A Document Recognition System for Early Modern Latin

Sravana Reddy
Dept. of Computer Science
The University of Chicago
sravana@cs.uchicago.edu

Gregory Crane
The Perseus Project
Tufts University
gregory.crane@tufts.edu

1 Introduction

Large-scale digitization of manuscripts is facilitated by high-accuracy optical character recognition (OCR) engines. The focus of our work is on using these tools to digitize Latin texts. Many of the texts in the language, especially the early modern, make heavy use of special characters like ligatures and accented abbreviations. Current OCRs are inadequate for our purpose: their built-in training sets do not include all these special characters, and further, post-processing of OCR output is based on data and methods specific to the domain language – most of the current systems do not implement error-correction tools for Latin.

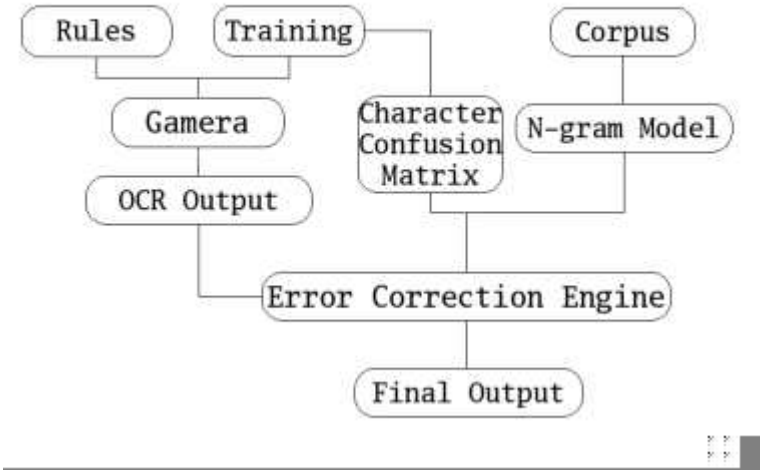
This abstract outlines the development of a document recognition system for medieval and early modern Latin texts. We first evaluate the performance of the open source OCR framework, Gamera, on these manuscripts. We then incorporate language modeling functions to sharpen the character recognition output.

2 Methodology

Gamera (http://gamera.sourceforge.net), developed by the Digital Knowledge Center at the Johns Hopkins University, is a framework for analysis and recognition of humanities documents. It performs the four basic steps of OCR: pre-processing of the image (to remove noise, deskew, and binarize), page segmentation (to separate the text from images), character segmentation and classification, and output page construction from the classified characters. [2].

We build a Latin-specific OCR engine from Gamera by training it on document images representative of the typesetting and characters unique to the domain. The output of the OCR is refined by passing it through a linguistic post-processing system. The system makes use of a corpus of ten texts from the Latin collection of the Perseus Digital Library[1] for data on character n-grams, and weighted character confusion probabilities gleaned from the OCR engine.

[1] http://alkmene.perseus.tufts.edu/hopper/collection.jsp?collection=Perseus%3Acorpus%3Aperseus%2CLatin+Texts



3 Evaluation of Gamera Output

Gamera segments page glyphs using connected component analysis, and builds glyph feature vectors. Classification from the training data is done using the k-NN (k-nearest neighbor) algorithm [1]. Rule-based classification can also be coded in to reduce the amount of computation.

We use the mathematical work of Jordanus Nemorarius, *De Ponderibus Jordani*, as the training set to classify pages from Sebastian Brant's *Stultifera Navis*. The first classification is done with a training file of the order of 1000 glyphs; the last is run with a file of around 20000 glyphs. Without any built-in rules, the performance peaks at the point of the 16000-glyph training file, identifying about 76% of the characters correctly. Adding rulesets for specific glyph-sizes or types of characters increases the performance of a 12000-glyph training set to 80%.

In comparison, the commercial OCR, PrimeRecognition©, recognizes about 70% of the glyphs on the same pages. The output of ABBYY FineReader©is marginally better than Gamera, with an accuracy of nearly 84%.[2] However, FineReader does not have the capability to recognize certain non-standard characters, which sets an upper limit to its performance.

4 Linguistic Post-processing

The most common error-correcting method built into OCR engines is a lexical check on a dictionary. However, this is not a foolproof post-processing strategy for an inflectional language. Rather, we check the output against a character-sequence trigram model, built from the Perseus Latin corpus.[3]

[2]The major drawback of Gamera that restricts its performance compared to commercial OCR engines seems to be the inability to define textual zones.

[3]We divide the corpus into prose and poetry, and use the appropriate genre depending on the text to be classified. However, since it's only at the character-level, the choice of corpora does not seem to make much difference.

This method of OCR correction has been explored by Kolak and Resnik in [3], where the entire model is a character-stream (word boundary probabilities being encoded by treating spaces as characters). It treats the OCR engine as a black box, with no information about its performance available to the post-processor. However, we have found that weighting the language model with the probabilities of characters being misclassified by the OCR largely improves our results.

During earlier Gamera evaluations and training, a weighted character confusion matrix is created. To check that a certain character in the OCR output is correct, we consider all those characters with non-zero weights in the corresponding row of the confusion matrix. The likelihood of an alternate character replacing the output one is defined as the sum of its **character confusion weight** and the **trigram probabilities** with the alternate character as part of the trigram. In order to minimize the distance between the original and the corrected output, we replace a character *only* if the probabilities of all three of the trigrams in which it occurs are higher than those of the original.

Many of the early modern documents use accents and special characters as abbreviations, where a mark over a 'base' character indicates the omission of one or more letters. The problem with resolving these is that they are ambiguous – the same mark can stand for different sequences depending on the context. Rydberg-Cox develops a method to resolve them in [4], by doing a search on the dictionary and morphological space around the base character.

We propose an alternate method that directly applies the n-gram character model to expanding abbreviations. When the post-processor encounters a special symbol, it searches the n-gram sequences for those sequences of characters that would likely precede and follow the base character. Provided that the training corpus is similar enough to the document image, the language model will contain a large number of these expansions as sequences, eliminating the need to hardcode a list of possible expansions. Since it is only an extension of the error-correcting algorithm, it can easily be implemented as part of it, performing as a good substitute for the Rydberg-Cox method when a morphological parser is not available.

5 Conclusion and Future Work

We have presented a pipeline system for Latin document recognition, by building a post-processor over an OCR framework. N-gram character models are the first step towards building a language verifier for Latin and other inflected languages. In ongoing work, we explore the possibility of using morphological stemming and analysis to build a richer language model. This will allow verification on the word *as well as the sentence level*, by checking the collocation of word forms. We are also working on using Gamera to choose its training data and generate rule-sets depending on the genre of the input image, and developing a closer interaction between the program and the post-processor which would allow on-the-fly customization of the training corpus.

References

[1] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[2] Michael Droettboom, Karl MacMillan, and Ichiro Fujinaga. The gamera framework for building custom recognition systems. In *Symposium on Document Image Understanding Technologies*, pages 275–286, 2003.

[3] Okan Kolak and Philip Resnik. Ocr post-processing for low density languages. In *HLT/EMNLP*, 2005.

[4] Jeffrey A. Rydberg-Cox. Automatic disambiguation of latin abbreviations in early modern texts for humanities digital libraries. In *JCDL*, pages 372–, 2003.