TECHNICAL REPORT

No. CCLS-10-02

Title: Automatic Identification of Errors in Arabic Handwriting Recognition

Authors: Nizar Habash and Ryan Roth

Automatic Identification of Errors in Arabic Handwriting Recognition

Nizar Habash and Ryan Roth
Center for Computational Learning Systems
Columbia University
475 Riverside Drive MC 7717
New York, NY 10115, USA
habash@cs.columbia.edu, 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. We also consider a learning curve varying in two dimensions: number of segments and number of n-best hypotheses to train on. We additionally evaluate the performance on different test sets with different degrees of errors in them. Our best approach achieves a roughly $\sim$20% absolute increase in F-score over a simple but reasonable baseline. A detailed error analysis shows that linguistic features, such as lemma models, help improve HR-error detection precisely where we expect them to: semantically inconsistent error words.

1 Introduction

Optical character recognition (OCR) has proven to be an incredibly valuable technology. After years of development, OCR systems for Latin-character languages such as English have been refined greatly. Arabic, however, possesses a complex morphology and orthography 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 throughly 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) – see Figure 1.

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). Common OCR/HR post-processing strategies are similar to spelling correction solutions involving dictionary lookup (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). However, the precise nature and depth of the morphological information is important; for example, one study reported that using a shallow 6-gram morphology model based only on Arabic stems and affixes was less useful than a simple word trigram model (Magdy and Darwish, 2006).

In this paper, we present a HR error detection system that uses lexical and morphological feature

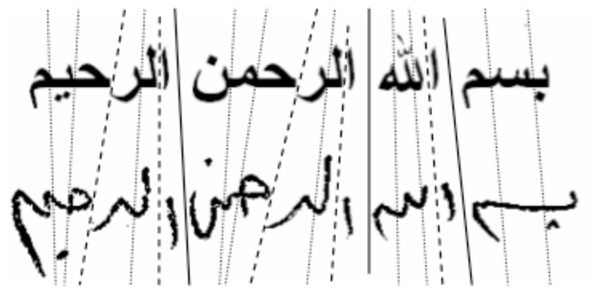

Figure 1: Example of aligning printed text (top line) with handwriting (bottom line). The dark solid lines indicate word boundaries, the dotted lines indicate character boundaries and the dashed lines indicate sub-word boundaries where the script is not connected.

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.

This paper is structured as follows: Section 2 provides background on the difficulties of the Arabic HR task. Section 3 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.

2 Arabic Handwriting Recognition

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 (see Figure 1). This is certainly not a property unique to Arabic; methods, such as Hidden Markov Models, developed for other cursive script languages 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 HR/OCR (Darwish and Oard, 2002). Sets of dots and optional diacritic markers are used to create character distinctions in Arabic. Unfortunately, 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 morphology, wherein words are constructed from affixes and stems (which are themselves constructed from many possible root patterns), 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 OCR methods that rely on conventional, word-based dictionary lookup strategies. In this paper we consider the value of morpho-lexical features such as lemmas and part-of-speech tags that may allow machine learning algorithms to learn generalizations.

2.3 HR Error Classifications

We can classify three types of OCR errors: substitutions, insertions and deletions. Substitutions involve replacing the correct word by another incorrect form. Insertions are words that are incorrectly added into the OCR hypothesis. An insertion error is typically paired with a substitution error, where the two errors reflect a mis-analysis of a single word split into two words. Deletions are simply missing words. Examples of these different types of errors appear in Table 1.

In the evaluation set that we study here, 25.8% of the words are marked as problematic. Of these,

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 in both Arabic script and transliteration. 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) words and the specific type of problem: 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 example words are shown in Arabic visual order (right-to-left). 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 '*Was loaned then O'Moslems Pattani Constantinople, and Persia and taking from whom that you spend on him N Yeoh high quality*'.

87.2% are letter-based words (henceforth words), as opposed to 9.3% punctuation and 3.5% digits. These ratios are comparable to the overall distribution of word types in the whole data set (words=86.8%, punctuation=9.3% and digits=3.9%).

Orthogonally, 81.4% of all problem words involve a substitution error, 10.6% an insertion error and 7.9% a deletion error. Whereas punctuation symbols are 9.3% of all errors, they represent over 38% 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. Punctuations 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*[1] and ! or ر and ,. Digits on the other hand are a hard class to language model since the vocabulary (of multi-digit numbers) is very large.

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 these cases can all 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 large proportion of errors involving lemma difference is consistent with the perception of most OCR errors creating semantically incoherent sentences. This suggests that language models on word and lemma level can be very helpful in identifying such errors.

3 Problem Zone Definition

Prior to developing a model for PZD, it is necessary to define what is considered a 'problem'. In addition, model training and evaluation requires that gold problem tags be generated for the training and test data.[2] We define 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).

To locate insertions, deletions and substitutions,

[1]All Arabic transliterations are provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007).

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

we run all the data sets through the *sclite* software originally developed for speech NLP (Kahn, 2001). *sclite* was used to compare each hypothesis with its reference, which are then used to create the gold tags for each data set.

4 Experimental Settings

4.1 Training and Test Data

The training and test data used in this system were produced by the off-the-shelf BBN Byblos OHR system (Natarajan et al., 2008; Saleem et al., 2009). This system takes scans of handwritten segments (see Figure 1) as input and produces a ranked N-best list of hypotheses for each segment. For this data, N could be as high as 300, but for some segments a smaller number was produced. On average, a segment has 6.87 words (including punctuation), with each segment and its neighbors forming a larger, complete sentence.

We divide the data into training and test sets. To reduce the computational times required to gain a clear understanding of the system performance, the data sets are further divided into subsets of reduced size, according to number of segments and maximum number of hypotheses. For training, sets using 20, 200, 2000 or 4000 segments (S_r) are used, with 1, 10, or 100 of the top-ranked hypotheses (H_r) for each segment. The references are also included in the training sets to provide examples of perfect text. Testing sets use 500 segments (S_e), with 1, 10 or 100 of the top-ranked hypotheses (H_e). Since we are mainly concerned with improving on the HR system's ranking and its top choice, we mainly focus on the $\{S_e$=500, H_e=1$\}$ test set for evaluation. We consider a total of 8 different training sets and 3 test sets.

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, 2005). Over 30 different combinations of features were considered. Table 2 shows the individual feature definitions.

In order to obtain morpho-lexical 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). 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 illegal, 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 a 220M word excerpt 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 were 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 as the ratio of the number of hypotheses in the set that the word appears in to the total number of hypotheses in the set. These numbers are calculated using the original N-best hypothesis list, before the data is trimmed to H=$\{1, 10, 100\}$. 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 1 (i.e., when the word appears in every hypothesis in the set).

5 Results

In the following sections, we describe different experiments conducted by varying the training data size and S_r/H_r composition (to determine which has a greater impact on PZD performance), the test data H_e distribution (to examine how well the PZD system works on lower-ranked hypotheses), and the features used in the PZD model (to determine which

Simple Features	**Description**
`word`	The surface word form
`nw`	Normalized word: the word after Alef, Ya and digit normalization
`pos`	The part-of-speech of the word
`lem`	The lemma of the word
`na`	No-analysis: a binary feature indicating whether a morphological analyzer is able to produce any analyses for the word
Binned Features	**Description**
`nw` N-grams	1, 2 and/or 3-gram probabilities of the normword
`lem` N-grams	1, 2 and/or 3-gram probabilities of the lemma
`pos` N-grams	1, 2 and/or 3-gram probabilities of the pos
`conf`	Word confidence: the ratio of the number of hypotheses in the set that contain the word over the total number of hypotheses in the set

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

F-score	S_r**=20**	S_r**=200**	S_r**=2000**
H_r=1	38.84	46.21	59.32
H_r=10	49.22	54.36	62.44
H_r=100	50.30	54.60	–
Train Time (sec)	S_r**=20**	S_r**=200**	S_r**=2000**
H_r=1	< 1	4	865
H_r=10	1	36	29,659
H_r=100	17	4,417	–

Table 3: PZD F-scores and training times for various training set S_r/H_r partitions; the model uses all the available features (**all**), and was evaluated with the {S_e=500, H_e=1} test set.

	% of Gold Hypothesis PROBs	**Trivial Baseline F-score**	**F-score** S_r**=2000** H_r**=10**
H_e=1	25.84	41.07	62.44
H_e=10	31.11	47.45	69.86
H_e=100	39.06	56.18	77.57

Table 4: F-score of H_e values (S_e=500 in each case). The first column shows the percentage of PROB tags that exist in the hypotheses of that test set – a trivial tagger would have the corresponding F-scores shown in the second column. The third column shows the F-score derived from the {S_r=2000, H_r=10, **all**} model on that test set.

have the most informative impact). 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 Training Data Composition

Table 3 and Table 4 illustrate the effect of increasing the number of segments and maximum number of hypotheses in both the training and test data. In these experiments, all the features listed in Table 2 (henceforth referred to as the **all** feature set) were used in the PZD model. In Table 3, there is a clear improvement in PZD F-score when increasing the training S_r or H_r. For the most part, increases in S_r provide more improvement than a similar increase in H_r, indicating the PZD model benefits more from segment diversity than from more hypothetical versions of the same segment. However, increasing S_r (and to a lesser extent, H_r) comes with a significantly non-linear increase in training time. The {S_r=2000, H_r=100} case was not examined because of this.

The first column of Table 4 shows the number of gold PROB tags present in test sets having different values of H_e. A trivial tagger which marks every input word as PROB would have precision equal to these numbers and perfect recall; the corresponding F-score for this tagger is shown the second column and can be considered as a trivial baseline. The third column shows how well a {S_r=2000, H_r=10, **all**} PZD model performs when evaluated on these

Feature Set	F-score	Improvement
word (`word` only)	43.85	–
`word` + `nw`	43.86	0.03 %
`word` + `na`	44.78	2.13 %
`word` + `lem`	45.85	4.57 %
`word` + `pos`	45.91	4.70 %
`word` + `nw` + `pos` + `lem`	46.02	4.94 %
`word` + `nw` + `pos` + `lem` + `na`	46.34	5.68 %

Table 5: PZD F-scores for simple feature combinations. The training set used was $\{S_r$=2000, H_r=10$\}$ and the models were evaluated on the $\{S_e$=500, H_e=1$\}$ test set. The improvement over the **word** baseline case is also indicated.

test sets. Here, there is a marked improvement as H_e increases; this is no doubt due to the inclusion of hypotheses that have lower HR rank, and thus PROBs are more frequent and more readily identified. In comparing these scores to the trivial baseline, we find that the PZD model provides a consistent $\sim$22% improvement, regardless of H_e. Since the intent of PZD is to improve on the top selection made by the HR system, we focus our evaluations on the $\{S_e$=500, H_e=1$\}$ henceforth.

5.2 Effect of Feature Set Choice

Selecting an appropriate set 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.

5.2.1 Simple Features

Table 5 examines 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. We note that the `word` baseline model (henceforth referred to as the **word** baseline) is only slightly improved over the trivial baseline shown in Table 4.

5.2.2 Binned Features

Table 6 shows models which include both Simple and Binned features. First, Table 6 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 5). 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 6, we see that the single feature (`pos`) which provided the highest benefit in Table 5 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, and `lem` as the only features. However, the differences among this model and the five that follow it in Table 6 are not statistically significant. The differences between this model and the other lower performing models are statistically significant ($p<0.05$).

5.2.3 Word Confidence

The `conf` feature deserves special consideration because it is the only feature which draws on information from across the hypothesis set. In Table 7, we show the effect of adding `conf` as a feature to several base feature sets taken from Table 6. Except for the baseline case, `conf` provides a relatively consistent benefit. The large (27.32%) improvement gained by adding `conf` to the **word** baseline shows that `conf` is valuable as a 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 in Table 7 are not statistically significant. The differences between these four models and the first two are statistically significant ($p<0.05$).

5.3 Effect of Training Data Size

In order to allow rapid examination of multiple feature combinations, we restricted the size of the training set (S_r) to maintain manageable training times.

Feature Set	**F-score**	**Improve.**
word baseline (`word` only)	43.85	–
`word` + `nw` unigram	49.51	12.91 %
`word` + `nw` unigram and bigram	59.26	35.16 %
`word` + `nw` unigram, bigram and trigram (N-grams)	59.33	35.32 %
`word` + `nw` N-grams + `pos`	58.50	33.41 %
`word` + `nw` N-grams + `pos` + `pos` N-grams	57.35	30.79 %
`word` + `nw` N-grams + `lem`	60.92	38.93 %
`word` + `nw` N-grams + `lem` + `na`	60.47	37.91 %
`word` + `nw` N-grams + `lem` + `lem` N-grams	60.44	37.85 %
`word` + `nw` N-grams + `pos` + `pos` N-grams + `lem` + `lem` N-grams	59.63	35.98 %
`word` + `nw` N-grams + `pos` + `pos` N-grams + `lem` + `lem` N-grams + `na`	59.93	36.67 %
`word` + `nw` N-grams + `pos` + `pos` N-grams + `lem` + `lem` N-grams + `na` + `nw`	59.77	36.31 %

Table 6: PZD F-scores for models that include Binned features. The training set used was {S_r=2000, H_r=10} and the models were evaluated on the {S_e=500, H_e=1} test set. The improvement over the **word** baseline case is also indicated. "N-grams" refers to using each of the 1, 2 and 3-grams as included Binned features.

	Original	**F-score**	**Improvement with**
Base Feature Set	**F-score**	`conf`	`conf` **added**
`word` only	43.85	55.83	27.32 %
`word` + `nw` N-grams	59.33	61.71	4.00 %
`word` + `nw` N-grams + `lem`	60.92	62.60	2.76 %
`word` + `nw` N-grams + `lem` + `na`	60.47	63.14	4.41 %
`word` + `nw` N-grams + `lem` + `lem` N-grams	60.44	62.88	4.03 %
`word` + `nw` N-grams + `pos` + `pos` N-grams + `lem` + `lem` N-grams + `na` + `nw`	59.77	62.44	4.47 %

Table 7: PZD F-scores for models when word confidence is added to the feature set. The training set used was {S_r=2000, H_r=10} and the models were evaluated on the {S_e=500, H_e=1} test set. The improvement generated by including word confidence is indicated. "Ngrams" refers to using each of the 1, 2 and 3-grams as included Binned features.

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 7 and training new models using twice the training data (S_r=4000). The results are shown in Table 8. 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 7) , 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_r 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_r. The differences between the final four models in Table 8 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 8 as the **best** system (because it has the highest absolute F-score on the large data set), but readers should recall that the these four models are roughly equivalent in performance.

5.4 Error Analysis

In this section, we look closely at the performance of a subset of systems on different types of prob-

Feature Set	S_r **= 2000 F-score**	S_r **= 4000 F-score**	**Improvement with more training data**
`word` only	43.85	52.08	18.77 %
`word` + `conf`	55.83	57.50	3.00 %
`word` + `nw` N-grams + `lem` + `conf` (**best** system)	62.60	66.34	5.97 %
`word` + `nw` N-grams + `lem` + `na` + `conf`	63.14	66.21	4.87 %
`word` + `nw` N-grams + `lem` + `lem` N-grams + `conf`	62.88	64.43	2.47 %
all	62.44	65.62	5.10 %

Table 8: PZD F-scores for selected models when the number of training segments (S_r) is doubled. The training set used was {S_r=2000, H_r=10} and {S_r=4000, H_r=10}, and the models were evaluated on the {S_e=500, H_e=1} test set. "Ngrams" refers to using each of the 1, 2 and 3-grams as included Binned features.

(a)	S_r**=4000**				S_r**=2000**
	word	**wconf**	**best**	**all**	**all**
Precision	54.70	59.45	67.12	67.35	62.36
Recall	49.69	55.68	65.57	63.98	62.52
F-score	52.08	57.50	66.34	65.62	62.44

(b)	**%GoldProb**	**word**	**wconf**	**best**	**all**	**all**
Words	87.2	51.8	57.3	68.5	67.1	64.9
Punctuation	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
Insertions	10.6	46.0	49.4	62.1	62.1	55.2
Deletions	7.9	29.2	20.0	24.6	21.5	27.7
Substitutions	81.4	52.2	60.0	70.0	68.4	66.9
OrthoVariant	21.6	63.3	51.4	52.5	53.7	48.6
Same Lemma	5.6	45.7	52.2	63.0	52.2	54.4
Semantic	48.6	48.5	64.1	79.1	77.1	76.1

Table 9: Error analysis results comparing the performance of multiple systems over different error types.

lem words. We compare the following model settings: for {S_r=4000} training, we use `word`, `word` + `conf`, the best system from Table 8 and the model using all possible features (**word**, **wconf**, **best** and **all**, respectively); and we also use **all** trained with {S_r=2000}. We consider the performance in terms of precision and recall in addition to F-score – see Table 9 (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 9 (b). The second column in this table (**%GoldProb**) shows importance of a category as a percentage of all gold-tagged problem words.

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 9 (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 9 (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 prune 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 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 79.1% 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 30% 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 the next section exploring some simple system combination ideas.

5.5 Preliminary Combination Analysis

In a preliminarily investigation of the value of complementarity across these different systems, we tried two model simple combination techniques. We restricted the search to the systems in the error analysis (Table 9).

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 $\{S_r$=4000$\}$ training) and the **all** model (in $\{S_r$=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.

6 Conclusions and Future Work

We presented a large 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 $\sim$20% 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 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.

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.

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.

Jeremy Kahn. 2001. Speech recognizer sclite. http://cpan.uwinnipeg.ca/dist/Speech-Recognizer-ScLite.

Taku Kudo. 2005. Yamcha: Yet another multipurpose chunk annotator. http://chasen.org/ taku/software/yamcha/.

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. Morpholexical 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.