Transcribing Latin Manuscripts in Respect to Linguistics

Manqing Feng
Thayer Academy
Braintree, US
feng.mq12@gmail.com

Guoxin Huang
Department of Computer Science
University of Arizona
Tucson, US
gxhuang039@email.arizona.edu

Chuyuan Zhang Department
of Linguistics Univeristy of
California, Los Angeles
Los Angeles, US
jamesczhang@ucla.edu

Yang Liu
Computer Science &Artifical
Intelligence Lab
MIT
Cambridge, US
yangliu5@mit.edu

Yajun Fang
Computer Science &Artifical
Intelligence Lab
MIT
Cambridge, US
yjfang@mit.edu

Berthold K.P. Horn
Computer Science
&Artifical Intelligence Lab
MIT
Cambridge, US
bkph@csail.mit.edu

***Abstract*—Current text detection software, although can transcribe modern languages with high accuracy, has flaws detecting texts and transcribing original Latin manuscripts sufficiently. This paper proposes a general approach for transcribing Latin manuscripts in respect to linguistics and develops a system to transcribe Latin manuscripts containing intricate abbreviations, which combines basic object detection algorithms with linguistics. We used methods from image processing and made changes based on the characteristics of Latin.**

Keywords—transcription, text detection, linguistics, projection, sliding windows

I. Introduction

Latin manuscripts have always been favored by collectors and libraries. Famous manuscripts such as Virgil's *Aeneid* are the pride of celebrated libraries. In order to preserve the manuscripts in their best states, libraries would scan the manuscripts into digital version so that people could view the manuscripts online and prevent tactile damages. Latin manuscripts record poems and fold talks that allow modern scholars to comprehend Roman life. However, being able to appreciate Latin manuscripts has been a luxury for most Latin students because of the difficulties in recognizing handwritten Latin manuscripts. Furthermore, because the writing customs for Latin are significantly different from modern language, current text detection algorithms cannot efficiently transcribe Latin manuscript. For the purpose of experiment, we are using images from College of the Holy Cross's database. [1]

II. State of the Art

The current text detection technologies can effectively transcribe modern languages with an accuracy of 98 percent. Furthermore, software such as Adobe Acrobat claims to be able to transcribe Latin. However, because of the specialness of Latin manuscripts, these algorithms still fail to transcribe the aboriginal text. There are three main problems faced by current text detection technologies: the presence of images in manuscripts, authors' personal preference for abbreviations, and the combinations and variations of ancient hand-written letter. Here's a review of the problems faced by algorithms:[2]

A. Failing to Detect Text With the Presence of Images

Most current text detection technologies require the inputs to be manually managed beforehand: with images and most noises cropped and fixed, only text can go in as inputs. Otherwise, the software would mistake drawing strokes as terrible hand writings. Fig. 1 is an example of the inputs that most text detection programs are not able to extract texts accurately.

Fig. 1. An example of a recto page of a Latin manuscript with the presence of images. This would be an example that current text detection algorithms have trouble with.

B. *Inability of Recognizing Abbreviations*

Different authors have different customs for hand writings. Although the authors of the Latin manuscripts were all educated citizens and had similar hand writings, they had their personal systems of utilizing abbreviations, as in Fig. 2. These personal systems require the software to learn the systems every time it tries to transcribe a piece of text from the manuscripts. In other words, the laborious process of learning has to be repeated by the algorithms each time it was dealing with the inputs from a different author.

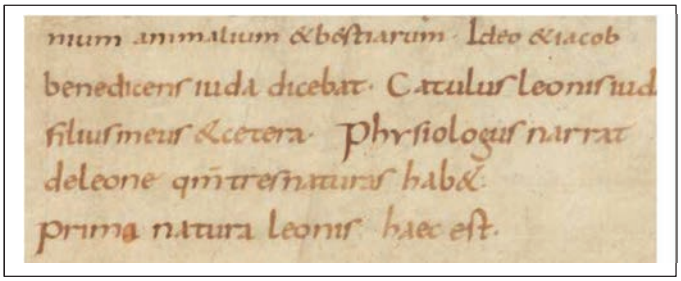

Fig. 2. In the second word of the 4th line of this example, qm (with a long mark) is an abbreviation for "quoniam" in Latin meaning "since."

C. *Mistaking letters and the combinations of letters*

Fig. 3 is a short list of common mistaken letters and mistaken combinations of letters.

"i n"	for	"i u"
" e "	for	" c "
" b "	for	" h "
" o "	for	" u "
" li "	for	" u "
" l "	for	" t "
" t "	for	" f "
" . "	for	" , "
" n "	for	" u "
(dashes)		
(non-letter)		
Etc.		

Fig. 3. A short list of common mistaken letters and combinations

III. The Proposed Solution

These are not only the obstacles faced by Latin transcription but also other ancient language recognitions. Based on linguistics, a word is consisted of one root for its meaning and multiple affixes indicating its usage, as in (1). The dictionaries most text recognition algorithms used cannot check for affixes and roots separately. If an algorithm aims for accuracy it would lose much speed checking each output and vice versa. Furthermore, because of the peculiar rules for transcribing Latin manuscripts that all abbreviation should be expanded, a special dictionary has to be imported [3]. As a result, linking linguistics with text detection would be an indispensable step to improve contemporary algorithms.

$$Word = (affix)\text{^}x(root)(affix)\text{^}x \quad (1)$$

IV. Medthods

Because the methods used for image segmentation only operate on a 2-dimensional map, all inputs were first converted from RGB to gray scale. Then for the purpose of projection, a threshold was set to help further distinguish the blank spaces and the letter strokes.

```
_ ,img = cv2.threshold(img,175,255,cv2.THRESH_BINARY_INV)
```

Fig. 4. A thresholding method from CV2 was used for this purpose.

A. *Projection and Image Segmentation*

A horizontal projection, as in Fig. 5, was done to the monotonous input to segment the text image into lines.

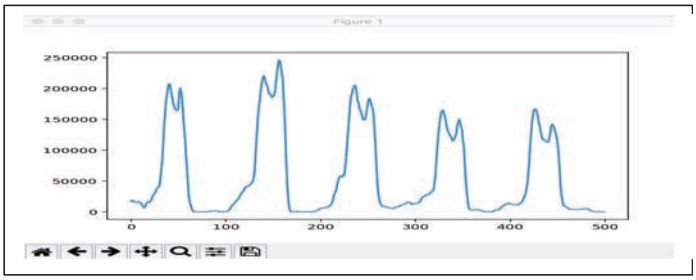

Fig. 5. A horizonal projection of one of the inputs.

Then, a vertical projection, as in Fig. 6, was used to find the possible cutting points that would cut the lines into words.

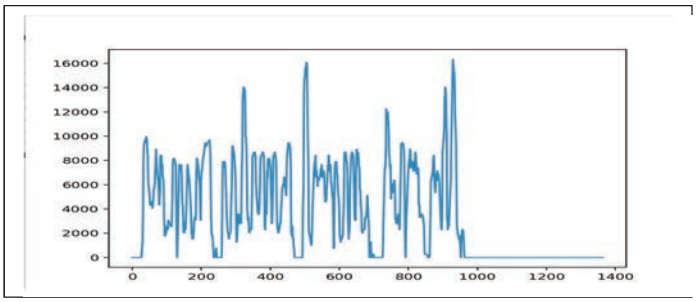

Fig. 6. A vertical projection of the segmented lines. The valleys in this plot represent the possible cutting points.

One of the main problems for image segmentation is that the heads and tails of a letter would cover the space between words. In other words, if we just simply use projection once for segmentation, the result would contain unseparated words like Fig. 7.

Fig. 7. Segmented words based on vertical projection. This output contains three unsegmented words "catulus," "leonis," and "iuda." Because the letter "s" in Latin has its head extended into the next character, the vertical projection fails to cut the lines into words accurately.

However, by further cutting the words based on its two peaks in the projection as in Fig. 8, the problem of the heads and the tails of the words covering blank spaces between words could be solved by separating the image into two parts based on the peaks.

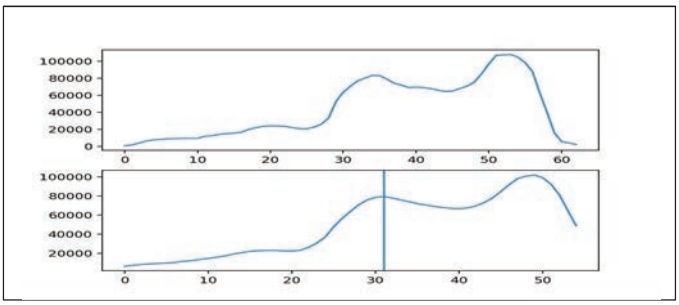

Fig. 8. A horizontal projection of Fig. 7. The two peaks are obvious.

The cutting line is represented by the red line shown in Fig. 9. The vertical projections are used to separate the words from each other.

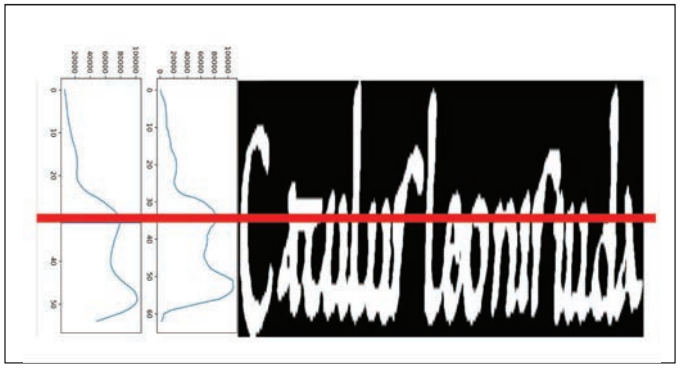

Fig. 9. The red line indicates the second cutting line that help segment words from the line input.

The cutting points are determined by the starting positions and the ending positions of valleys based on the vertical projection as in Fig. 10. The cutting points are used for segmenting both the peaks and the valleys so that complete words would be produced as the results. The segments from corresponding peaks and valleys are stacked back together in the end.[4]

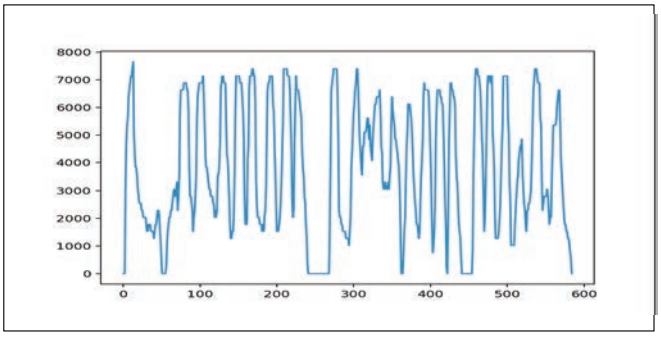

Fig. 10. The vertical projection of Fig. 7 after second cutting.

B. *Sliding window object detection*

With all the words saved in one folder, we can use 2-dimensional sliding window searching by first building an image pyramid. Then a searching window loops through each layer of the pyramid [5]. The window size is defined by sample letters from a folder. An example is shown in Fig. 11. With the contents in the searching windows and the sample letter having been compared by using sum of squared difference (SSD), a histogram is produced to estimate the possibility of the letter as in Fig. 12. In this example, "n" template is used for finding the existence of "n." The result is accurate because based on the histogram, "n" has the lowest SSD. This method deals with the inability of identifying abbreviations and ambiguous combinations of letters effectively.

Fig. 11. A sliding window is running through the word "nium."

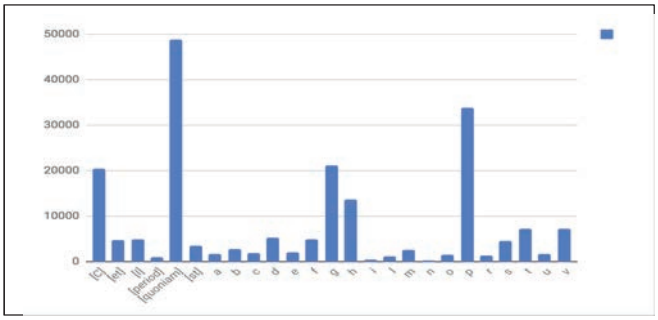

Fig. 11. The evaluation histogram of Fig. 11. According to the histogram, the most possible first letters for Figure 10 are "n," "i," and "[period]."

C. *Evaluating and building the system*

There are four main factors that would determine the determination of a letter: the shape of the letter, the position of the letter, the possible word containing the letter, noises around the letter. Each of the four factors would contribute to the output by different degrees. The result bid fair to be displayed as a possibility. One input could have multiple possible output. The one with the highest possibility would not be necessarily correct. Therefore, all possible output should be displayed to the user for investigation.

V. Morphosyntactic Analysis of Possible Interpretations

The previous sections illustrate the approaches of identifying isolated letters, wherein the ligatures and alternate forms existing in the manuscript could prove to be great obstacles in correctly interpreting the passage. Further analysis needs to be implemented to lower the number of possible interpretations.

The Latin language has not seen much change since the last works in Renaissance Latin. In our situation with the manuscripts, the Latin language used is even more dated. It is technically possible to store every single word in a language that has not seen much change in the last few centuries.

However, as a fusional language, Latin utilizes inflection heavily. Whereas in English relationships between words are described extensively with helper words such as prepositions, Latin relies on those words much less than English, using inflected endings to convey the relationship.

Depending on the (syntactic) role of each word in the sentence, inflected words can take on anywhere from 12 forms (nouns without attested locative case) to around 100 forms (non-deponent, non-defective irregular verbs like "video, videre, visi, visum". Storing every form of every lemma in Latin for later interpretations to be checked against is both space consuming and time consuming, from both labor and complexity perspectives.

However, as demonstrated in the gloss above, it is possible to segment Latin words into one or more morphemes, each containing one or more meanings. A list of all possible morphemes can be stored as a triplet (t, c, p) where t is the text, c is the category (lexical class) and p is the position (stem or affix; if affix, prefix or suffix). By adding another slot r specifying the morphological role of the morpheme, we can further restrict the combination of morphemes, e.g. if we have the following morphemes:

- (*vide*, verb, stem, (+IND, +PRS))
- (*visi*, verb, stem, (+IND, PRF))
- (t, verb, suffix, (3SG))
- (∅, verb, suffix, (1SG, +PRF))
- (o, verb, suffix, (1SG, +IND, +PRS))

we can disqualify words such as **vide*, **visio*, **ovide*, **tvisi*, and **visiot*.

By analyzing the preliminary results obtained from image processing, it is possible to reduce the number of potential valid interpretations.

VI. Conclusion & Future Work

With the help of the projection, 96.667% of the words were able to be segmented out accurately as independent words. If common abbreviations were treated as one letter in the process of evaluating each letter, the segmentation success rate would have been improved significantly. Using image pyramid and sliding windows has been successful in finding letters so far as well. However, the process of creating a template folder takes lots of human efforts: it requires specialists to select letters as templates by hand. Fortunately, because most Latin transcriptions have clear and similar handwriting, we can use existing template folders that are similar to the calligraphy of the input.

In order to get more accurate results for the evaluated letters, we plan to utilize Convoluted Neural Network (CNN) with manual selected data.

Future work in this area includes automatically separating texts and images from an input through identification of the shapes in an input. In other words, because texts in Latin manuscripts often come in blocks, or rectangular shape, by segmenting the input based on shapes we can extract most of the texts. Regard for modern technologies' inability of recognizing abbreviations and confusing combinations of letters, sliding windows combining with CNN would solve the problem.

Acknowledgment

Feng Manqing thanks Ms. Callie Schneider and College of the Holy Cross.

Reference

[1] Holy cross manuscript hackathon. (2018, February 24). Retrieved November 22, 2018, from https://hcmid.github.io/ms-hackathon-2018/thumbs/

[2] Hawk, B. W. (2015, April). *OCR and medieval manuscripts: Establishing a baseline*.

[3] Treharne, E. (n.d.). Making a transcript of a manuscript text. *A.R.Rumble's Teaching Materials*, *I*(IV). Retrieved from https://lagunita.stanford.edu/assets/courseware/v1/7865cb0e3bed0ca7c86eada70c395ddb/c4x/English/DiggingDeeper1/asset/PP_Transcribing_a_Manuscript.pdf

[4] Ghaleb, H., Nagabhushan, P., & Pal, U. (2017). Segmentation of offline handwritten Arabic text. *2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*.

[5] Rosebrock, A. (2015, March 23). Sliding windows for object detection with python and opencv. Retrieved November 22, 2018, from pyimagesearch website: https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/