevant to this paper.

Phonological Spelling CODA generally uses phonological spelling for EGY (as MSA spelling does for MSA).

Etymologically Spelled Consonants A limited number of consonants may be spelled differently from their phonology if the following two conditions are met: (1) the consonant must be an EGY root radical and (2) the EGY root must have a cognate MSA root. If the conditions are met, then we spell the consonant using the corresponding radical from the cognate MSA root of the dialectal word's root. One such example is the spelling of the EGY verb pronounced /kitir/ as كثر *kθr* ' it increased'.

Morphologically Faithful CODA preserves dialectal morphology and spells the dialectal morphemes (clitics and inflections) phonologically. For example, for the attachable future marker clitic, the variant ح+ is chosen, not the MSA س+, so that EGY /Hatiktar/ (and its variant /hatiktar/) are both spelled حتكثر *Htkθr*. The negation prefix and indirect object pronoun (*l+pronoun*) suffixes are separated, e.g., ما قلت لهاش *mA qlt lhAš* /ma'ultilhāš/ 'I did not tell her'.

Alif-Maqsura The letter ى *ý* is often used in Egypt to write word-final ي *y* and *vice versa* (even when writing MSA). In CODA, all rules for using Alif-Maqsura are the same as MSA. For example, EGY /maSrī/ 'Egyptian' can be seen in spontaneous orthography as مصرى *mSrý*, but in CODA it is مصري *mSry*.

Ta-Marbuta As in MSA, the Ta-Marbuta (ة *ħ*) is used morphemically in CODA. The Ta-Marbuta is always written as ة *ħ* in CODA, e.g., /'arbaςa/ 'four' is أربعة *Ârbςħ* in CODA, though it can be found as أربعه *Ârbςh* (or اربعه *Arbςh*) in spontaneous orthography.

Lexical Exceptions EGY CODA guidelines include a word list specifying ad hoc spellings of EGY words that may be inconsistent with the default mapping outlined above. An example is /kida/ 'like this', which we find as both كدا *kdA* and كده *kdh* in spontaneous orthography; the CODA spelling is كده *kdh*.

4 Related Work

To our knowledge, this paper is the first to discuss the task of automatically providing a conventionalized spelling for a written Arabic dialect text. While there is no direct precedent, we discuss here some related research.

Our proposed work has some similarity to **automatic spelling correction** (ASC) and related tasks such as post editing for optical character recognition (OCR). Our task is different from ASC since ASC work assumes a standard orthography that the writer is also assumed to aim for. Both supervised and unsupervised approaches to this task have been explored. Unsupervised approaches rely on improving the fluency of the text and reducing the percentage of out-of-vocabulary words using NLP tools, resources, and heuristics, e.g., morphological analyzers, language models, and edit-distance measure, respectively (Kukich, 1992; Oflazer, 1996; Ben Othmane Zribi and Ben Ahmed, 2003; Shaalan et al., 2003; Haddad and Yaseen, 2007; Hassan et al., 2008; Shaalan et al., 2010; Alkanhal et al., 2012). Supervised approaches learn models of correction by training on paired examples of errors and their corrections. This data is hard to come by naturally, though for applications such as OCR corpora can be created from the application itself (Kolak and Resnik, 2002; Magdy and Darwish, 2006; Abuhakema et al., 2008; Habash and Roth, 2011).

There has been some work on **conversion** of dialectal Arabic to MSA. Al-Gaphari and Al-Yadoumi (2010) introduced a rule-based method to convert Sanaani dialect to MSA, and Shaalan et al. (2007) used a rule-based lexical transfer approach to transform from EGY to MSA. Similarly, both Sawaf

(2010) and Salloum and Habash (2011) showed that translating dialectal Arabic to MSA can improve dialectal Arabic machine translation into English by pivoting on MSA. A common feature across these conversion efforts is the use of morphological analysis and morphosyntactic transformation rules (for example, Al-Gaphari and Al-Yadoumi (2010)). While all this work is similar to ours in that dialectal input is processed, our output is still dialectal, while the work on conversion aims for a transformation into MSA.

The work most closely related to ours is that of Dasigi and Diab (2011). They identify the spelling variants in a given document and normalize them. However, they do not present a system that converts spontaneous spelling to a pre-existing convention such as CODA, and thus their results cannot be directly related to ours. Furthermore, their technique is different. First, similarity metrics based on string difference are used to identify if two strings are similar. Also, a contextual string similarity is used based on the fact that if two words are orthographic variants of each other, then they are bound to appear in similar contexts. After identifying the similar strings, the strings of interest are modeled in a vector space and clustered according to the similarity of their vectors.

5 Data

In this work, we use a manually annotated EGY Arabic corpus, developed by the Linguistic Data Consortium (LDC), and labeled as "ARZ" (Maamouri et al., 2012), parts 1, 2, 3, 4 and 5. The corpus consists of about 160K words (excluding numbers and punctuations), and follows the part-of-speech (POS) guidelines used by the LDC for Egyptian Arabic. The corpus contains a full analysis of Egyptian Arabic text in spontaneous orthography. The analysis includes the correct CODA orthography of the raw text, in addition to the full morphological/POS annotations.

Data Preparation We divide the ARZ corpus into three parts: training, development and test, which are of about 122K, 19K and 19K words, respectively. We only consider the orthographic information in the ARZ corpus: for every word in the corpus, we retain the spontaneous orthographic form and its CODA-compliant form.

We manually checked the CODA-compliant annotations for about 500 words in the development corpus. We found that the accuracy of the gold annotations in this subset is about 93%. We performed next an error analysis for the erroneous gold annotations. About one half of the gold errors are CODA phonological and orthographical errors. Examples of the CODA phonological errors include wrong additions and deletions of ا *A* and ي *y*, in addition to the ت/ث *t/θ* transformations, and the transformations that correspond to the different phonological forms of pronouncing the letter ه *h*. The CODA orthographical errors are those errors where a word orthography looks the same as its pronunciation, while it should not be, such as the ه/ة *h/ħ* transformations. One fifth of the gold errors are annotation typos, such as writing قطورات *qTwrAt* instead of قطارات *qTArAt* 'trains'. Moreover, 9% of the gold errors are wrong merges for the negation particle ما *mA* and the indirect object pronouns (*l+pronouns*). Since we use the gold in our study, and given the error analysis for the gold, we expect a qualitatively better gold standard to yield better results.

Transformation Statistics We observe two types of transformations when converting from spontaneous orthography to CODA: character substitutions that do not affect the word length, and character additions/deletions that change the word length. Tables 1 and 2 show the most common character substitution and addition/deletion transformations, respectively, as they appear in the training corpus, associated with their frequencies relative to the occurrence of transformations. The character substitutions are dominant, and constitute about 84% of all the transformations in the training corpus. While the classification of the character substitutions is automatically generated, the classification of the character additions/deletions is done manually using a random sample of 400 additions/deletions in the training corpus. This is because many additions/deletions are ambiguous.

6 Approach

We describe next the various approaches for spontaneous orthography codafication. Our codafication techniques fall into two main categories: contex-

Transformation	Frequency %
ا/أ/إ/آ *A/Â/Ǎ/Ā* ⇔ ا/أ/إ/آ *A/Â/Ǎ/Ā*	38.5
ي *y* ⇔ ى *ý*	29.7
ه *h* ⇔ ة *ħ*	16.9
ه *h* ⇔ ح *H*	2.5
ث *θ* ⇔ ت/س *t/s*	1.0
ا *A* ⇔ ه *h*	0.7
ق *q* ⇔ ا/أ/إ/آ/ء/ؤ/ئ *A/Â/Ǎ/Ā/'/ŵ/ŷ*	0.4
و *w* ⇔ ه *h*	0.3
ذ *ð* ⇔ د/ز *d/z*	0.3
ة *ħ* ⇔ ت *t*	0.3
ا *A* ⇔ ة *ħ*	0.2
ض *D* ⇔ د/ز/ظ *d/z/Ď*	0.2

Table 1: Spontaneous to CODA character substitution transformations

Transformation	Frequency %
Errors in closed class words	22.0
Missing space after ما/لا/يا *mA/lA/yA*	19.0
ا *A* **additions & deletions**	16.8
Gold errors	10.3
Speech effects	8.5
Missing space before ل *l* **(+pron)**	8.5
و/ه/وا *w/h/wA* ⇔ و/ه/وا *w/h/wA*	8.3
ي *y* **additions & deletions**	3.0

Table 2: Spontaneous to CODA character addition/deletion transformations

tual and non-contextual, where the non-contextual approaches are a lot faster than the contextual ones.

6.1 Speech Effect Handling

Before applying any codafication techniques, we perform a special preprocessing step for speech effects, which represent redundant repetitions of some letter in sequence. Sometimes people intend these repetitions to show affirmation or intensification. This is simply handled by removing the repetitions when a letter is repeated more than twice in a row, except for some letters whose repetitions for more than once indicates a speech effect; these letters are ا *A*, آ *Ā*, ء *'*, ئ *ŷ*, ى *ý* and ة *ħ*. Handling speech effects on its own corrects about 2% of the non-CODA spontaneous orthography to its CODA-compliant form, without introducing any new errors. In all experiments we report in this paper, we have initially processed speech effects.

6.2 Character Edit Classification (CEC)

In this approach, a set of transformations is applied on a character level, where a character may receive a change or not. As a result, a word changes if one or more of its characters is changed. The output of these transformations is what constitutes the CODA orthography. This is a surface modeling technique that does not depend on the word context (though it does depend on character context inside the word).

First, we train classifiers for the most frequent transformations from EGY spontaneous orthography to the corresponding CODA, listed in Tables 1 and 2 in Section 5. Second, we apply the trained classifiers to generate the CODA output.

Training the classifiers For each transformation listed in the data section, we train a separate classifier. The classifiers are trained on our training corpus using the k-nearest neighbor algorithm (k-NN) (Wang et al., 2000), which is a method for classifying objects based on the closest training examples in the feature space. We did experiments using the other classification methods included in the WEKA machine learning tool (Hall et al., 2009), including SVMs, Naïve Bayes, and decision trees. However, k-NN gives the best results for our problem.

In the training process, a set of nine static features is applied, which are the character that is queried for the transformation with its preceding and following two characters (a special token indicates a character position that does not exist because it is beyond the word boundary), along with the first two and the last two characters in the underlying word.

In this model, each data point is a character, where a classifier determines whether a character should receive a substitution, deletion, or addition of another character. The effect of each classifier is examined separately on the development set. We then determine for each classification whether it helps or weakens the process by comparing its effect to the baseline which is doing nothing, i.e., no change to a word occurs. Those classifiers that on their own perform worse than the baseline are eliminated. For eliminating the classifiers, we examine them in a descending order according to the frequencies of their corresponding transformations. We now discuss the seven classifiers we retain in our system:

1. The different ا *A* form (إ/أ/آ/ا *A/Â /Ǎ/Ā*) classifier. The classifier can change any ا *A* form into any other ا *A* form. The arbitrary selection of the different ا *A* forms represents the most frequent divergence from CODA in Arabic spontaneous orthography.
2. The ي/ى *y/ý* classifier. The classifier handles transformations between ي *y* and ى *ý* in both directions, as their selection is mostly arbitrary in EGY spontaneous orthography.
3. The ه/ة/و *h/ħ/w* classifier. The classifier handles transformations between ه *h*, ة *ħ* and و *w* in both directions. These transformations are likely to happen at word endings, since they represent common misspellings in writing ه *h*, ة *ħ* and و *w*, where ه *h* is often substituted for the graphically similar ة *ħ*, and ه *h* and و *w* can both be used to represent the 3rd person masculine singular accusative or genitive clitic. (Note that in Table 1, we list transformations between ه *h* and ة *ħ* as well as between ه *h* and و *w*; the remaining transformations are not frequent.)
4. The ه/ح *h/H* classifier. The transformation from ه *h* to ح *H* is likely to happen at word beginnings, since it represents a common deviation in writing the ح *H* future particle.
5. The ا *A* deletion classifier. The classifier handles the deletion of extra ا *A* at some positions, which is a common deviation in EGY spontaneous orthography, where the CODA orthography requires only short vowels instead.
6. The ا *A* addition classifier. The classifier handles the addition of ا *A* in some positions, where it is mostly omitted in EGY spontaneous orthography, such as adding ا *A* after the ح *H* future particle and the ب *b* progressive particle (when used with the 1st person singular imperfective), as well as the و *w* plural pronoun at word endings. When training this classifier, we target the letter after which the ا *A* should be added.
7. The space addition classifier. The classifier handles the addition of spaces in the middle of words, i.e., splitting a word into two words. This is required to add spaces after ما/لا/يا *mA/lA/yA* for negation and vocation, and before the indirect object *l+pronoun*, so that the text becomes CODA-compliant.

Generating CODA **Orthography** Next, we apply the trained classifiers on the spontaneous-orthography text. Each classifier determines a set of character corrections, where the characters may receive transformations corresponding to those on which the classifier is trained. The classifiers are independent of one another, so their order of application is irrelevant.

By way of example, we apply the classifiers on the word اربعه *Arbςh*, 'four'. The first classifier, corresponding to the different ا *A* forms, determines the transformation of ا *A* to أ *Â*, while the ه/ة *h/ħ* classifier determines the correction of ه *h* to ة *ħ*. The other classifiers are either not involved since they do not work on any of the word characters, or they determine that no character transformation should happen for this word. Thus applying the CEC technique in this case changes the word اربعه *Arbςh* to أربعة *Ârbςħ*, which is the correct CODA form.

6.3 Maximum Likelihood Estimate (MLE)

Another surface modeling approach for spontaneous orthography codafication is to use a maximum likelihood model that operates on the word level. In this approach, we build a unigram model that replaces every word in the spontaneous orthography with its most likely CODA form as seen in the training data. This assumes that the underlying word exists in the training corpus. For unseen words, the technique keeps them with no change.

The MLE approach chooses the correct CODA form for most of the words seen in training, making this approach highly dependent on the training data. It is efficient at correcting common misspellings in frequent words, especially those that are from closed classes.

6.4 Morphological Tagger

In addition to the approaches discussed above, we use a morphological tagger, MADA$_{ARZ}$ (Morphological Analysis and Disambiguation for Egyptian Arabic) (Habash et al., 2013). Although MADA$_{ARZ}$ is originally developed to work as a morphological tagger, it still can help the codafication process, since the choice of a full morphological analysis for a word in context determines its CODA spelling. Therefore, MADA$_{ARZ}$ is able to correct many word misspellings that are common in spontaneous orthography. These corrections include (آ/إ/أ/ا *A/Â/Ǎ/Ā*), ى/ي *y/ý* and ة/ه *h/ħ* transformations. However, MADA$_{ARZ}$, as a codafication technique, uses the context of the word, which makes it a contextual modeling approach unlike CEC and MLE. It is much slower than they are.

6.5 Combined Techniques

The CEC and MLE techniques can be applied alone, or they can be applied together in a pipeline in either order. This gives a total of four possible combinations. Next, we conducted experiments with MADA$_{ARZ}$, running alone and as a pre- or postprocessor for a combination of CEC and/or MLE. In all cases, when we first apply one module and then another on the output of the first, we train the second module on the training corpus which has been passed through the first module. The results of running the different codafication approaches are discussed next.

7 Evaluation

7.1 Accuracy Evaluation

The different codafication approaches, discussed in the previous section, are tested against the development set, which was not used as part of our training. The evaluation metric we use is a word accuracy metric, i.e., we evaluate how well we can correctly predict the CODA form of the input spontaneous orthography.

Table 3 lists the effects of using the different codafication approaches. For each approach, two numbers are reported; exact and normalized. In the exact evaluation, the output of the codafication approach is exactly matched against the correct CODA orthography, while in the normalized evaluation, the match is relaxed for the (آ/إ/أ/ا *A/Â/Ǎ/Ā*) and ى/ي *y/ý* alternations, i.e., these differences do not count as errors. In many NLP applications (such as machine translation), the input is normalized for these two phenomena, so that the normalized evaluation gives a sense of the relevance of codafication to downstream processes which normalize.

In this evaluation we compare our different codafication techniques, CEC, MLE, CEC+MLE and MLE+CEC, against the baseline. We also show the effect of using MADA$_{ARZ}$ as a codafication system. We see that MLE on its own outperforms CEC. Running CEC first and then MLE gives us our best result using surface techniques, namely 91.5%, for an error reduction of 63.4% against the baseline. This configuration also gives the highest normalized accuracy of 95.2%, for an error reduction of 49.5% against the baseline.

We now turn to deep modeling techniques. The performance of MADA$_{ARZ}$ on its own as a codafication system is close to the performance of CEC+MLE, by which it is outperformed in the exact-match accuracy by 0.4%.

The best deep modeling (and the best overall) performance is achieved when running MADA$_{ARZ}$ on top of MLE. This gives the highest accuracy of 92.6% (exact) and 95.8% (normalized), for error reductions of 68.1% (exact) and 55.8% (normalized) against the baseline, respectively. Note that the non-contextual modeling techniques CEC (5,584 words/sec) and MLE (6,698 words/sec) are a lot faster than the deep modeling technique MADA$_{ARZ}$ (53 words/sec), while their combination CEC+MLE+MADA$_{ARZ}$ is the slowest among all the approaches, operating at a rate of 52 words/sec. Thus, a small drop in accuracy results in a large increase in speed.

We also evaluated using MADA$_{MSA}$ (v 3.2) (Morphological Analysis and Disambiguation for MSA) (Habash and Rambow, 2005; Habash et al., 2010). MADA$_{MSA}$ is able to do some codafication, but it performs far worse than our codafication approaches.

Table 4 lists the results of the best performing codafication surface approach, CEC+MLE, and deep approach, MLE+MADA$_{ARZ}$, when applied on the test set, which was not used as part of our training or development, i.e., a completely blind test. We see that on the test set, the addition of MADA$_{ARZ}$ improves results relatively more as compared to the development set.

Approach	Exact Match		Norm Match		w/s
	Acc%	ER%	Acc%	ER%	
Baseline	76.8	____	90.5	____	____
MADA$_{MSA}$	83.6	29.3	91.7	12.6	70
CEC	90.0	56.9	93.9	35.8	5,584
MLE	90.5	59.1	94.6	43.2	6,698
CEC+MLE	91.5	63.4	95.2	49.5	4,284
MLE+CEC	90.7	59.9	94.7	44.2	4,284
MADA$_{ARZ}$	91.1	61.6	95.2	49.5	53
MADA$_{ARZ}$+CEC	91.5	63.4	95.4	51.6	53
MADA$_{ARZ}$+MLE	91.9	65.1	95.8	55.8	53
CEC+MADA$_{ARZ}$	92.2	66.4	95.6	53.7	53
MLE+MADA$_{ARZ}$	**92.6**	**68.1**	**95.8**	**55.8**	**53**
MADA$_{ARZ}$+CEC+MLE	91.8	64.7	95.6	53.7	52
CEC+MLE+MADA$_{ARZ}$	92.0	65.5	95.8	55.8	52

Table 3: Comparison of the performance of the different codafication approaches on the development corpus. *Acc* stands for Accuracy; *ER* is error reduction against the Baseline. w/s is speed (words/sec).

Approach	Exact Match		Norm Match	
	Acc%	ER%	Acc%	ER%
Baseline	75.5	____	89.7	____
CEC+MLE	91.3	64.5	94.8	49.5
MLE+MADA$_{ARZ}$	**92.9**	**71.0**	**95.5**	**56.3**

Table 4: Comparison of the performance of the different codafication approaches on the test corpus. *Acc* stands for Accuracy; *ER* is error reduction against the Baseline.

7.2 Extrinsic Evaluation

Morphological Analysis We tested the effect of codafication on morphological tagging, specifically full POS and lemma determination in context by the morphological tagger MADA$_{ARZ}$. Here, we are evaluating MADA$_{ARZ}$ not on its conversion to CODA (as above), but on its core functionality, namely morphological tagging. We compare the performance of MADA$_{ARZ}$ against running CEC+MLE+MADA$_{ARZ}$. When tested on the development set, the initial CEC+MLE codafication step helps MADA$_{ARZ}$ improve the identification of the complete Arabic (Buckwalter) POS tag from 84% to 85.3%, for an error reduction of 8.1%, while the correct lemma choice increases from 85.2% to 85.7%, for an error reduction of 3.4%. When tested on the test set, we get improvements on the choice of the complete Buckwalter POS tag and lemma from 84.5% to 85.4% (5.8% error reduction) and from 86.3% to 86.7% (2.9% error reduction), respectively.

Arabic to English MT The goal of this experiment is to test the effect of codafication on machine translation from dialectal Arabic to English. We use the open-source Moses toolkit (Koehn et al., 2007) to build a phrase-based SMT system. We use MGIZA++ for word alignment (Gao and Vogel, 2008). Phrase translations of up to 8 words are extracted in the phrase table. We use SRILM (Stolcke, 2002) with modified Kneser-Ney smoothing to build two 4-gram language models. The first model is trained on the English side of the bitext, while the other is trained on the English Gigaword data. Feature weights are tuned to maximize BLEU (Papineni et al., 2002) on a development set using MERT (Och, 2003). We perform case-insensitive evaluation in terms of the BLEU metric.

We train the system on dialectal Arabic-English parallel data, obtained from several LDC corpora, which amounts to ∼500k sentences with 3.8M untokenized words on the Arabic side. The development set, used for tuning the parameters of the MT system, has 1,547 sentences with 15,585 untokenized Arabic words. The test set has 1,065 sentences with

12,116 untokenized Arabic words. Both development and test sets have two reference translations each. The English data is lower-cased and tokenized using simple punctuation-based rules.

We build two systems which vary in preprocessing of the Arabic text. The baseline system applies only simple punctuation-based rules. The second system applies our codafication in addition to punctuation separation. The Arabic text is Alif/Ya normalized and is kept untokenized in both settings. The baseline system achieves a BLEU score of 22.1%. The system using codafication obtains a BLEU score of 22.6%, and outperforms the baseline by 0.5% absolute BLEU points. This result shows that improvements observed in intrinsic evaluation of codafication carry on to the extrinsic task of machine translation.

7.3 Error Analysis

We conducted an error analysis for the best performing codafication approach on the development set. The most frequent error types are listed in Table 5. About two thirds of the errors are CODA phonological and orthographical errors, denoted by CODA-Phon and CODA-Orth, respectively. The wrong additions and deletions of ا *A* and ي *y* and the ت/ث *t/θ* transformations are examples of CODA phonological errors. The CODA orthographic errors include cases such as the ي/ى *y/ý* transformations. 21% of the errors are not real errors in the codafication output, but result from gold errors. Finally, about 13% of the errors are wrong merges and splits for the the negation particle ما *mA*, the vocative particle يا *yA* and the indirect-object *l+pronouns*.

Error Type	Description	Percentage
Gold Error	*Annotation Error*	21.0
CODA-**Orth**	ه $h \Leftrightarrow$ ة $\hbar$	13.7
CODA-**Phon**	ا $A \rightarrow \epsilon$	8.7
Merge	ما *mA*/NEG_PART	7.3
CODA-**Phon**	$\epsilon \rightarrow$ ي y	6.8
CODA-**Phon**	$\epsilon \rightarrow$ ا A	5.9
CODA-**Orth**	ي $y \Leftrightarrow$ ى $\acute{y}$	4.1
Merge	يا *yA*/VOC_PART	3.7
CODA-**Phon**	ه $h \Leftrightarrow$ ح H	3.2
CODA-**Phon**	ي $y \Leftrightarrow$ ى $\acute{y}$	3.2

Table 5: System Error Analysis: the most frequent error types.

8 Conclusion and Future Work

We have presented the problem of transforming spontaneous orthography of the Egyptian Arabic dialect into a conventionalized form, CODA. Our best technique involves a combination of character transformations, whole-word transformations, and the use of a full morphological tagger. The tagger can be omitted for a small decrease in performance and a large increase in speed.[2] In future work, we plan to extend our approach to other Arabic dialects. We will also investigate incorporating the unsupervised work of Dasigi and Diab (2011) into our algorithm, as well as other unsupervised techniques.

Acknowledgment

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. We thank three anonymous reviewers for helpful comments, and Ryan Roth for help with running MADA.

[2]Our system will be freely available. Please contact the authors for more information.

References

G. Abuhakema, R. Faraj, A. Feldman, and E. Fitzpatrick. 2008. Annotating an Arabic Learner Corpus for Error. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

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.

Mohamed I. Alkanhal, Mohammed A. Al-Badrashiny, Mansour M. Alghamdi, and Abdulaziz O. Al-Qabbany. 2012. Automatic Stochastic Arabic Spelling Correction With Emphasis on Space Insertions and Deletions. *IEEE Transactions on Audio, Speech & Language Processing*, 20:2111–2122.

Chiraz Ben Othmane Zribi and Mohammed Ben Ahmed. 2003. Efficient Automatic Correction of Misspelled Arabic Words Based on Contextual Information. In *Proceedings of the Knowledge-Based Intelligent Information and Engineering Systems Conference*, Oxford, UK.

Pradeep Dasigi and Mona Diab. 2011. CODACT: TowardsIdentifying Orthographic Variants in Dialectal Arabic. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 318–326, Chaing Mai, Thailand.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, SETQA-NLP '08, pages 49–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash and Ryan Roth. 2011. Using deep morphology to improve automatic error detection in arabic handwriting recognition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 875–884, Portland, Oregon, USA, June. Association for Computational Linguistics.

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, Owen Rambow, and Ryan Roth. 2010. MADA+TOKAN Manual. Technical Report CCLS-10-01, Center for Computational Learning Systems (CCLS), Columbia University.

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

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

Bassam Haddad and Mustafa Yaseen. 2007. Detection and Correction of Non-Words in Arabic: A Hybrid Approach. *International Journal of Computer Processing Of Languages (IJCPOL)*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Ahmed Hassan, Sara Noeman, and Hany Hassan. 2008. Language Independent Text Correction using Finite State Automata. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP 2008)*.

Clive Holes. 2004. *Modern Arabic: Structures, Functions, and Varieties*. Georgetown Classics in Arabic Language and Linguistics. Georgetown University Press.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Okan Kolak and Philip Resnik. 2002. OCR error correction using a noisy channel model. In *Proceedings of the second international conference on Human Language Technology Research*.

Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4).

Mohamed Maamouri, Ann Bies, Seth Kulick, Dalila Tabessi, and Sondos Krouna. 2012. Egyptian Arabic Treebank Pilot.

Walid Magdy and Kareem Darwish. 2006. Arabic OCR Error Correction Using Character Segment Correction, Language Modeling, and Shallow Morphology. In *Proceedings of 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 408–414, Sydney, Austrailia.

Franz Josef Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22:73–90.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

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.

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, Amin Allam, and Abdallah Gomah. 2003. Towards Automatic Spell Checking for Arabic. In *Conference on Language Engineering, ELSE*, Cairo, Egypt.

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

K. Shaalan, R. Aref, and A. Fahmy. 2010. An approach for analyzing and correcting spelling errors for non-native Arabic learners. *Proceedings of Informatics and Systems (INFOS)*.

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.

Jun Wang, Zucker, and Jean-Daniel. 2000. Solving multiple-instance problem: A lazy learning approach. In Pat Langley, editor, *17th International Conference on Machine Learning*, pages 1119–1125.

Janet C. E. Watson. 2002. *The Phonology and Morphology of Arabic*. Oxford University Press.