A Modified Character-Level Deciphering Algorithm for OCR in Degraded Documents

Chi Fang and Jonathan J. Hull*

Center of Excellence for Document Analysis and Recognition
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260
chifang@cs.buffalo.edu

*RICOH California Research Center
2882 Sand Hill Road, Suite 115
Menlo Park, CA 94025
hull@crc.ricoh.com

ABSTRACT

Modifications to a previous character-level deciphering algorithm for OCR are presented in this paper that are able to handle touching characters and are tolerant to mistakes made at the clustering stage. The objective of a character-level deciphering algorithm is to assign alphabetic identities to character patterns such that the character repetition pattern in an input text matches the letter repetition pattern provided by a language model. Degradation in document images usually causes the occurrence of touching characters and mistakes in clustering the character patterns, which pose difficulties for character-level deciphering algorithms. The modifications proposed in this paper tightly integrate visual constraints from characters and touching patterns with constraints from a language model. This solves the problem of touching characters and reverses clustering mistakes. The provides a deciphering algorithm with robust performance under image degradation.

Key Words: deciphering algorithm, language constraints, substitution cipher, degraded document recognition, touching characters, clustering, OCR.

1 Introduction

Character-level deciphering algorithms have been proposed for OCR ([1, 2, 3, 4, 5]). These techniques recognize character images in a printed text by solving a substitution cipher. A clustering step first converts each character on the input text page into a computer readable code in such a manner that every shape corresponds to a distinct code. A deciphering algorithm

is then applied that uses language statistics to assign alphabetic labels to the cipher codes so that the character repetition pattern in the input text passage best matches the letter repetition pattern provided by a language model.

A technique that is representative of the many proposed character-level deciphering algorithms is that presented by Nagy ([3]). This algorithm solves a substitution cipher by extending a tentative assignment of letters to character patterns according to the degree of match between the decrypted portion of the ciphertext and a vocabulary of common words.

Image degradation in an input document causes two problems that are difficult for current deciphering algorithms to solve. One problem is the occurrence of touching characters. This invalidates the assumption that character repetition patterns in the input text match that of a language model. The other problem is that mistakes usually appear in the clustering stage. Either visually similar but different characters are included in a single cluster or identical characters are included in different clusters.

Previous deciphering algorithms can handle clustering mistakes that split identical characters into different clusters. But it is difficult for them to recognize touching characters and to correct clustering mistakes that group different characters into the same cluster. Also, previous methods are solely based on using dictionary constraints. No attempt has been made to use visual constraints above the level of isolated words to improve deciphering.

Modifications to a character-level deciphering algorithm are presented in this paper that address some of the above problems. Starting with the algorithm NONMATCHES presented by Nagy in [3], the proposed modifications integrate constraints from visual patterns extracted across the text page and constraints provided by a language model . This allows the algorithm to decode characters and touching patterns. Short touching patterns are solved by matching each cipher word against the dictionary with the assumption that it might include touching patterns. The long touching patterns are handled at the end of the deciphering process when most of the single characters and short touching patterns have already been deciphered. The font information from these patterns are used to help identify the long touching patterns by doing partial pattern matching.

The modifications proposed here result in an algorithm that is tolerant to both kinds of clustering mistakes mentioned above, and is able to decipher touching characters. The algorithm can detect and reverse incorrect assignments of characters or touching patterns made earlier in the deciphering process.

The rest of the paper contains three sections. Section 2 discusses the deciphering algorithm and how it solves the problem of touching characters and handles clustering mistakes. Section 3 presents and discusses the experimental results of applying this deciphering algorithm to different types of degraded documents. Finally, conclusions and some future directions are pointed out in section 4.

2 Algorithm

2.1 Touching Characters

Touching characters usually belong to two categories, as exemplified in the text passage shown in Figure 1: those that are caused by regular noise or by font specificities, such as ligatures (e.g., "ff", "fl", etc.), and those that are caused by image degradation, such as the touching patterns in the words "perfect" and "parts" in Figure 1. Touching patterns in the first category are usually short, involving two or three characters. Touching patterns in the second category are usually long and irregular, and are seldom repeated elsewhere in the text.

Short touching patterns are solved together with the single characters in the deciphering process. A potential short touching pattern is detected and labeled as such if it has larger than the usual aspect ratio and a small cluster size. A preliminary estimate N that denotes the number of characters involved in a potential touching pattern is derived by comparing the physical width of this touching pattern with the average character width. The cipherword that includes this potential touching pattern is then matched against the words in the dictionary under the assumption that $M \in [N - l, N + l]$ characters are involved in this touching pattern. This provides tolerance to mistakes made in the preliminary estimation of N.

A current cipher word is then matched against all the words in the dictionary using all such "wild cards." The match results in a set of potential assignments for the touching patterns. The optimal assignment is chosen using *cover set* constraints ([3]) and character bigram statistics.

The deciphered single characters and short touching patterns are then used to build up a partial font base as the deciphering proceeds This consists of image prototypes for deciphered characters and short touching patterns. This information will be used to decipher the long touching patterns.

Long touching patterns cannot be deciphered the same way as short touching patterns for two reasons. First, in a cipher word that includes irregular long touching patterns there usually are very few deciphered single characters to provide constraints for the dictionary match. This usually results in a great number of potential matches. Also, there are usually no constraints in the cover set to resolve the many potential matches because irregular long touching patterns are not likely to be repeated elsewhere in the text page. For these reasons, long touching patterns are processed at the end of deciphering when most of the single characters and short touching patterns have been deciphered and their font information is available. A recursive partial pattern matching process is applied to each long touching pattern to decide its identity based on the learned font information. The matching results are then passed to a post-processing stage where dictionary and character bigram statistics are used to make the final decision.

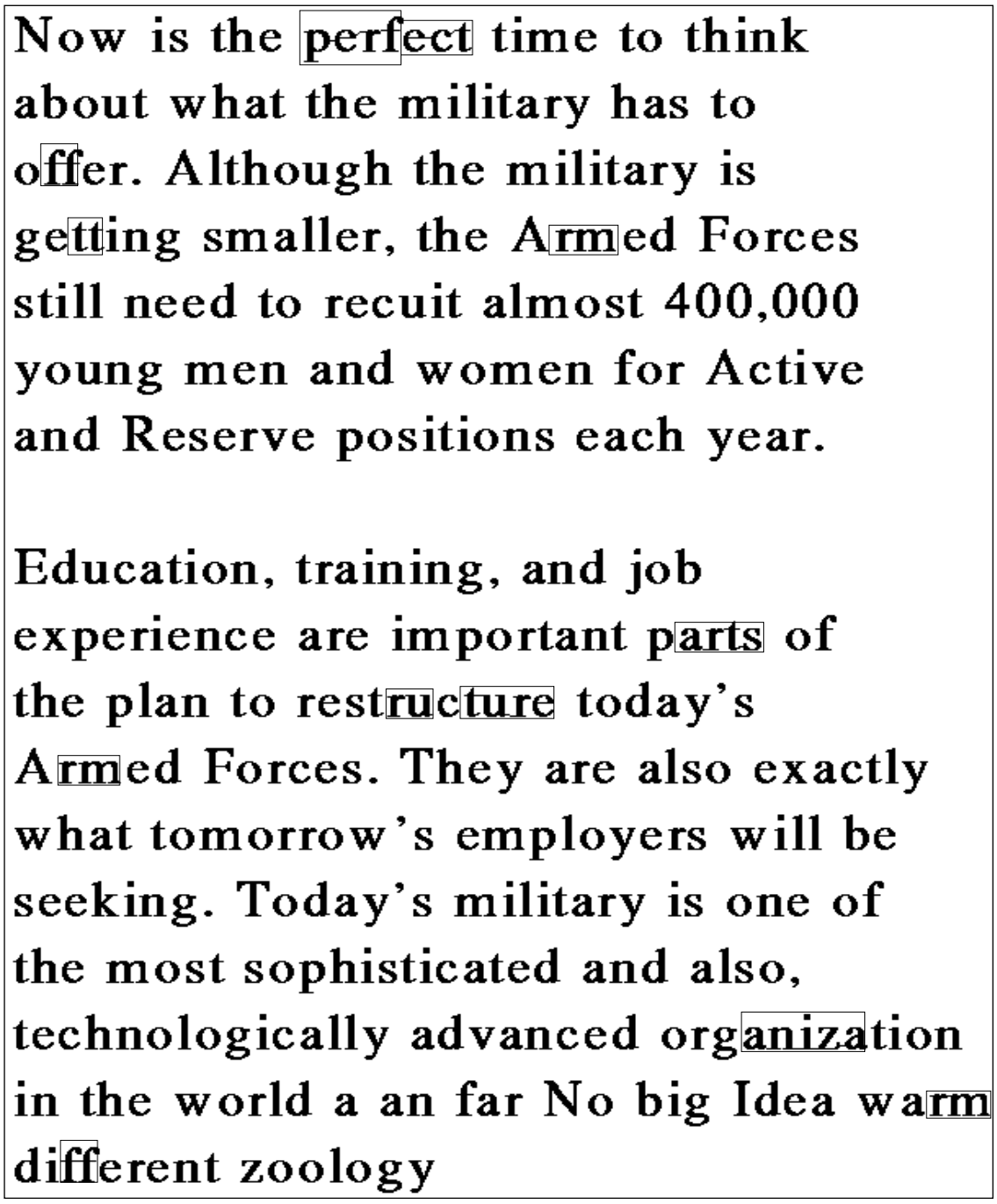

Figure 1: Touching Characters in Degraded Text Document

2.2 Clustering Mistakes

The following discussion is concerned with clustering mistakes that group different characters or touching patterns into the same cluster.

The letter identity of a cluster is determined when a cipher word that includes one of the cluster's members as its undecided pattern is deciphered. This causes some members to be assigned wrong letter identities if the cluster includes different characters. This, in turn, will result in cipher words for which no match can be found in the dictionary because they include these incorrectly deciphered characters or touching patterns. This provides us with a clue to detect the incorrectly deciphered characters or touching patterns and then trace back to the cluster where the problem originated.

For the majority of common font types, there appears to exist a relatively stable relationship in that certain character patterns closely resemble certain other character pattern(s), and thus are very likely to cause clustering mistakes. Examples are 'e' and 'c', 'b' and 'h', 'u' and 'n', and, 'l', 'f' and 'I'. Of course, some set will be easier to differentiate when the characters are not degraded and a certain special feature representation is used for the clustering. This heuristic of character confusion sets is used for helping the algorithm to detect and reverse clustering mistakes. During the deciphering process, in case a cipher word does not match any words in the dictionary, we check each of its deciphered characters to see whether it is in one of the confusion sets. If it is, the cipher word is matched with the dictionary again with the confusing character being assigned different identities in the confusion set. Multiple matches are resolved with constraints from the cover set or from the learned font information. The optimal assignment is used to reverse mistakes in the character assignment and character clustering.

A description of this deciphering algorithm is shown in Figure 2. The original outline is borrowed from the algorithm NONMATCHES originally presented by Nagy in [3]. The portions of step 3 that use a confusion matrix to reverse the identity of confusing characters before rematching against the dictionary as well as the decomposition of long touching patterns by visual pattern matching are new. Also, the maintenance of a learned document-specific font database that contains examples of previously deciphered characters for use in visual pattern matching and verification of potential assignments is novel.

3 Experimental Results And Discussion

Different types of degraded documents have been used to test the algorithm. The document image in Figure 1 is an artificially created document image that contains many occurrences of long touching patterns. It consists of 90 words that include 12 touching patterns, 5 of them being relatively long touching patterns. There is only one single case of a clustering mistake where 'E' is included with the two 'F's.

The document image in Figure 3 is a degraded scanned document. It was first generated from the same text passage as in Figure 1. The image was then printed on a laser printer

Initialize partial assignment *PA* to be empty;
Begin
Repeat
1. Find the most frequent undecided cipher word *CW* and its associated *Cover Set*, long touching patterns postponed to process later;
2. Match the selected *CW* against dictionary, undecided character or touching pattern act as a continuum of number of wildcards;
3. **if** no dictionary match found
then
re-assign identities of confusing characters,
re-match the *CW* against dictionary;
else if including long touching pattern
then
decipher long touching pattern by visual pattern matching using learned partial font base;
else if *Cover Set* not empty
then
choose the assignment that results in the least number of un-matched words in the *Cover Set*;
else
verify potential assignments with learned font information;
4. Update partial assignment *PA*;
5. Update font base;
Until partial assignment *PA* is complete
End

Figure 2: Outline of Deciphering Algorithm

and re-scanned with 300 pixel-per-inch (ppi) resolution. The final binary image is the result of thresholding the scanned gray-level image by choosing a single fixed threshold that produces many distorted and touching characters. This page contains 42 touching patterns and many clustering mistakes, mainly between 'e' and 'c', 'b' and 'h', 'u' and 'n', and, 'l', 'f' and 'I'.

The language model for the deciphering algorithm consists of a dictionary and a character bigram statistics database. The dictionary contained all the 53121 English words from the Brown Corpus. The character bigram statistics database was also compiled from the Brown Corpus.

For the artificially generated document in Figure 1, all the single characters and touching patterns were correctly deciphered. Three of the long touching patterns ('perf' and 'ect' in "perfect, and 'arts' in "parts") were solved by using learned font information and the other two long touching patterns were solved by constraints from within the cipher words themselves. The

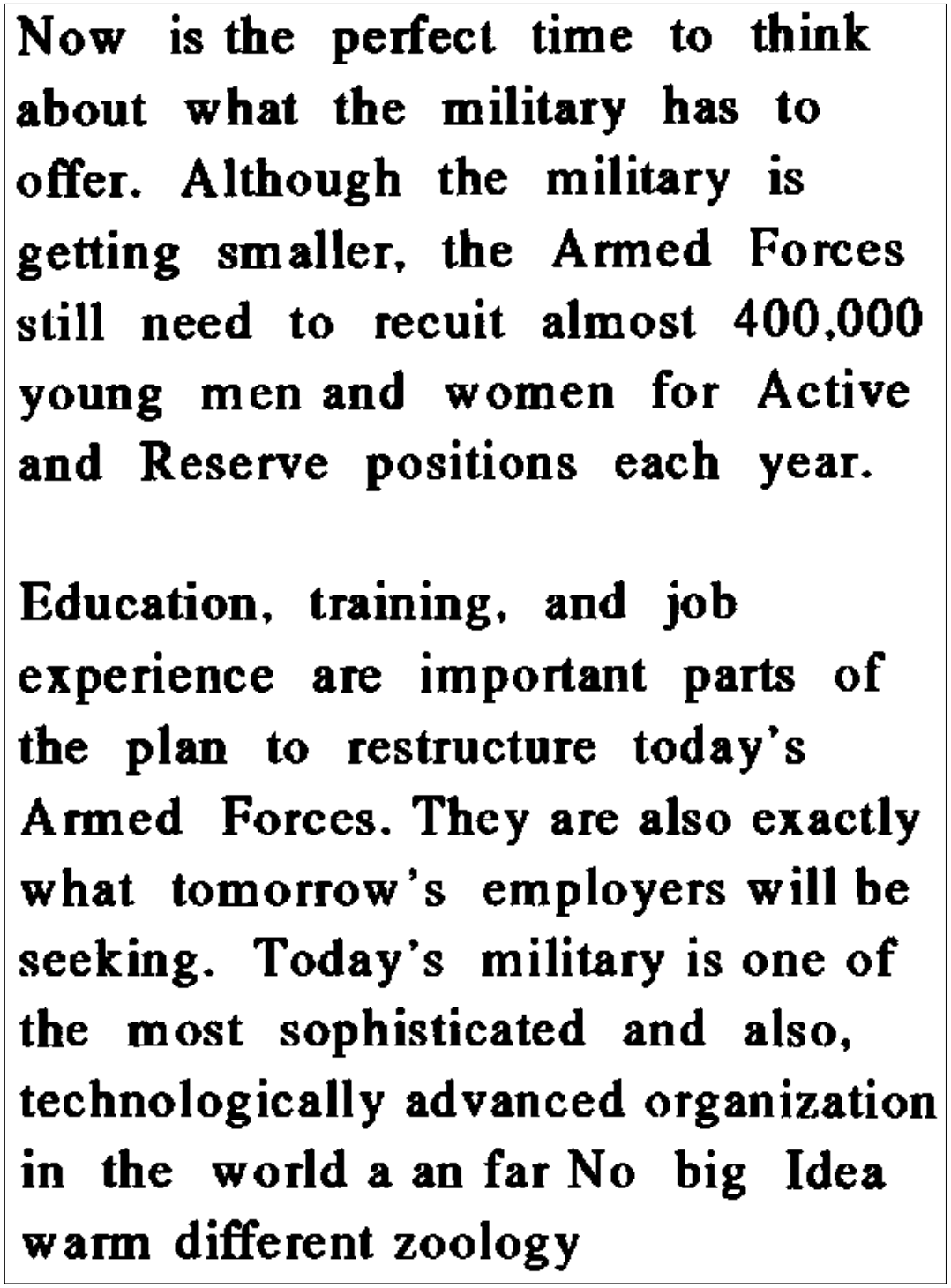

Figure 3: Scanned Document Image with Touching Characters

only clustering mistake caused the 'F's to be first assigned to 'E' and the mistake being reversed later in the deciphering process.

For the scanned image, only three words were not deciphered correctly. One of them is the word "training", which includes one deciphered touching pattern 'ra', and two undeciphered touching patterns 'in'. Because of clustering mistakes, 'ra' was put in the cluster with many 'm's and assigned an 'm' when the cluster was deciphered. Because this rare case of confusing patterns was not covered by the character confusion set heuristics we used, the mistake was not detected and reversed. A similar situation happens with another unsolved word "Forces"(its first occurrence in Figure 3), where the deciphered touching pattern 'rc' is put in the same cluster as 're', and the combination makes the 'F' undecipherable. The third unsolved word is the number "400,000" which does not appear in the dictionary. Except for these three words, all the other words and the other touching patterns are correctly deciphered. Also,all the other clustering mistakes detected and corrected. The algorithm takes about 2.5 minutes to decipher the scanned image on SUN SPARC station.

4 Conclusions and Future Directions

We presented a modified character-level deciphering algorithm for OCR that has the ability to solve touching characters caused by document degradation, and is tolerant to mistakes in the clustering algorithm. Visual constraints are systematically combined with language constraints to decipher touching patterns and to detect and reverse clustering errors. The deciphering algorithm was tested on both artificially created and scanned degraded documents with extensive occurrences of touching patterns and clustering mistakes, and achieved satisfactory results in both cases.

Despite efforts in improving the performance of a clustering algorithm, clustering mistakes in the presence of image degradation are an impediment to results in deciphering algorithms. More extensive heuristics for confused characters as well as touching patterns are being investigated to help detect and correct clustering mistakes and to improve the performance of the deciphering algorithm.

References

[1] R. Casey and G. Nagy, "Autonomous reading machine," *IEEE Trans. Comput.*,vol. C-7, May 1968

[2] R. Casey, "Text OCR by solving a cryptogram," *Proc. ICPR-8*, Paris, 1986, pp. 349-351

[3] G. Nagy, "Efficient algorithms to decode substitution cipher with application to OCR," *Proc. ICPR-8*, Paris, 1986, pp. 352-355

[4] G. Nagy, S. Seth and K. Einspahr, "Decoding substitution cipher by means of word matching with application to OCR," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, Sept. 1987

[5] S. Peleg and A. Rosenfeld, "Breaking substitution ciphers using a Relaxation Algorithm", *CACM*, vol. 22, no. 11, Nov. 1979, pp. 598-605