Michael Zhong

mqz2001

Final Project: Typography Classifier

I. Problem Description

I sought out to build a typography classification system, that given a series of text produced in one of seven typographies I selected (Arial, Baskerville, Georgia, Ideal Sans Book, Minion Pro, Times New Roman, Verdana), could detect which typography was used in the text. Typography classification systems already exist both commercially and for free, the most visited free online typography classifier by far according to Alexa rankings www.myfonts.com/WhatTheFont. However, my personal experiences from using these typography classifiers have been less than stellar.

Typography classification is far from a solved problem. The best published research results can achieve roughly 96 percent accuracy, but are based on a limited number of typography selections and almost all research conducted on typography classification have been performed in idealized environments (perfect textual orientation on a white sheet of paper).

Typography classification research has been used to further extend research in many fields. In addition to the field of improved typography recognition, typography classification has also been used in research to improve optical character recognition systems.

II. Dataset

Data for the typography classification system was hard to get by. For the training data, I used high-resolution fonts (typically 250 pixel sized letters for a given typography). I saved images of every character of size 250 pixels for lowercased letters, uppercase letters, and numerical characters. For seven fonts, this exercise was tedious but manageable. However, there is an inherent scalability problem for using the dataset in this manner. Over 100,000 different typographies exist, and for a far more comprehensive typography classifier than the seven-type classifier I implemented for this project, it would be unrealistic to manually collect the dataset in this manner.

Another attempt I made at collecting the dataset was to outlay all the different characters in one image file, than using an optical character recognition system, extrapolate the character and typography features there. However, that was problematic because the optical character recognition system classified on average just 92 percent of the characters correctly. So individual image files for each character of each typography were used as the main data source instead. Due to the large size of each character, jagged pixel resolutions are not a problem for feature extraction, though gathering the training data is tremendously more tedious.

For test data, most research test data used perfectly horizontal black text interpolated with a white background. I did the same. I had a rather limited 15 samples with two samples of each typography (except for Times New Roman, which had three samples), with each typography having different sizes.

III. Approach

3.1 Deciding on a Methodology

Overwhelmingly in research, feature extraction methods have involved some form of summing up the number of pixels in a given region. The basic process of pixel summation is shown in Figure 1. In Figure 1, all the black pixels for each vertical position are summed up and then plotted on a histogram with respect to the given vertical position. Then, typically by comparing the slopes of the pixel summation from one pixel summation set to another, one can then compare the feature set of the slopes between the training data and the test data.

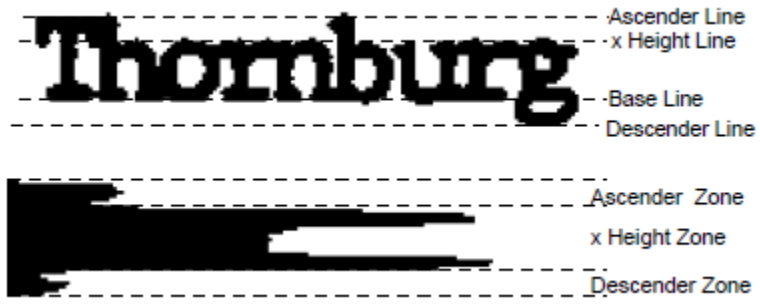
Figure 1: Vertical pixel summation

The pixel summation method is by far the most prevalent in research. However, there are many variations of this approach which are applied. The simplest, and one of the most effective approaches, is as demonstrated in Figure 1—sum up all the pixels vertically and/or horizontally.

A closely related idea—and the one I implemented—is to segment each different character, then create feature vectors for each character. Each character is then classified, with the typography achieving the most classifications being the result. Segmentation by each character has its benefits—since many fonts tend to have remarkably similar or at times indistinguishable features for some characters, but different features for certain letters, the letters with the different features should have more importance. Segmenting by character would give each individual character a weight. However, the drawback to this approach, and the reason why segmenting over a line of text is another common approach, is that characters with little distinguishable difference in

features (thus making a classification error more likely), have equal weight as characters with large distinguishable differences.

Figure 2: Feature Segmentation



Attempts have been made to further segment the features. Particularly between serif and sans-serif fonts (a serif is a small semi-structural detail at the end of the stroke of a character), attempts have been made to segment the serifs. Segmentation by zones where typically a text is segmented by upper zone, middle zone, and lower zone (as seen in Figure 2), is also a popular approach. These approaches are nice in theory, but in practice have been difficult to implement and these classifiers typically score no better than the simpler horizontal or vertical pixel summation methods.

The approach I've used is thus as follows:

1) Segment each individual character in a sample of text.

2) Apply horizontal pixel summation for each individual character and using the k-nearest neighbor algorithm, find the best type for that particular character.

3) Repeat for all the characters in the text, then make a decision on the typography that was used.

3.2 Character Segmentation

Optical character recognition is a difficult and far from solved problem—it is difficult to even somewhat accurately distinguish characters given an image. Implementing an optical character recognition program was beyond the scope of this project; therefore I used Tesseract, a free open-source optical character recognition library package. It is a respected open-source solution, and is currently being maintained by researchers at Google.

Tesseract itself was not a perfect classifier. For the test data, 92 percent of characters were correctly identified. However, other open-source options fared far worse. Tesseract also has an API function that can draw a bounding box for each individual character, thus making isolating the character for horizontal pixel summation easy.

3.3 The Slope Feature Set

Give the pixel distributions of both the training set character and the test set character, the slope feature set can be calculated. Since the training set character was set in size 250 font, typically the training set character would have more pixels. To account for this, I took the ratio between the test set character width and the training set character width, then for the slope between every two pixels in the test set, I corresponded that slope with the slope between every calculated n pixels in the training set. In theory, the slopes should be similar. Then, using the k-nearest neighbor algorithm, I found the most similar typography for each individual character.

3.4 Results

On a sample of 15 test data samples, the classifier scored 80 percent, getting 12 out of 15 of the samples correct. Both Ideal Sans typography samples were misclassified (one time as Verdana, the other time as Arial), and a Baskerville sample was misclassified as Times New Roman. The results are well-below leading typography classification research (the best typically score at

roughly 95 percent accuracy). However, even among the misclassified samples, the results to an extent are promising. Ideal Sans, a sans-serif font, is similar in style to Verdana and Arial, both also sans-serif fonts. In addition, Times New Roman as a serif font was inspired in part by Baskerville and thus have many similar traits. Thus, the misclassifications are not egregious—meaning sans-serifs are being mixed with similar sans-serifs. The misclassifications in general also tended to be predictions with little confidence (for the Times New Roman classification, Times won out by a single character over Baskerville), while the correct classifications tended to be resounding predictions of confidence.

I also experimented with a classifier that used vertical pixel summation rather than horizontal pixel summation, but results were not as good, achieving only 9 out of 15 correct classifications. Certainly though this is too small of a test data sample to conclusively determine horizontal pixel summation created a better feature set than vertical pixel summation.

IV. Conclusion

For this project, I implemented a typography classifier that can classify between seven of the most popular typographies today. I used horizontal pixel summation for each individual character, then extracted the slopes from these horizontal pixel distributions to form the feature set between the training data and the test data. A k-nearest neighbor classifier was used to then predict the closest matching typography. The typography classifier achieved 80 percent accuracy, but even on the miscalculations, tended to predict similar typefaces.

There is ample opportunity for future work in this project. For one, the manual insertion of both training and test data for typography is not realistic for a large-scale project or for classifiers with many possible typefaces. Better data collecting methods could be explored. The project could

expand to more typefaces. In addition, more combinations of different feature extraction methods (for example, vertical pixel summation with baseline analysis or vertical and horizontal pixel summation) could be applied for experimentation.

V. References

[1] *Multifont Classification using Typographical Attributes*, Jung, Shin and Srihari, 1999, IEEE

[2] "Alexa - The Web Information Company."*Alexa - The Web Information Company*. N.p., n.d. Web. 13 Dec. 2012. <http://www.alexa.com/>.

[3] *Optical font recognition using typographical features*, Zramdini, Ingold, 1998, IEEE

[4] Linux.com :: Google's Tesseract OCR engine is a quantum leap forward."*Linux.com :: Forum Index*. N.p., n.d. Web. 13 Dec. 2012.

[5] *Font recognition based on global texture analysis*, Zhu, Tan, Wang, 2001, IEEE