Exploration of Contextual Constraints for Character Pre-Classification

Tin Kam Ho
Bell Labs, Lucent Technologies
tkh@bell-labs.com

George Nagy
Rensselaer Polytechnic Institute
nagy@ecse.rpi.edu

Abstract

We present strategies and results for identifying the symbol type (lower-case, upper-case, digit, and punctuation or special symbols) of every character in a text document by using various kinds of information from neighboring characters. In the expectation of reasonable word and character segmentation for shape clustering, we designed several type recognition methods that depend on cluster n-grams, shape codes, and within-word context. On an ASCII test corpus of 925 articles that simulates perfect image-level processing, these methods achieve a substantial improvement over default assignment of all characters to lower case.

1. Introduction

OCR is usually based on “universal” concepts of the generic shape of printed symbols, combined with linguistic context. In document-specific systems, where every occurrence of a given symbol has the same shape, language context can be exploited to label bitmaps whose shape does not conform to preconception.

The following aspects of printed text have been exploited to facilitate recognition.

1. Consistency of the shape of individual characters within a document.

2. Font-independent characteristics, such as ascenders, descenders, relative size and vertical position.

3. Ease of character and word-level segmentation in many documents.

4. Stability of symbol-n-gram frequencies across documents.

5. Prevalence of lexicon words, which constitute a small fraction of all possible combinations of symbols.

6. Partial recognition of the text by an omnifont recognition engine.

In most previous work - selectively cited below - such information has been used for character or word recognition. In contrast, here we propose to use the information in a preprocessor. Recognition is much easier if the symbol shapes can be pre-classified into character-types, i.e., lower-case, upper-case, digit, and punctuation or special symbols.

OCR based on context alone has already been demonstrated when the text is segmented at the word and character level, *restricted to a single type-case, and all digits, special symbols and punctuation are suppressed.* The clusters can then be mapped to the alphabet either by considering the frequencies of N-grams of cluster-labels [1] or by enforcing the lexical constraints contained in a small (about 500 words) lexicon [8]. A more recent, vastly superior method for solving very short (but perfectly encoded) substitution ciphers is [2].

If there is some (imperfect) shape information already available, then at least two different “bootstrapping” methods, based on linguistic context in the form of commonly occurring ”stop” words, have been explored. Hull conjectured that word shapes are more robust than character shapes. Therefore if the stop words can be identified from their shape and occurrence frequency, the shapes of their constituent letters can be used as prototypes for training a shape-based classifier for most of the alphabetic classes [6]. Spitz constructed an OCR system based on a specially-constructed lexicon and character shape codes [10]. Ho demonstrated that the stop words can be reliably extracted using no other information than the lengths of all word triplets in the text [4]. In [5] we showed that a great portion of symbol identities can be recovered using only clustering information, with no prior knowledge of symbol shape.

All of these methods work best on single-case alphabetic text without extraneous symbols. It is therefore desirable to partition the bitmaps into character types, so that the most appropriate method for each type can be applied to discover the character codes of clusters of that type. At a minimum, the methods cited above will yield the lower-case alphabetic portion of the text, which may suffice for some information-retrieval tasks. They may also be used with existing recognition engines to resolve inter-type ambiguities (Z-2, l-1, !-I, etc.)

The remaining difficulty is imperfect clustering, due either to shape confusions due to noise, or to character-level segmentation errors that give rise to clusters of fragmented or conjoined characters. However, the word-based contextual methods are still applicable as long as none of the clusters contain bitmaps from several classes. Pure clusters can be guaranteed by setting the similarity threshold in the clustering algorithm high enough. This will, of course, result in multiple clusters for many symbols, and also in non-conforming clusters of fragmented and joined characters. Fortunately, the lexicon-based substitution-cipher decoding algorithms can still assign alphabetic labels to all the clusters which contain at least one member that appears in a valid lexicon word.

We now study the conjecture that the context provided by various information from an unknown character's neighbors can be used to map it into a character-type code, given that the document is in English, of a single typeface, and that the segmentation into words and characters is reliable. The notion of a "context window" has already proved useful for restoring diacritic marks lost in e-mail or transcription [9][11]. Our method differs from these only in depending on arbitrary class labels, instead of known symbol identity, in the context window.

If the shapes of the symbols are indeed not ambiguous within a single document, and the difference between the bitmaps corresponding to different symbols is greater than the differences (due to "noise") between bitmaps corresponding to the same symbol, then it is possible to cluster the patterns and to give each cluster, and therefore every symbol, an arbitrary label. We use an ASCII database to represent this situation. Our first method uses the n-gram statistics of these labels. If some crude knowledge is available about the character shapes, such as their heights and locations relative to the baseline, then the joint occurrences of such shape descriptors within a context window may reveal something about the symbol types. Our second method explores the usefulness of coarse character shape codes in context. Finally, we study the possibility of recovering the symbol type of characters rejected by an omnifont OCR, given that the neighboring characters are reliably recognized for their identities or only pre-classified to their symbol types.

2. Experimental Setup

Our test database is Reuters-21578 (Distribution 1.0), a collection of news articles from Reuters. Specifically, we used the first section of the database (reut2-000.sgm) that consists of 925 articles, ranging from 14 words to 830 words (only 137 words on average), or 70 to 5219 characters (854 characters on average). The majority of the articles (686 of 925) have less than 1000 characters, and there is considerable variation among them from an OCR perspective (many acronyms and proper nouns, numerical tables, financial data).

All articles were pre-processed by removing the last two words which are the signature of the agency. The first 425 articles were used as the training set and the remaining 500 as the test set. There are no obvious differences in the type-statistics between the training and testing articles, but there are large differences between individual articles.

In each experiment, our goal is to (pre-)classify each character found on a text page into one of four types: Lower case (l), Upper case (u), Digit (d), or Punctuation (p). Punctuation includes special symbols such as $, /, *, %. Any number of consecutive spaces or line breaks is considered as a single character of a known type (Space, b). To simplify the following descriptions, we use the word *symbol* to refer to a character identify class (such as 'A', 'a'), the word *type* to refer to its type among the set {l, u, d, p, b}, or {LOWER, UPPER, DIGIT, PUNCT, SPACE}, and we call a particular instance of a symbol or type a *character*. Most of the articles are dominated by lower case characters, but there are exceptions. Table 1 shows variations in the size (total number of characters and words) and the proportion of each symbol type across articles in the database. The percentage of lower case letters shows the accuracy of a default assignment of all characters to the type LOWER.

3. Selective N-gram and Within-Word Context

We began by studying the n-gram statistics of the types calculated from each training article. Assigning one distinct label to each ASCII symbol, we calculate the bigram frequencies of all label pairs (including space) in each document. We observed that:

- PUNCTs are far more likely to be followed by SPACE than preceded by it,
- UPPERs are far more likely to be preceded by SPACE than followed by it,
- DIGITs usually occur at least once next to SPACE, and
- DIGITs are far more likely to be next to SPACE than to any LOWER.

Based on these observations, we designed two heuristic features that are ratios of relevant bigram frequencies and attempted an iterative procedure to assign the character types. This simple procedure appeared to be successful in separating most of the UPPERs and PUNCTs from the rest. However, it failed almost entirely on DIGITs, and they were trapped in the initial (default) assignment LOWER. Therefore we concluded that further discriminating features and rules were necessary. This

Table 1. Size and proportion of characters in each type.

	training set					test set				
	Overall	Min	Max	Mean	Stddev	Overall	Min	Max	Mean	Stddev
#Characters	334,353	38	4280	786.7	785.0	298,160	37	4035	596.3	611.1
%LOWER	88.9	28.8	97.4	85.8	10.6	88.7	30.6	97.1	83.7	13.3
%UPPER	3.9	0.7	43.3	4.5	2.9	4.1	1.1	17.4	5.0	2.9
%DIGIT	3.7	0.0	49.0	6.0	8.1	3.8	0.0	51.7	7.5	11.0
%PUNCT	3.5	0.0	25.0	3.7	2.2	3.4	0.0	12.9	3.8	2.2
#Words	66,254	12	828	155.9	152.8	59,303	12	761	118.6	118.2

preliminary study led us to the more elaborate method described next.

Using the same labels that represent symbols of unknown identities, we calculated the n-gram frequencies of all symbols, symbol pairs and triplets observed in the document. For each symbol, we also observed how often it appears in the document, its position (averaged over all members of the corresponding cluster) in the words containing it, and the average length of such words. Based on these we calculated 11 numerical features to describe each symbol cluster. Each feature is a single frequency estimated from a whole document. The definition of these features, together with their Fisher's discriminant ratio for the 6 pairs of classes (measured using the training set as a single, long document) are shown in Table 2, where N(.) is the count, CharCount and WordCount are the document length in terms of character and word counts. An example calculation of the features for the symbol 'a' in a one-sentence document is shown below.

```
<Example document>
Isaac’s papa saw a large aardvark.

<CharCount for document> = 29
<WordCount for document> = 6
<Features for the symbol ‘a’>
f1: 10/29 = 0.3448   f2: 2/10 = 0.2
f3:  0/10 = 0.0      f4: 2/10 = 0.2
f5:  1/10 = 0.1      f6: 2/10 = 0.2
f7:  0/10 = 0.0      f8: 1/10 = 0.1
f9: (2/7+3/7+1/4+3/4+1/3+0/1+
     1/5+0/9+1/9+5/9)/10 = 0.2914
f10: (7+7+4+4+3+1+5+9+9+9)/10 = 5.8
f11: 10/6 = 1.6667
```

From these measures it appears that the bigram frequency from SPACE to character or Position-in-Word are useful for separating UPPERs from the rest. There is at least one good feature to separate any character type from any other type; all but p,d have at least two discriminating features. Other effects are less obvious, and there may be joint effects among the features that cannot be revealed by these single-feature Fisher ratios. Therefore we proceeded to build a statistical classifier using these features.

These vectors of 11 elements were standardized and then used in a nearest neighbor classifier using Euclidean distance. The classes are the four types {l,u,d,p}. A key question in this method is how performance is affected by deviations of test article n-gram statistics from those estimated with large corpora, especially when the test articles are very short. To study this effect, we trained two separate classifiers. Classifier 1 used the training set as if it were a very long document, i.e., all features were calculated using one single set of bigrams and trigrams, and there are 78 reference (training) vectors corresponding to the 78 observed symbols. Classifier 2 used the training set as separate articles, i.e., bigram and trigram frequencies were calculated on a per-article basis, so there were 18604 reference (training) vectors (many symbols did not occur in each article). Both classifiers were tested on each test article as well as the entire test set treated as one long article.

Table 3 summarizes the results using each classifier in both testing setups. With the test set as one single, long article of 298K words, the overall correct rate using Classifier 1 is 99.96%, (298042 / 298160), Classifier 2 is 93.34% (278305 / 298160). Classifier 1 is thus far more accurate, and both classifiers are far better than the default assignment of every cluster to the type LOWER (accounting for 88.72% of all characters). Classifier 1 makes only 2 errors (assigning 'X' to LOWER and '(' to UPPER). Classifier 2 makes more diverse errors and the most common type is the assignment of UPPERs {B,C,D,E,H,M,O,W} to LOWER.

Arguably the n-gram statistics are good with such a long test text, so the more interesting results are those for individual test articles. As expected, Classifier 1 did not perform very well with such noisy estimates, but Classifier 2 is substantially better. The per-article results are shown in Figure 1. For 449 of the 500 test articles (89.8%), Classifier 2 is better than the default assignment. The eleven-feature vector of an unknown symbol in a test article is often similar to the feature vector representing that symbol in one of the documents in the reference set. The improvement is most obvious for shorter articles, especially those with only a few hundred characters. With Classifier 1, the increase in error rate over using the entire test set as a single article shows

Table 2. Features used for type discrimination.

		Normalization	Fisher's discriminant ratio					
f	Description	factor	d,l	d,u	p,d	p,l	p,u	u,l
1	Frequency N(x)	CharCount	1.11	1.35	0.13	1.20	0.03	1.28
2	Bigram Diagonal N(xx)	N(x)	0.26	0.42	0.36	0.05	0.01	0.19
3	Trigram Diagonal N(xxx)	N(x)	0.12	0.12	0.12	0.10	0.00	0.11
4	Beginning of Word N(bx)	N(x)	0.07	3.39	0.04	0.00	2.19	5.87
5	Beginning of Word N(bxx)	N(x)	2.66	1.71	0.03	0.07	0.04	0.29
6	End of Word N(xb)	N(x)	0.17	2.34	0.04	0.15	0.55	0.40
7	End of Word N(xxb)	N(x)	0.24	0.27	0.14	0.02	0.04	0.01
8	Singlet Word N(bxb)	N(x)	0.00	0.11	0.04	0.02	0.03	0.09
9	Position-in-Word(x)	N(x),length(w)	0.00	2.56	0.34	0.30	2.17	2.59
10	WordLength(x)	N(x)	3.26	2.08	2.16	0.46	0.95	0.36
11	WordCount(x)	WordCount	1.11	1.37	0.13	1.21	0.03	1.28

Table 3. Type classification accuracies compared to default assignment.

Classifier	Test case	#Characters	% Correct	% LOWER
1	entire test set	298,160	99.96%	88.72%
2	entire test set	298,160	93.34%	88.72%
		(unweighted average over the set of articles)		
1	individual article	596	85.53%	83.71%
2	individual article	596	90.06%	83.71%

the effect of noisy n-gram statistics in very short articles. Often the n-gram statistics must be estimated from only one or two instances of a symbol! However, we can see that if the classifier (such as Classifier 2) is built using the same kind of noisy estimates, it is still possible to have fairly reliable type identification.

4. Type Identification in a Context Window

To what extent is the symbol type determined by its neighbors? In this section, we examine this question under alternative assumptions.

4.1. Coarse Estimates of Character Height and Vertical Location

Instead of shape clustering, we now consider the utility of reliable, coarse measures of character height and vertical location relative to baseline. Using a broad categorization of all characters into three shape categories {ascender, x-height, descender}, we encoded each character with a pair of numbers according to its category. Ascender characters were encoded with (2,0), x-height characters (1,0), and descender characters (1,-1). Space was encoded as (0,0). We refer to these as *height codes*, because they are locations of the upper and lower bounds of the character relative to baseline, measured in units of x-height. To include context information, we used a sliding window that is centered at the unknown character and includes three neighboring characters from each side. Recall that all contiguous white spaces were collapsed into one single SPACE character, and line breaks were ignored. Thus each character was associated with a feature vector of 14 elements which are the height codes of itself and three neighbors on each side. For example, the letter 'u' in the word 'drought' has the feature vector (2,0,1,0,1,0,1,0,1,-1,2,0,2,0). The position of the character relative to the word containing it is implicitly described by the context window.

The feature vectors computed from the training set were used to train a decision forest classifier [3] for the four symbol types. On the test set, the classifier scored a correct rate of 89.18%.

4.2. Recovering Symbol Types of OCR Rejects

Next, we considered the case when an OCR engine has reliably recognized most of the characters on the page, and rejected the rest. We wanted to see how well the type of those rejected characters can be identified if only the types of its six neighbors are known. To investigate this, we numerically coded the type of each character, and trained a decision forest classifier on symbol type using these codes. This classifier scored 94.6% correct over all characters in the test set.

Then we examined the situation if the identities of a character's six neighbors are known. For this we numerically coded the identity of each character, and trained another decision forest classifier on symbol type using those identity codes. The classifier scored a correct rate

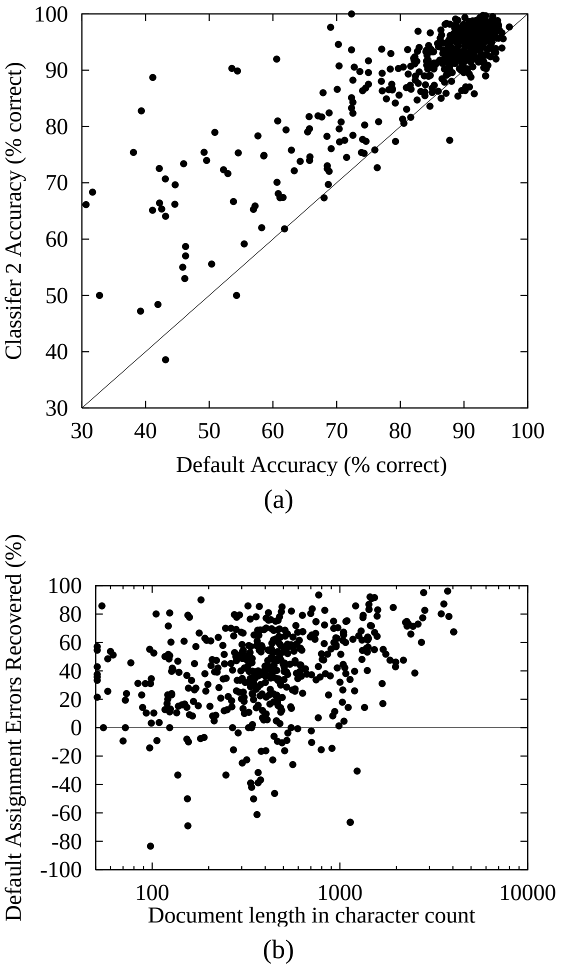

Figure 1. (a) Classifier 2 accuracy versus default assignment accuracy. (b) (%) Errors of default assignment of all symbols to LOWER that are recovered by Classifier 2.

of 97.0% over the test set. Interestingly, most errors are mistaking UPPERs for LOWERs, or PUNCTs for LOWERS, and the digits were decided very reliably (97.4% correct).

Notice that here we did not assume any shape clustering. When reliable shape clusters are available, these methods can be improved with voting within the clusters, or combined with Classifier 2 of the previous section.

5. Conclusions

We presented several methods to determine the type of a character from its context. Assuming reliable segmentation and shape clustering to determine the benefits of contextual information under ideal conditions, we found that a classifier trained on features of the n-gram statistics was useful and performed much better then the default assignment of all characters to lower case. Alternative assumptions of knowing approximate character heights, or neighboring types or identities also yielded satisfactory results.

From a pattern recognition point of view, we have the unusual situation of having discriminating features but immense variation in the frequencies of symbols and character types among the documents. It is possible that this situation calls for different types of classifiers than the ones we used.

These methods have good potential for improving recognition accuracy, especially for the cryto-analysis based methods, which may gain popularity with the advent of symbol-matching techniques for document compression. They can also be used with existing recognition engines to resolve conflicts and recover rejects.

References

[1] R. Casey, G. Nagy, Autonomous reading machine, *IEEE Trans. Comput.*, **C-17**, 5, May 1968, 492-503.

[2] G. Hart, To decode short cryptograms, *Commun. ACM*, **37**, 9, Sept 1994, 102-108.

[3] T.K. Ho, The Random Subspace Method for Constructing Decision Forests, *IEEE Trans. PAMI*, **20**, 8, Aug 1998, 832-844.

[4] T.K. Ho, Stop Word Location and Identification for Adaptive Text Recognition, *Int'l. J. of Document Analysis and Recognition*, **3**, 1, Aug 2000, 16-26.

[5] T.K. Ho, G. Nagy, OCR with No Shape Training, *Proc. of 15th ICPR*, Barcelona, Sept 3-8, 2000, 27-30.

[6] S. Khoubyari, J.J. Hull, Font and Function Word Identification in Document Recognition, *Computer Vision and Image Understanding*, **63**, 1, Jan 1996, 66-74.

[7] D.D. Lewis, Reuters-21578 text categorization test collection, Distribution 1.0, Sept 26, 1997.

[8] G. Nagy, S. Seth, K. Einspahr, Decoding Substitution Ciphers by Means of Word Matching, *IEEE Trans. PAMI*, **9**, 5, Sept 1987, 710-715,

[9] G. Nagy, N. Nagy, M. Soubourin, Signes diacritiques: perdus et retrouves, *Actes du 1er Colloque International Francophone sur l'Ecrit et le Document*, Quebec, 1998, pp. 404-411.

[10] A.L. Spitz, An OCR Based on Character Shape Codes and Lexical Information, *Proc. of 3rd ICDAR*, Montreal, August 14-18, 1995, 723-728.

[11] D. Yarowsky, A comparison of corpus-based techniques for restoring accents in Spanish and French text, *Proc. 2nd Workshop on Very Large Corpora*, Kyoto, 1994, 319-324.