Using Deep Morphology to Improve Automatic Error Detection in Arabic Handwriting Recognition

Nizar Habash and **Ryan M. Roth**
Center for Computational Learning Systems
Columbia University
{habash,ryanr}@ccls.columbia.edu

Abstract

Arabic handwriting recognition (HR) is a challenging problem due to Arabic's connected letter forms, consonantal diacritics and rich morphology. In this paper we isolate the task of identification of erroneous words in HR from the task of producing corrections for these words. We consider a variety of linguistic (morphological and syntactic) and non-linguistic features to automatically identify these errors. Our best approach achieves a roughly ∼15% absolute increase in F-score over a simple but reasonable baseline. A detailed error analysis shows that linguistic features, such as lemma (i.e., citation form) models, help improve HR-error detection precisely where we expect them to: semantically incoherent error words.

1 Introduction

After years of development, optical character recognition (OCR) for Latin-character languages, such as English, has been refined greatly. Arabic, however, possesses a complex orthography and morphology that makes OCR more difficult (Märgner and Abed, 2009; Halima and Alimi, 2009; Magdy and Darwish, 2006). Because of this, only a few systems for Arabic OCR of printed text have been developed, and these have not been thoroughly evaluated (Märgner and Abed, 2009). OCR of Arabic handwritten text (handwriting recognition, or HR), whether online or offline, is even more challenging compared to printed Arabic OCR, where the uniformity of letter shapes and other factors allow for easier recognition (Biadsy et al., 2006; Natarajan et al., 2008; Saleem et al., 2009).

OCR and HR systems are often improved by performing post-processing; these are attempts to evaluate whether each word, phrase or sentence in the OCR/HR output is legal and/or probable. When an illegal word or phrase is discovered (error detection), these systems usually attempt to generate a legal alternative (error correction). In this paper, we present a HR error *detection* system that uses deep lexical and morphological feature models to locate possible "problem zones" – words or phrases that are likely incorrect – in Arabic HR output. We use an off-the-shelf HR system (Natarajan et al., 2008; Saleem et al., 2009) to generate an N-best list of hypotheses for each of several scanned segments of Arabic handwriting. Our problem zone detection (PZD) system then tags the potentially erroneous (problem) words. A subsequent HR post-processing system can then focus its effort on these words when generating additional alternative hypotheses. We only discuss the PZD system and not the task of new hypothesis generation; the evaluation is on error/problem identification. PZD can also be useful in highlighting erroneous text for human post-editors.

This paper is structured as follows: Section 2 provides background on the difficulties of the Arabic HR task. Section 3 presents an analysis of HR errors and defines what is considered a problem zone to be tagged. The experimental features, data and other variables are outlined in Section 4. The experiments are presented and discussed in Section 5. We discuss and compare to some related work in detail in Section 6. Conclusions and suggested avenues of for future progress are presented in Section 7.

2 Arabic Handwriting Recognition Challenges

Arabic has several orthographic and morphological properties that make HR challenging (Darwish and Oard, 2002; Magdy and Darwish, 2006; Märgner and Abed, 2009).

2.1 Arabic Orthography Challenges

The use of cursive, connected script creates problems in that it becomes more difficult for a machine to distinguish between individual characters. This is certainly not a property unique to Arabic; methods developed for other cursive script languages (such as Hidden Markov Models) can be applied successfully to Arabic (Natarajan et al., 2008; Saleem et al., 2009; Märgner and Abed, 2009; Lu et al., 1999).

Arabic writers often make use of elongation (tatweel/kashida) to beautify the script. Arabic also contains certain ligature constructions that require consideration during OCR/HR (Darwish and Oard, 2002). Sets of dots and optional diacritic markers are used to create character distinctions in Arabic. However, trace amounts of dust or dirt on the original document scan can be easily mistaken for these markers (Darwish and Oard, 2002). Alternatively, these markers in handwritten text may be too small, light or closely-spaced to readily distinguish, causing the system to drop them entirely. While Arabic disconnective letters may make it hard to determine word boundaries, they could plausibly contribute to reduced ambiguity of otherwise similar shapes.

2.2 Arabic Morphology Challenges

Arabic words can be described in terms of their morphemes. In addition to concatenative prefixes and suffixes, Arabic has templatic morphemes called *roots* and *patterns*. For example, the word وكمكاتبهم *wkmkAtbhm*[1] (*w+k+mkAtb+hm*) 'and like their offices' has two prefixes and one suffix, in addition to a stem composed of the root ك ت ب *k-t-b* 'writing related' and the pattern *m1A23*.[2] Arabic words can also be described in terms of lexemes and inflectional features. The set of word forms that only vary inflectionally among each other is called the *lexeme*. A *lemma* is a particular word form used to represent the lexeme word set – a citation form that stands in for the class (Habash, 2010). For instance, the lemma مكتب *mktb* 'office' represents the class of all forms sharing the core meaning 'office': مكاتب *mkAtb* 'offices' (irregular plural), المكتب *Almktb* 'the office', لمكتبها *lmktbhA* 'for her office', and so on.

Just as the lemma abstracts over inflectional morphology, the root abstracts over both inflectional and derivational morphology and thus provides a very high level of lexical abstraction, indicating the "core" meaning of the word. The Arabic root ك ت ب *k-t-b* 'writing related', e.g., relates words like مكتب *mktb* 'office', مكتوب *mktwb* 'letter', and كتيبة *ktybħ* 'military unit (of conscripts)'.

Arabic morphology allows for tens of billions of potential, legal words (Magdy and Darwish, 2006; Moftah et al., 2009). The large potential vocabulary size by itself complicates HR methods that rely on conventional, word-based dictionary lookup strategies. In this paper we consider the value of morpholexical and morpho-syntactic features such as lemmas and part-of-speech tags, respectively, that may allow machine learning algorithms to learn generalizations. We do not consider the root since it has been shown to be too general for NLP purposes (Larkey et al., 2007). Other researchers have used stems for OCR correction (Magdy and Darwish, 2006); we discuss their work and compare to it in Section 6, but we do not present a direct experimental comparison.

3 Problem Zones in Handwriting Recognition

3.1 HR Error Classifications

We can classify three types of HR errors: substitutions, insertions and deletions. Substitutions involve replacing the correct word by another incorrect form. Insertions are words that are incorrectly added into the HR hypothesis. An insertion error is typically paired with a substitution error, where the two errors reflect a mis-identification of a single word as two words. Deletions are simply missing words. Examples of these different types of errors appear in Table 1. In the dev set that we study here (see Section 4.1), 25.8% of the words are marked as problematic. Of these, 87.2% are letter-based words (henceforth words), as opposed to 9.3% punctuation and 3.5% digits.

Orthogonally, 81.4% of all problem words are substitution errors, 10.6% are insertion errors and 7.9% are deletion errors. Whereas punctuation symbols are 9.3% of all errors, they represent over 38%

[1] All Arabic transliterations are presented in the HSB transliteration scheme (Habash et al., 2007).

[2] The digits in the pattern correspond to positions where root radicals are inserted.

REF	! !	الجودة Aljwdħ	عالية ςAlyħ	زبدة zbdħ		علبة ςlbħ	تصنعوا tSnςwA	أن Ân	والأندلس wAlÂndls		وفارس wfArs	القسطنطينية AlqsTnTynyħ	يافاتحي yAfAtHy	المسلمون Almslmwn	أيها ÂyhA	أعجزتم Âςjztm	
HYP		الجودة Aljwdħ	عالية ςAlyħ	يوه ywh	ن n	عليه ςlyh	تصرفوا tSrfwA	ان An	لمن lmn	والاخذ wAlAxð	وفارس wfArs	القسطنطينية AlqsTnTynyħ	باتانى bAtAný	المسلمون Almslmwn	ايها AyhA	ثم θm	أعير Âςyr
PZD		PROB DELX	OK	PROB INS	PROB SUB	PROB DOTS	PROB SUB	PROB ORTH	PROB INS	PROB SUB	OK	OK	PROB SUB	OK	PROB ORTH	PROB INS	PROB SUB

Table 1: An example highlighting the different types of Arabic HR errors. The first row shows the reference sentence (right-to-left). The second row shows an automatically generated hypothesis of the same sentence. The last row shows which words in the hypothesis are marked as *problematic* (PROB) by the system and the specific category of the problem (illustrative, not used by system): SUB (substituted), ORTH (substituted by an orthographic variant), DOTS (substituted by a word with different dotting), INS (inserted), and DELX (adjacent to a deleted word). The remaining words are tagged as OK. The reference translates as '*Are you unable O'Moslems, you who conquered Constantinople and Persia and Andalusia, to manufacture a tub of high quality butter!*'. The hypothesis roughly translates as '*I loan then O'Moslems Pattani Constantinople, and Persia and taking from whom that you spend on him N Yeoh high quality*'.

of all deletion errors, almost 22% of all insertion errors and less than 5% of substitution errors. Similarly digits, which are 3.5% of all errors, are almost 14% of deletions, 7% of insertions and just over 2% of all substitutions. Punctuation and digits bring different challenges: whereas punctuation marks are a small class, their shape is often confusable with Arabic letters or letter components, e.g., إ *Ǎ* and **!** or ر *r* and **,**. Digits on the other hand are a hard class to language model since the vocabulary (of multi-digit numbers) is infinite. Potentially this can be addressed using a pattern-based model that captures forms of digit sequences (such as date and currency formats); we leave this as future work.

Words (non-digit, non-punctuation) still constitute the majority in every category of error: 47.7% of deletions, 71.3% of insertions and over 93% of substitutions. Among substitutions, 26.5% are simple orthographic variants that are often normalized in Arabic NLP because they result from frequent inconsistencies in spelling: Alef Hamza forms (ا/أ/إ/آ *A/Â/Ǎ/Ā*) and Ya/Alef-Maqsura (ي/ى *y/ý*). If we consider whether the lemma of the correct word and its incorrect form are matchable, an additional 6.9% can be added to the orthographic variant sum (since all of these cases can share the same lemmas). The rest of the cases, or 59.7% of the words, involve complex orthographic errors. Simple dot misplacement can only account for 2.4% of all substitution errors. The HR system output does not contain any illegal non-words since its vocabulary is restricted by its training data and language models. The large proportion of errors involving lemma differences is consistent with the perception that most OCR/HR errors create semantically incoherent sentences. This suggests that lemma models can be helpful in identifying such errors.

3.2 Problem Zone Definition

Prior to developing a model for PZD, it is necessary to define what is considered a 'problem'. Once a definition is chosen, gold problem tags can be generated for the training and test data by comparing the hypotheses to their references.[3] We decided in this paper to use a simple binary *problem* tag: a hypothesis word is tagged as "PROB" if it is the result of an insertion or substitution of a word. Deleted words in a hypothesis, which cannot be tagged themselves, cause their adjacent words to be marked as PROB instead. In this way, a subsequent HR post-processing system can be alerted to the possibility of a missing word via its surroundings (hence the idea of a problem 'zone'). Any words not marked as PROB are given an "OK" tag (see the PZD row of Table 1). We describe in Section 5.6 some preliminary experiments we conducted using more fine-grained tags.

4 Experimental Settings

4.1 Training and Evaluation Data

The data used in this paper is derived from image scans provided by the Linguistic Data Consortium (LDC) (Strassel, 2009). This data consists of high-resolution (600 dpi) handwriting scans of Arabic text taken from newswire articles, web logs and

[3]For clarity, we refer to these tags as 'gold', whereas the correct segment for a given hypothesis set is called the 'reference'.

newsgroups, along with ground truth annotations and word bounding box information. The scans include variations in scribe demographic background, writing instrument, paper and writing speed.

The BBN Byblos HR system (Natarajan et al., 2008; Saleem et al., 2009) is then used to process these scanned images into sequences of *segments* (sentence fragments). The system generates a ranked N-best list of hypotheses for each segment, where N could be as high as 300. On average, a segment has 6.87 words (including punctuation).

We divide the N-best list data into training, development (dev) and test sets.[4] For training, we consider two sets of size 2000 and 4000 segments (S) with the 10 top-ranked hypotheses (H) for each segment to provide additional variations.[5] The references are also included in the training sets to provide examples of perfect text. The dev and test sets use 500 segments with one top-ranked hypothesis each $\{H{=}1\}$. We can construct a trivial PZD baseline by assuming all the input words are PROBs; this results in baseline % Precision/Recall/F-scores of 25.8/100/41.1 and 26.0/100/41.2 for the dev and test sets, respectively. Note that in this paper we eschew these baselines in favor of comparison to a non-trivial baseline generated by a simple PZD model.

4.2 PZD Models and Features

The PZD system relies on a set of SVM classifiers trained using morphological and lexical features. The SVM classifiers are built using Yamcha (Kudo and Matsumoto, 2003). The SVMs use a quadratic polynomial kernel. For the models presented in this paper, the static feature window context size is set to +/- 2 words; the previous two (dynamic) classifications (i.e. targets) are also used as features. Experiments with smaller window sizes result in poorer performance, while a larger window size (+/- 6 words) yields roughly the same performance at the expense of an order-of-magnitude increase in required training time. Over 30 different combinations of features were considered. Table 2 shows the individual feature definitions.

[4]Naturally, we do not use data that the BBN Byblos HR system was trained on.

[5]We conducted additional experiments where we varied the number of segments and hypotheses and found that the system benefited from added variety of segments more than hypotheses. We also modified training composition in terms of the ratio of problem/non-problem words; this did not help performance.

Simple	**Description**
`word`	The surface word form
`nw`	Normalized word: the word after Alef, Ya and digit normalization
`pos`	The part-of-speech (POS) of the word
`lem`	The lemma of the word
`na`	No-analysis: a binary feature indicating whether the morphological analyzer produced any analyses for the word

Binned	**Description**
`nw` N-grams	Normword 1/2/3-gram probabilities
`lem` N-grams	Lemma 1/2/3-gram probabilities
`pos` N-grams	POS 1/2/3-gram probabilities
`conf`	Word confidence: the ratio of the number of hypotheses in the N-best list that contain the word over the total number of hypotheses

Table 2: PZD model features. Simple features are used directly by the PZD SVM models, whereas Binned features' (numerical) values are reduced to a small, labeled category set whose labels are used as model features.

In order to obtain the morphological features, all of the training and test data is passed through MADA 3.0, a software tool for Arabic morphological analysis disambiguation (Habash and Rambow, 2005; Roth et al., 2008; Habash et al., 2010). For these experiments, MADA provides the `pos` (using MADA's native 34-tag set) and the lemma for each word. Occasionally MADA will not be able to produce any interpretations (analyses) for a word; since this is often a sign that the word is misspelled or uncommon, we define a binary `na` feature to indicate when MADA fails to generate analyses.

In addition to using the MADA features directly, we also develop a set of nine N-gram models (where N=1, 2, and 3) for the `nw`, `pos`, and `lem` features defined in Table 2. We train these models using 220M words from the Arabic Gigaword 3 corpus (Graff, 2007) which had also been run through MADA 3.0 to extract the `pos` and `lem` information. The models are built using the SRI Language Modeling Toolkit (Stolcke, 2002). Each word in a hypothesis can then be assigned a probability by each of these nine models. We reduce these probabilities into one of nine bins, with each successive bin representing an order of magnitude drop in probability (the final bin is reserved for word N-grams which did not appear in the models). The bin labels are used as the SVM features.

Finally, we also use a word confidence (`conf`) feature, which is aimed at measuring the frequency with which a given word is chosen by the HR system for a given segment scan. The `conf` is defined here as the ratio of the number of hypotheses in the N-best list that the word appears in to the total number of hypotheses. These numbers are calculated using the original N-best hypothesis list, before the data is trimmed to H={1, 10}. Like the N-grams, this number is binned; in this case there are 11 bins, with 10 spread evenly over the [0,1) range, and an extra bin for values of exactly 1 (i.e., when the word appears in every hypothesis in the set).

5 Results

We describe next different experiments conducted by varying the features used in the PZD model. We present the results in terms of F-score only for simplicity; we then conduct an error analysis that examines precision and recall.

5.1 Effect of Feature Set Choice

Selecting an appropriate set of features for PZD requires extensive testing. Even when only considering the few features described in Table 2, the parameter space is quite large. Rather then exhaustively test every possible feature combination, we selectively choose feature subsets that can be compared to gain a sense of the incremental benefit provided by individual features.

5.1.1 Simple Features

Table 3 illustrates the result of taking a baseline feature set (containing `word` as the only feature) and adding a single feature from the Simple set to it. The result of combining all the Simple features is also indicated. From this, we see that Simple features, even collectively, provide only minor improvements.

5.1.2 Binned Features

Table 4 shows models which include both Simple and Binned features. First, Table 4 shows the effect of adding `nw` N-grams of successively higher orders to the `word` baseline. Here we see that even a simple unigram provides a significant benefit (compared to the improvements gained in Table 3). The largest improvement comes with the addition of the bigram (thus introducing context into the model), but the trigram provides only a slight improvement above that. This implies that pursuing higher order N-grams will result in negligible returns.

In the next part of Table 4, we see that the single feature (`pos`) which provided the highest single-feature benefit in Table 3 does not provide similar improvements under these combinations, and in fact seems detrimental. We also note that using all the features in one model is outperformed by more selective choices. Here, the best performer is the model which utilizes the `word`, `nw` N-grams,

Feature Set	**F-score**	*%Imp*
`word`	43.85	–
`word+nw`	43.86	~0
`word+na`	44.78	2.1
`word+lem`	45.85	4.6
`word+pos`	45.91	4.7
`word+nw+pos+lem+na`	**46.34**	**5.7**

Table 3: PZD F-scores for simple feature combinations. The training set used was {S=2000, H=10} and the models were evaluated on the dev set. The improvement over the **word** baseline case is also indicated. *%Imp* is the relative improvement over the first row.

Feature Set	**F-score**	*%Imp*
`word`	43.85	–
`word+nw` 1-gram	49.51	12.9
`word+nw` 1-gram+`nw` 2-gram	59.26	35.2
`word+nw` N-grams	59.33	35.3
`+pos`	58.50	33.4
`+pos` N-grams	57.35	30.8
`+lem+lem` N-grams	59.63	36.0
`+lem+lem` N-grams`+na`	59.93	36.7
`+lem+lem` N-grams`+na+nw`	59.77	36.3
`+lem`	**60.92**	**38.9**
`+lem+na`	60.47	37.9
`+lem+lem` N-grams	60.44	37.9

Table 4: PZD F-scores for models that include Binned features. The training set used was {S=2000, H=10} and the models were evaluated on the dev set. The improvement over the **word** baseline case is also indicated. The label "N-grams" following a Binned feature refers to using 1, 2 *and* 3-grams of that feature. Indentation marks accumulative features in model. The best performing row (with bolded score) is *word+nw N-grams+lem*.

Base Feature Set	**F-score**		*%Imp*
		`+conf`	
`word`	43.85	55.83	27.3
`+nw` N-grams	59.33	61.71	4.0
`+lem`	**60.92**	62.60	2.8
`+lem+na`	60.47	**63.14**	4.4
`+lem+lem` N-grams	60.44	62.88	4.0
`+pos+pos` N-grams `+na+nw` (**all** system)	59.77	62.44	4.5

Table 5: PZD F-scores for models when word confidence is added to the feature set. The training set used was {S=2000, H=10} and the models were evaluated on the dev set. The improvement generated by including word confidence is indicated. The label "N-grams" following a Binned feature refers to using 1, 2 *and* 3-grams of that feature. Indentation marks accumulative features in model. *%Imp* is the relative improvement gained by adding the *conf* feature.

	S = **2000**	S = **4000**	
Feature Set	**F-score**	**F-score**	*%Imp*
`word`	43.85	52.08	18.8
`word+conf`	55.83	57.50	3.0
`word+nw` N-grams`+lem` `+conf` (**best** system)	62.60	**66.34**	6.0
`+na`	**63.14**	66.21	4.9
`+lem` N-grams	62.88	64.43	2.5
all	62.44	65.62	5.1

Table 6: PZD F-scores for selected models when the number of training segments (S) is doubled. The training set used was {S=2000, H=10} and {S=4000, H=10}, and the models were evaluated on the dev set. The label "N-grams" following a Binned feature refers to using 1, 2 *and* 3-grams of that feature. Indentation marks accumulative features in model.

and `lem` as the only features. However, the differences among this model and the other models using `lem` Table 4 are not statistically significant. The differences between this model and the other lower performing models are statistically significant ($p<0.05$).

5.1.3 Word Confidence

The `conf` feature deserves special consideration because it is the only feature which draws on information from across the entire hypothesis set. In Table 5, we show the effect of adding `conf` as a feature to several base feature sets taken from Table 4. Except for the baseline case, `conf` provides a relatively consistent benefit. The large (27.3%) improvement gained by adding `conf` to the **word** baseline shows that `conf` is a valuable feature, but the smaller improvements in the other models indicate that the information it provides largely overlaps with the information already present in those models. The differences among the last four models (all including `lem`) in Table 5 are not statistically significant. The differences between these four models and the first two are statistically significant ($p<0.05$).

5.2 Effect of Training Data Size

In order to allow for rapid examination of multiple feature combinations, we restricted the size of the training set (S) to maintain manageable training times. With this decision comes the implicit assumption that the results obtained will scale with additional training data. We test this assumption by taking the best-performing feature sets from Table 5 and training new models using twice the training data {S=4000}. The results are shown in Table 6. In each case, the improvements are relatively consistent (and on the order of the gains provided by the inclusion of `conf` as seen in Table 5), indicating that the model performance does scale with data size. However, these improvements come with a cost of a roughly 4-7x increase in training time. We note that the value of doubling S is roughly 3-6x times greater for the **word** baseline than the others; however, simply adding `conf` to the baseline provides an even greater improvement than doubling S. The differences between the final four models in Table 6 are not statistically significant. The differences between these models and the first two models in the table are statistically significant ($p<0.05$). For convenience, in the next section we refer to the third model listed in Table 6 as the **best** system (because it has the highest absolute F-score on the large data set), but readers should recall that these four models are roughly equivalent in performance.

5.3 Error Analysis

In this section, we look closely at the performance of a subset of systems on different types of problem words. We compare the following model settings: for {S=4000} training, we use `word`, `word` + `conf`, the best system from Table 6 and the model

(a)	S=4000				S=2000
	word	**wconf**	**best**	**all**	**all**
Precision	54.7	59.5	67.1	**67.4**	62.4
Recall	49.7	55.7	**65.6**	64.0	62.5
F-score	52.1	57.5	**66.3**	65.6	62.4
Accuracy	76.4	78.7	**82.8**	82.7	80.6

(b)	**%Prob**	**word**	**wconf**	**best**	**all**	**all**
Words	87.2	51.8	57.3	**68.5**	67.1	64.9
Punc.	9.3	39.5	44.7	**50.0**	46.1	40.8
Digits	3.5	24.1	44.8	34.5	34.5	**62.1**
INS	10.6	46.0	49.4	**62.1**	**62.1**	55.2
DEL	7.9	**29.2**	20.0	24.6	21.5	27.7
SUB	81.4	52.2	60.0	**70.0**	68.4	66.9
Ortho	21.6	**63.3**	51.4	52.5	53.7	48.6
Lemma	5.6	45.7	52.2	**63.0**	52.2	54.4
Semantic	54.2	48.4	64.2	**77.7**	75.9	75.5

Table 7: Error analysis results comparing the performance of multiple systems over different metrics (a) and word/error types (b). %Prob shows the distribution of problem words into different word types (word, punctuation and digit) and error types. INS, DEL and SUB stand for insertion, deletion and substitution error types, respectively. *Ortho* stands for orthographic variant. *Lemma* stands for 'shared lemma'. The columns to the right of the %Prob column show recall percentage for each word/error type.

using all possible features (**word**, **wconf**, **best** and **all**, respectively); and we also use **all** trained with {S=2000}. We consider the performance in terms of precision and recall in addition to F-score – see Table 7 (a). We also consider the percentage of recall per error type, such as word/punctuation/digit or deletion/insertion/substitution and different types of substitution errors – see Table 7 (b). The second column in this table (**%Prob**) shows the distribution of gold-tagged problem words into word and error type categories.

Overall, there is no major tradeoff between precision and recall across the different settings; although we can observe the following: (i) adding more training data helps precision more than recall (over three times more) – compare the last two columns in Table 7 (a); and (ii) the best setting has a slightly lower precision than **all** features, although a much better recall – compare columns 4 and 5 in Table 7 (a).

The performance of different settings on words is generally better than punctuation and that is better than digits. The only exceptions are in the digit category, which may be explained by that category's small count which makes it prone to large percentage fluctuations.

In terms of error type, the performance on substitutions is better than insertions, which is in turn better than deletions, for all systems compared. This makes sense since deletions are rather hard to detect and they are marked on possibly correct adjacent words, which may confuse the classifiers. One insight for future work is to develop systems for different types of errors. Considering substitutions in more detail, we see that surprisingly, the simple approach of using the word feature only (without `conf`) correctly recalls a bigger proportion of problems involving orthographic variants than other settings. It seems that the more complex the model, the harder it is to model these cases correctly. Error types that include semantic variations (different lemmas) or shared lemmas (but not explained by orthographic variation), are by contrast much harder for the simple models. The more complex models do quite well recalling errors involving semantically incoherent substitutions (around 77.7% of those cases) and words that share the same lemma but vary in inflectional features (63% of those cases). These two results are quite a jump from the basic word baseline (around 29% and 18% respectively).

The simple addition of data seems to contribute more towards the orthographic variation errors and less towards semantic errors. The different settings we use (training size and features) show some degree of complementarity in how they identify errors. We try to exploit this fact in Section 5.5 exploring some simple system combination ideas.

5.4 Blind Test Set

Table 8 shows the results of applying the same models described in Table 7 to a blind test set of yet unseen data. As mentioned in Section 4.1, the trivial baseline of the test set is comparable to the dev set. However, the test set is harder to tag than the dev set; this can be seen in the overall lower F-scores. That said, the relative order of performing features is the same as with the dev set, confirming that our **best** model is optimal for test too. On further study, we noticed that the reason for the test set difference is that the overlap in word forms between test and

	word	**wconf**	**best**	**all**
Precision	37.55	51.48	**57.01**	55.46
Recall	51.73	53.39	**61.97**	60.44
F-score	43.51	52.42	**59.39**	57.84
Accuracy	65.13	74.83	**77.99**	77.13

Table 8: Results on test set of 500 segments with one hypothesis each. The models were trained on the {S=4000, H=10} training set.

train is less than dev and train: 63% versus 81%, respectively on {S=4000}.

5.5 Preliminary Combination Analysis

In a preliminarily investigation of the value of complementarity across these different systems, we tried two simple model combination techniques. We restricted the search to the systems in the error analysis (Table 7).

First, we considered a sliding voting scheme where a word is marked as problematic if at least n systems agreed to that. Naturally, as n increases, precision increases and recall decreases, providing multiple tradeoff options. The range spans 49.1/83.2/61.8 (% Precision/Recall/F-score) at one end ($n = 1$) to 80.4/27.5/41.0 on the other ($n = all$). The best F-score combination was with $n = 2$ (any two agree) producing 62.8/72.4/67.3, an almost 1% higher than our best system.

In a different combination exploration, we exhaustively sought the best three systems from which any agreement (2 or 3) can produce an even better system. The best combination included the **word** model, the **best** model (both in {S=4000} training) and the **all** model (in {S=2000}). This combination yields 70.2/64.0/66.9, a lower F-score than the best general voting approach discussed above, but with a different bias towards better precision.

These basic exploratory experiments show that there is a lot of value in pursuing combinations of systems, if not for overall improvement, then at least to benefit from tradeoffs in precision and recall that may be appropriate for different applications.

5.6 Preliminary Tag Set Exploration

In all of the experiments described so far, the PZD models tag words using a binary tag set of PROB/OK. We may also consider more complex tag sets based on problem subtypes, such as SUB/INS/DEL/OK (where all the problem subtypes are differentiated), SUB/INS/OK (ignores deletions), and SUB/OK (ignores deletions and insertions). Care must be taken when comparing these systems, because the differences in tag set definition results in different baselines. Therefore we compare the % error reduction over the trivial baseline achieved in each case.

For an **all** model trained on the {S=2000, H=10} set, using the PROB/OK tag set results in a 36.3% error reduction over its trivial baseline (using the dev set). The corresponding SUB/INS/DEL/OK tag set only achieves a 34.8% error reduction. The SUB/INS/OK tag set manages a 40.1% error reduction, however. The SUB/OK tag set achieves a 38.9% error reduction. We suspect that the very low relative number of deletions (7.9% in the dev data) and the awkwardness of a DEL tag indicating a neighboring deletion (rather than the current word) may be confusing the models, and so ignoring them seems to result in a clearer picture.

6 Related Work

Common OCR/HR post-processing strategies are similar to spelling correction solutions involving dictionary lookup (Kukich, 1992; Jurafsky and Martin, 2000) and morphological restrictions (Domeij et al., 1994; Oflazer, 1996). Error detection systems using dictionary lookup can sometimes be improved by adding entries representing morphological variations of root words, particularly if the language involved has a complex morphology (Pal et al., 2000). Alternatively, morphological information can be used to construct supplemental lexicons or language models (Sari and Sellami, 2002; Magdy and Darwish, 2006).

In comparison to (Magdy and Darwish, 2006), our paper is about error detection only (done in using discriminative machine learning); whereas their work is on error correction (done in a standard generative manner (Kolak and Resnik, 2002)) with no assumptions of some cases being correct or incorrect. In essence, their method of detection is the same as our trivial baseline. The morphological features they use are *shallow* and restricted to breaking up a word into *prefix+stem+suffix*; whereas we analyze words into their lemmas, abstracting away over a large number of variations. We also made use of part-of-speech tags, which they do not use, but suggest may help. In their work, the morphological features did not help (and even hurt a little), whereas for us, the lemma feature actually helped. Their hypothesis that their large language model (16M words) may be responsible for why the word-based models outperformed stem-based (morphological) models is challenged by the fact that our language model data (220M words) is an order of magnitude larger, but we are still able to show benefit for using morphology. We cannot directly compare to their results because of the different training/test sets and target (correction vs detection); however, we should note that their starting error rate was quite high (39% on Alef/Ya normalized words), whereas our starting error rate is almost half of that (∼26% with unnormalized Alef/Yas, which account for almost 5% absolute of the errors). Perhaps a combination of the two kinds of efforts can push the perfomance on correction even further by biasing towards problematic words and avoiding incorrectly changing correct words. Magdy and Darwish (2006) do not report on percentages of words that they incorrectly modify.

7 Conclusions and Future Work

We presented a study with various settings (linguistic and non-linguistic features and learning curve) for automatically detecting problem words in Arabic handwriting recognition. Our best approach achieves a roughly ∼15% absolute increase in F-score over a simple baseline. A detailed error analysis shows that linguistic features, such as lemma models, help improve HR-error detection specifically where we expect them to: identifying semantically inconsistent error words.

In the future, we plan to continue improving our system by considering smarter trainable combination techniques and by separating the training for different types of errors, particularly deletions from insertions and substitutions. We would also like to conduct an extended evaluation comparing other types of morphological features, such as roots and stems, directly. One additional idea is to implement a lemma-confidence feature that examines lemma use in hypotheses across the document. This could potentially provide valuable semantic information at the document level.

We also plan to integrate our system with a system for producing correction hypotheses. We also will consider different uses for the basic system setup we developed to identify other types of text errors, such as spelling errors or code-switching between languages and dialects.

Acknowledgments

We would like to thank Premkumar Natarajan, Rohit Prasad, Matin Kamali, Shirin Saleem, Katrin Kirchhoff, and Andreas Stolcke. This work was funded under DARPA project number HR0011-08-C-0004.

References

Fadi Biadsy, Jihad El-Sana, and Nizar Habash. 2006. Online Arabic handwriting recognition using Hidden Markov Models. In *The 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR'10)*, La Baule, France.

Kareem Darwish and Douglas W. Oard. 2002. Term Selection for Searching Printed Arabic. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 261–268, New York, NY, USA. ACM.

Rickard Domeij, Joachim Hollman, and Viggo Kann. 1994. Detection of spelling errors in Swedish not using a word list en clair. *J. Quantitative Linguistics*, 1:1–195.

David Graff. 2007. Arabic Gigaword 3, LDC Catalog No.: LDC2003T40. Linguistic Data Consortium, University of Pennsylvania.

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

Mohamed Ben Halima and Adel M. Alimi. 2009. A multi-agent system for recognizing printed Arabic words. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing*. Prentice Hall, New Jersey, USA.

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*.

Taku Kudo and Yuji Matsumoto. 2003. Fast Methods for Kernel-Based Text Analysis. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pages 24–31, Sapporo, Japan, July. Association for Computational Linguistics.

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

Leah S. Larkey, Lisa Ballesteros, and Margaret E. Connell, 2007. *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, chapter Light Stemming for Arabic Information Retrieval. Springer Netherlands, Kluwer/Springer edition.

Zhidong Lu, Issam Bazzi, Andras Kornai, John Makhoul, Premkumar Natarajan, and Richard Schwartz. 1999. A Robust, Language-Independent OCR System. In *the 27th AIPR Workshop: Advances in Computer Assisted Recognition, SPIE*.

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.

Volker Märgner and Haikal El Abed. 2009. Arabic Word and Text Recognition - Current Developments. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

Mohsen Moftah, Waleed Fakhr, Sherif Abdou, and Mohsen Rashwan. 2009. Stem-based Arabic language models experiments. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.

Prem Natarajan, Shirin Saleem, Rohit Prasad, Ehry MacRostie, and Krishna Subramanian, 2008. *Arabic and Chinese Handwriting Recognition*, volume 4768 of *Lecture Notes in Computer Science*, pages 231–250. Springer, Berlin, Germany.

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

U. Pal, P. K. Kundu, and B. B. Chaudhuri. 2000. OCR error correction of an inflectional Indian language using morphological parsing. *J. Information Sci. and Eng.*, 16:903–922.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 117–120, Columbus, Ohio, June. Association for Computational Linguistics.

Shirin Saleem, Huaigu Cao, Krishna Subramanian, Marin Kamali, Rohit Prasad, and Prem Natarajan. 2009. Improvements in BBN's HMM-based Offline Handwriting Recognition System. In Khalid Choukri and Bente Maegaard, editors, *10th International Conference on Document Analysis and Recognition (ICDAR)*, Barcelona, Spain, July.

Toufik Sari and Mokhtar Sellami. 2002. MOrpho-LEXical analysis for correcting OCR-generated Arabic words (MOLEX). In *The 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, Niagara-on-the-Lake, Canada.

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.

Stephanie Strassel. 2009. Linguistic Resources for Arabic Handwriting Recognition. In Khalid Choukri and Bente Maegaard, editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, April. The MEDAR Consortium.