Arabic OCR Error Correction Using Character Segment Correction, Language Modeling, and Shallow Morphology

Walid Magdy and Kareem Darwish
IBM Technology Development Center
P.O. Box 166 El-Ahram, Giza, Egypt
{wmagdy,darwishk}@eg.ibm.com

Abstract

This paper explores the use of a character segment based character correction model, language modeling, and shallow morphology for Arabic OCR error correction. Experimentation shows that character segment based correction is superior to single character correction and that language modeling boosts correction, by improving the ranking of candidate corrections, while shallow morphology had a small adverse effect. Further, given sufficiently large corpus to extract a dictionary and to train a language model, word based correction works well for a morphologically rich language such as Arabic.

1 Introduction

Recent advances in printed document digitization and processing led to large scale digitization efforts of legacy printed documents producing document images. To enable subsequent processing and retrieval, the document images are often transformed to character-coded text using Optical Character Recognition (OCR). Although OCR is fast, OCR output typically contains errors. The errors are even more pronounced in OCR'ed Arabic text due to Arabic's orthographic and morphological properties. The introduced errors adversely affect linguistic processing and retrieval of OCR'ed documents. This paper explores the effectiveness post-OCR error correction. The correction uses an improved character segment based noisy channel model, language modeling, and shallow morphological processing to correct OCR errors. The paper will be organized as follows: Section 2 provides background information on Arabic OCR and OCR error correction; Section 3 presents the error correction methodology; Section 4 reports and discusses experimental results; and Section 5 concludes the paper and provides possible future directions.

2 Background

This section reviews prior work on Arabic OCR for Arabic and OCR error correction.

2.1 Arabic OCR

The goal of OCR is to transform a document image into character-coded text. The usual process is to automatically segment a document image into character images in the proper reading order using image analysis heuristics, apply an automatic classifier to determine the character codes that most likely correspond to each character image, and then exploit sequential context (e.g., preceding and following characters and a list of possible words) to select the most likely character in each position. The character error rate can be influenced by reproduction quality (e.g., original documents are typically better than photocopies), the resolution at which a document was scanned, and any mismatch between the instances on which the character image classifier was trained and the rendering of the characters in the printed document. Arabic OCR presents several challenges, including:

• Arabic's cursive script in which most characters are connected and their shape vary with position in the word.

• The optional use of word elongations and ligatures, which are special forms of certain letter sequences.

• The presence of dots in 15 of the 28 letters to distinguish between different letters and the optional use of diacritic which can be confused with dirt, dust, and speckle (Darwish and Oard, 2002).

• The morphological complexity of Arabic, which results in an estimated 60 billion possible

Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pages 408–414,
Sydney, July 2006.

surface forms, complicates dictionary-based error correction. Arabic words are built from a closed set of about 10,000 root forms that typically contain 3 characters, although 4-character roots are not uncommon, and some 5-character roots do exist. Arabic stems are derived from these root forms by fitting the root letters into a small set of regular patterns, which sometimes includes addition of "infix" characters between two letters of the root (Ahmed, 2000).

There is a number of commercial Arabic OCR systems, with Sakhr's Automatic Reader and Shonut's Omni Page being perhaps the most widely used. Retrieval of OCR degraded text documents has been reported for many languages, including English (Harding et al., 1997), Chinese (Tseng and Oard, 2001), and Arabic (Darwish and Oard, 2002).

2.2 OCR Error Correction

Much research has been done to correct recognition errors in OCR-degraded collections. There are two main categories of determining how to correct these errors. They are word-level and passage-level post-OCR processing. Some of the kinds of word level post-processing include the use of dictionary lookup, probabilistic relaxation, character and word n-gram frequency analysis (Hong, 1995), and morphological analysis (Oflazer, 1996). Passage-level post-processing techniques include the use of word n-grams, word collocations, grammar, conceptual closeness, passage level word clustering, linguistic context, and visual context. The following introduces some of the error correction techniques.

• Dictionary Lookup: Dictionary Lookup, which is the basis for the correction reported in this paper, is used to compare recognized words with words in a term list (Church and Gale, 1991; Hong, 1995; Jurafsky and Martin, 2000). If a word is found in the dictionary, then it is considered correct. Otherwise, a checker attempts to find a dictionary word that might be the correct spelling of the misrecognized word.

Jurafsky and Martin (2000) illustrate the use of a noisy channel model to find the correct spelling of misspelled or misrecognized words. The model assumes that text errors are due to edit operations namely insertions, deletions, and substitutions. Given two words, the number of edit operations required to transform one of the words to the other is called the Levenshtein edit distance (Baeza-Yates and Navarro, 1996). To capture the probabilities associated with different edit operations, confusion matrices are employed. Another source of evidence is the relative probabilities that candidate word corrections would be observed. These probabilities can be obtained using word frequency in text corpus (Jurafsky and Martin, 2000). However, the dictionary lookup approach has the following problems (Hong, 1995):

a) A correctly recognized word might not be in the dictionary. This problem could surface if the dictionary is small, if the correct word is an acronym or a named entity that would not normally appear in a dictionary, or if the language being recognized is morphologically complex. In a morphological complex language such as Arabic, German, and Turkish the number of valid word surface forms is arbitrarily large which complicates building dictionaries for spell checking.

b) A word that is misrecognized is in the dictionary. An example of that is the recognition of the word "tear" instead of "fear". This problem is particularly acute in a language such as Arabic where a large fraction of three letters sequences are valid words.

• Character N-Grams: Character n-grams maybe used alone or in combination with dictionary lookup (Lu et al., 1999; Taghva et al., 1994). The premise for using n-grams is that some letter sequences are more common than others and other letter sequences are rare or impossible. For example, the trigram "xzx" is rare in the English language, while the trigram "ies" is common. Using this method, an unusual sequence of letters can point to the position of an error in a misrecognized word. This technique is employed by BBN's Arabic OCR system (Lu et al., 1999).

• Using Morphology: Many morphologically complex languages, such as Arabic, Swedish, Finnish, Turkish, and German, have enormous numbers of possible words. Accounting for and listing all the possible words is not feasible for purposes of error correction. Domeij proposed a method to build a spell checker that utilizes a stem lists and orthographic rules, which govern how a word is written, and morphotactic rules, which govern how morphemes (building blocks of meanings) are allowed to combine, to accept legal combinations of stems (Domeij et al., 1994). By breaking up compound words, dictionary lookup can be applied to individual constituent stems. Similar work was done for Turkish in which an error tolerant finite state

recognizer was employed (Oflazer, 1996). The finite state recognizer tolerated a maximum number of edit operations away from correctly spelled candidate words. This approach was initially developed to perform morphological analysis for Turkish and was extended to perform spelling correction. The techniques used for Swedish and Turkish can potentially be applied to Arabic. Much work has been done on Arabic morphology and can be potentially extended for spelling correction.

• Word Clustering: Another approach tries to cluster different spellings of a word based on a weighted Levenshtein edit distance. The insight is that an important word, specially acronyms and named-entities, are likely to appear more than once in a passage. Taghva described an English recognizer that identifies acronyms and named-entities, clusters them, and then treats the words in each cluster as one word (Taghva, 1994). Applying this technique for Arabic requires accounting for morphology, because prefixes or suffixes might be affixed to instances of named entities. DeRoeck introduced a clustering technique tolerant of Arabic's complex morphology (De Roeck and Al-Fares, 2000). Perhaps the technique can be modified to make it tolerant of errors.

• Using Grammar: In this approach, a passage containing spelling errors is parsed based on a language specific grammar. In a system described by Agirre (1998), an English grammar was used to parse sentences with spelling mistakes. Parsing such sentences gives clues to the expected part of speech of the word that should replace the misspelled word. Thus candidates produced by the spell checker can be filtered. Applying this technique to Arabic might prove challenging because the work on Arabic parsing has been very limited (Moussa et al., 2003).

• Word N-Grams (Language Modeling): A Word n-gram is a sequence of *n* consecutive words in text. The word n-gram technique is a flexible method that can be used to calculate the likelihood that a word sequence would appear (Tillenius, 1996). Using this method, the candidate correction of a misspelled word might be successfully picked. For example, in the sentence "I bought a peece of land," the possible corrections for the word peece might be "piece" and "peace". However, using the n-gram method will likely indicate that the word trigram "piece of land" is much more likely than the trigram "peace of land." Thus the word "piece" is a more likely correction than "peace".

3 Error Correction Methodology

This section describes the character level modeling, the language modeling, and shallow morphological analysis.

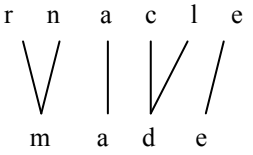
3.1 OCR Character Level Model

A noisy channel model was used to learn how OCR corrupts single characters or character segments, producing a character level confusion model. To train the model, 6,000 OCR corrupted words were obtained from a modern printing of a medieval religious Arabic book (called "The Provisions of the Return" or "Provisions" for short by *Ibn Al-Qayim*). The words were then manually corrected, and the corrupted and manually corrected versions were aligned. The Provisions book was scanned at 300x300 dots per inch (dpi), and Sakhr's Automatic Reader was used to OCR the scanned pages. From the 6,000 words, 4,000 were used for training and the remaining words were set aside for later testing. The Word Error Rate (WER) for the 2,000 testing words was 39%. For all words (in training and testing), the different forms of *alef* (*hamza*, *alef, alef maad*, *alef with hamza on top*, *hamza on w*a, *alef with hamza on the bottom*, and *hamza on ya*) were normalized to *alef*, and *ya* and *alef maqsoura* were normalized to *ya*. Subsequently, the characters in the aligned words can aligned in two different ways, namely: *1*:*1* (one-to-one) character alignment, where each character is mapped to no more than one character (Church and Gale, 1991); or using *m*:*n* alignment, where a character segment of length *m* is aligned to a character segment of length *n* (Brill and Moore, 2000). The second method is more general and potentially more accurate especially for Arabic where a character can be confused with as many as three or four characters. The following example highlights the difference between the *1*:*1* and the *m*:*n* alignment approaches. Given the training pair (rnacle, made):

1:*1* alignment :

r	n	a	c	l	e
|	|	|	|	|	|
m	ε	a	d	ε	e

m:*n* alignment:



For alignment, Levenstein dynamic programming minimum edit distance algorithm was used to produce *1*:*1* alignments. The algorithm computes the minimum number of edit operations required to transform one string into another. Given the output alignments of the algorithm, properly aligned characters (such as a → a and e → e) are used as anchors, ε's (null characters) are combined to misaligned adjacent characters producing *m*:*n* alignments, and ε's between correctly aligned characters are counted as deletions or insertions.

To formalize the error model, given a clean word $\chi = \#C_1..C_k..\ C_l..C_n\#$ and the resulting OCR degraded word $\delta = \#D_1..D_x..\ D_y..D_m\#$, where $D_x..\ D_y$ resulted from $C_k..\ C_l$, ε representing the null character, and # marking word boundaries, the probability estimates for the three edit operations for the models are:

$$P_{substitution}(C_k..C_l \text{ -> } D_x..\ D_y) = \frac{count(C_k..C_l \rightarrow D_x..D_y)}{count(C_k..C_l)}$$

$$P_{deletion}(C_k..C_l \text{ -> } \varepsilon) = \frac{count(C_k..C_l \rightarrow \varepsilon)}{count(C_k..C_l)}$$

$$P_{insertion}(\varepsilon \text{ -> } D_x..\ D_y) = \frac{count(\varepsilon \rightarrow D_x..D_y)}{count(C)}$$

When decoding a corrupted string δ composed of the characters $D_1..D_x..\ D_y..D_m$, the goal is to find a string χ composed of the characters $C_1..C_k..\ C_l..C_n$ such that $P(\delta|\chi)\cdot P(\chi)$ is maximum. $P(\chi)$ is the prior probability of observing χ in text and $P(\delta|\chi)$ is the probability of producing δ from χ.

$P(\chi)$ was computed from a web-mined collection of religious text by *Ibn Taymiya*, the main teacher of the medieval author of the "Provisions" book. The collection contained approximately 16 million words, with 278,877 unique surface forms.

$P(\delta|\chi)$ is calculated using the trained model, as follows:

$$P(\delta \mid \chi) = \prod_{all:D_x..D_y} P(D_x..D_y \mid C_k..C_l)$$

The segments $D_x..\ D_y$ are generated by finding all possible 2^{n-1} segmentations of the word δ. For example, given "macle" then all possible segmentations are (m,a,c,l,e), (ma,c,l,e), (m,ac,l,e), (mac,l,e), (m,a,cl,e), (ma,cl,e), (m,acl,e), (macl,e), (m,a,c,le), (ma,c,le), (m,ac,le), (mac,le), (m,a,cle), (ma,cle), (m,acle), (macle).

All segment sequences $C_k..\ C_l$ known to produce $D_x..\ D_y$ for each of the possible segmentations are produced. If a sequence of $C_1..\ C_n$ segments generates a valid word χ which exists in the web-mined collection, then $\text{argmax}_\chi\ P(\delta|\chi)\cdot P(\chi)$ is computed, otherwise the sequence is discarded. Possible corrections are subsequently ranked. For all the experiments reported in this paper, the top 10 corrections are generated. Note that error correction reported in this paper does not assume that a word is correct because it exists in the web-mined collection and assumes that all words are possibly incorrect.

The effect of two modifications to the *m*:*n* character model mentioned above were examined.

The first modification involved making the character model account for the position of letters in a word. The intuition for this model is that since Arabic letters change their shape based on their positions in words and would hence affect the letters with which they would be confused. Formally, given *L* denoting the positions of the letter at the boundaries of character segments, whether start, middle, end, or isolated, the character model would be:

$$P_{substitution}(C_k..C_l \text{ -> } D_x..\ D_y \mid L) = \frac{count(C_k..C_l \rightarrow D_x..D_y \mid L)}{count(C_k..C_l \mid L)}$$

$$P_{deletion}(C_k..C_l \text{ -> } \varepsilon \mid L) = \frac{count(C_k..C_l \rightarrow \varepsilon \mid L)}{count(C_k..C_l \mid L)}$$

$$P_{insertion}(\varepsilon \text{ -> } D_x..\ D_y) = \frac{count(\varepsilon \rightarrow D_x..D_y \mid L)}{count(C \mid L)}$$

The second modification involved giving a small uniform probability to single character substiutions that are unseen in the training data. This was done in accordance to Lidstone's law to smooth probabilities. The probability was set to be 100 times smaller than the probability of the smallest seen single character substitution*.

* Other uniform probability estimates were examined for the training data and the one reported here seemed to work best

3.2 Language Modeling

For language modeling, a trigram language model was trained on the same web-mined collection that was mentioned in the previous subsection without any kind of morphological processing. Like the text extracted from the "Provisions" book, *alef* and *ya* letter normalizations were performed. The language model was built using SRILM toolkit with Good-Turing smoothing and default backoff.

Given a corrupted word sequence $\Delta = \{\delta_1 .. \delta_i .. \delta_n\}$ and $\Xi = \{X_1 .. X_i .. X_n\}$, where $X_i = \{\chi_{i0} .. \chi_{im}\}$ are possible corrections of δ_i ($m = 10$ for all the experiments reported in the paper), the aim was to find a sequence $\Omega = \{\omega_1 .. \omega_i .. \omega_n\}$, where $\omega_i \in X_i$, that maximizes:

$$\underbrace{\left(\prod_{i=1..n, j=1..m} P\left(\chi_{ij} \mid \chi_{i-1,j}, \chi_{i-2,j}\right)\right)}_{LanguageModel} \cdot \underbrace{P(\delta_i \mid \chi_{ij})}_{CharacterModel}$$

3.3 Language Modeling and Shallow Morphological Analysis

Two paths were pursued to explore the combined effect of language modeling and shallow morphological analysis.

In the first, a 6-gram language model was trained on the same web-mined collection after each of the words in the collection was segmented into its constituent prefix, stem, and suffix (in this order) using language model based stemmer (Lee et al., 2003). For example, "وكتابهم – wktAbhm" was replaced by "w# ktAb +hm" where # and + were used to mark prefixes and suffixes respectively and to distinguish them from stems. Like before, *alef* and *ya* letter normalizations were performed and the language model was built using SRILM toolkit with the same parameters.

Formally, the only difference between this model and the one before is that $X_i = \{\chi_{i0} .. \chi_{im}\}$ are the {prefix, stem, suffix} tuples of the possible corrections of δ_i (a tuple is treated as a block). Otherwise everything else is identical.

In the second, a trigram language model was trained on the same collection after the language modeling based stemming was used on all the tokens in the collection (Lee et al., 2003). The top *n* generated corrections were subsequently stemmed and the stems were reranked using the language model. The top resulting stem was compared to the condition in which language modeling was used without morphological analysis (as in the previous subsection) and then the top resulting correction were stemmed. This path was pursued to examine the effect of correction on applications where stems are more useful than words such as Arabic information retrieval (Darwish et al., 2005; Larkey et al., 2002).

3.4 Testing the Models

The *1*:*1* and *m*:*n* character mapping models were tested while enabling or disabling character position training (CP), smoothing by the assignment of small probabilities to unseen single character substitutions (UP), language modeling (LM), and shallow morphological processing (SM) using the 6-gram model.

As mentioned earlier, all models were tested using sentences containing 2,000 words in total.

4 Experimental Results

Table 1 reports on the percentage of words for which a proper correction was found in the top *n* generated corrections using different models. The percentage of words for which a proper correction exists in the top 10 proposed correction is the upper limit accuracy we can achieve given than we can rerank the correction using language modeling. Table 2 reports the word error rate for the *1*:*1* and *m*:*n* models with and without CP, UP, LM, and SM. Further, the before and after stemming error rates are reported for setups that use language modeling. Table 3 reports on the stem error rate when using the stem trigram language model.

The best model was able to find the proper correction within the top 10 proposed correction for 90% of the words. The failure to find a proper correction within the proposed corrections was generally due to grossly misrecognized words and was rarely due to words that do not exist in web-mined collection. Perhaps, more training examples for the character based models would improve correction.

Corrections	1	2	3	4	5	10
1:1	75.3	80.3	83.1	84.5	85	86.5
1:1 + CP	76.9	82.1	83.5	83.2	85	86
1:1 + UP	76	81	83.6	84.6	85.2	86.7
m:*n*	78.3	83.5	85.4	86.7	87.1	88.5
m:*n* + CP	79.9	83.9	84.0	85.5	85.9	86.8
m:*n* + UP	78.4	83.7	85.6	84.1	87.0	90.0

Table 1: Percentage of words for which a proper correction was found in the top n generated corrections

Model	*1:1*		*m:n*	
	Word	Stem	Word	Stem
No Correction	39.0%	-	39.0%	-
Base Model	24.7%	-	21.8%	-
+ CP	23.1%	-	21.5%	-
+ UP	24%	-	21.6%	-
+ LM	**15.8%**	**14.6%**	**13.3%**	**12.1%**
+ LM + CP	16.5%	15.1%	15.5%	14.7%
+ LM + UP	**15.4%**	**14.3%**	**11.7%**	**10.8%**
+ SM + UP	27.8%	26.5%	24.5%	23.0%

Table 2: Word/stem error rate for correction with the different models

Model	*1:1*	*m:n*
Stem 3-gram	16.1%	12.9%

Table 3: Stem error rate for top correction using stem trigram language model

The results indicate that the *m*:*n* character model is better than the *1*:*1* model in two ways. The first is that the *m*:*n* model yielded a greater percentage of proper corrections in the top 10 generated corrections, and the second is that the scores of the top 10 corrections were better which led to better results compared to the *1*:*1* model when used in combination with language modeling. For the *m*:*n* model with language modeling, the language model properly picked the proper correction from the proposed correction 98% of the time (for the cases where a proper correction was within the proposed corrections).

Also the use of smoothing, UP, produced better corrections, while accounting for character positions had an adverse effect on correction. This might be an indication that the character segment correction training data was sparse. Using the 6-gram language model on the segmented words had a severely negative impact on correction accuracy. Perhaps is due to insufficient training data for the model. This situation lends itself to using a factored language model using the surface form of words as well as other linguistic features of the word such as part of speech tags, prefixes, and suffixes.

As for training a language model on words versus stems, the results suggest that word based correction is slightly better than stem based correction. The authors' intuition is that this resulted from having a sufficiently large corpus to train the language model and that this might have been reversed if the training corpus for the language model was smaller. Perhaps further investigation would prove or disprove the authors' intuition.

5 Conclusion and Future Work

The paper examined the use of single character and character segment models based correction of Arabic OCR text combined with language modeling and shallow morphological analysis. Further, character position and smoothing issues were also examined. The results show the superiority of the character segment based model compared to the single character based model. Further, the use of language modeling yielded improved error correction particularly for the character segment based model. Accounting for character position and shallow morphological analysis had a negative impact on correction, while smoothing had a positive impact. Lastly, given a large in-domain corpus to extract a correction dictionary and to train a language model is a sufficient strategy for correcting a morphologically rich language such as Arabic with a 70% reduction in word error rate.

For future work, a factor language model might prove beneficial to incorporate morphological information and other factors such as part of speech tags while overcoming training data sparseness problems. Also, determining the size of a sufficiently large corpus to generate a correction dictionary and to train a language model is desirable. Finally, word prediction might prove useful for cases where OCR grossly misrecognized words.

Reference

Agirre, E., K. Gojenola, K. Sarasola, and A. Voutilainen. Towards a Single Proposal in Spelling Correction. *In COLING-ACL'98* (1998).

Ahmed, M. A Large-Scale Computational Processor of Arabic Morphology and Applications. *MSc. Thesis, in Faculty of Engineering Cairo University: Cairo, Egypt.* (2000).

Baeza-Yates, R. and G. Navarro. A Faster Algorithm for Approximate String Matching. *In Combinatorial Pattern Matching (CPM'96), Springer-Verlag LNCS* (1996).

Brill, E. and R. Moore. An improved error model for noisy channel spelling correction. *In the proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286 – 293 (2000).

Church, K. and W. Gale. "Probability Scoringfor Spelling Correction." *Statistics and Computing*, 1: 93-103 (1991).

Darwish, K. and D. Oard. Term Selection for Searching Printed Arabic. *In SIGIR-2002* (2002).

Darwish, K., H. Hassan, and O. Emam. Examining the Effect of Improved Context Sensitive Morphology on Arabic Information Retrieval. *In ACL Workshop on Computation Approaches to Semitic Languages*, Ann Arbor, (2005).

De Roeck, A. and W. Al-Fares. A Morphologically Sensitive Clustering Algorithm for Identifying Arabic Roots. *In the 38th Annual Meeting of the ACL*, Hong Kong, (2000).

Domeij, R., J. Hollman, V. Kann. Detection of spelling errors in Swedish not using a word list en clair. *Journal of Quantitative Linguistics* (1994) 195-201.

Harding, S., W. Croft, and C. Weir. Probabilistic Retrieval of OCR-degraded Text Using N-Grams. *In European Conference on Digital Libraries* (1997).

Hong, T. Degraded Text Recognition Using Visual and Linguistic Context. *Ph.D. Thesis, Computer Science Department, SUNY Buffalo: Buffalo* (1995).

Jurafsky, D. and J. Martin. Speech and Language Processing. Chapter 5: pages 141-163. *Prentice Hall* (2000).

Larkey, L., L. Ballesteros, and M. Connell. Improving stemming for Arabic information retrieval: light stemming and cooccurrence analysis. *In proceedings of the 25th annual international ACM SIGIR conference*, pages 275-282 (2002).

Lee, Y., K. Papineni, S. Roukos, O. Emam, and H. Hassan. Language Model Based Arabic Word Segmentation. *In the Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 399 - 406 (2003).

Lu, Z., I. Bazzi, A. Kornai, J. Makhoul, P. Natarajan, and R. Schwartz. A Robust, Language-Independent OCR System. *In the 27th AIPR Workshop: Advances in Computer Assisted Recognition, SPIE* (1999).

Moussa B., M. Maamouri, H. Jin, A. Bies, X. Ma. Arabic Treebank: Part 1 - 10Kword English Translation. *Linguistic Data Consortium* (2003).

Oflazer, K. Error-Tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction. *Computational Linguistics 22(1)*, 73-90 (1996).

Taghva, K., J. Borsack, and A. Condit. An Expert System for Automatically Correcting OCR Output. *In SPIE - Document Recognition* (1994).

Tillenius, M., Efficient generation and ranking of spelling error corrections. *NADA* (1996).

Tseng, Y. and D. Oard. Document Image Retrieval Techniques for Chinese. *In Symposium on Document Image Understanding Technology, Columbia, MD* (2001).