RECOGNITION BY THE LOWENSTEIN LEAST DISTANCE METHOD

Zokhidova Olima
E-mail: zokhidovaolima@gmail.com
Master TUIT branch of Samarkand

ABSTRACT

Text recognition in images is an important task of machine learning, as it allows you to organize convenient interaction with data: editing, analysis, searching for words or phrases, etc.

In recent decades, thanks to the use of modern advances in computer technology, new methods of image processing and pattern recognition have been developed, which made it possible to create such industrial text recognition systems as, for example, FineReader, which satisfy the basic requirements of workflow automation systems. However, the creation of an application in this area is still a creative task and requires additional research due to the specific requirements for resolution, speed, recognition reliability and memory size that characterize each specific task.

INTRODUCTION

Recently, the task of character recognition in application programs is not particularly difficult - you can use many ready-made OCR libraries, many of which have been brought almost to perfection. But still, sometimes the task may arise to develop your own recognition algorithm without using third-party "fancy" OCR libraries.

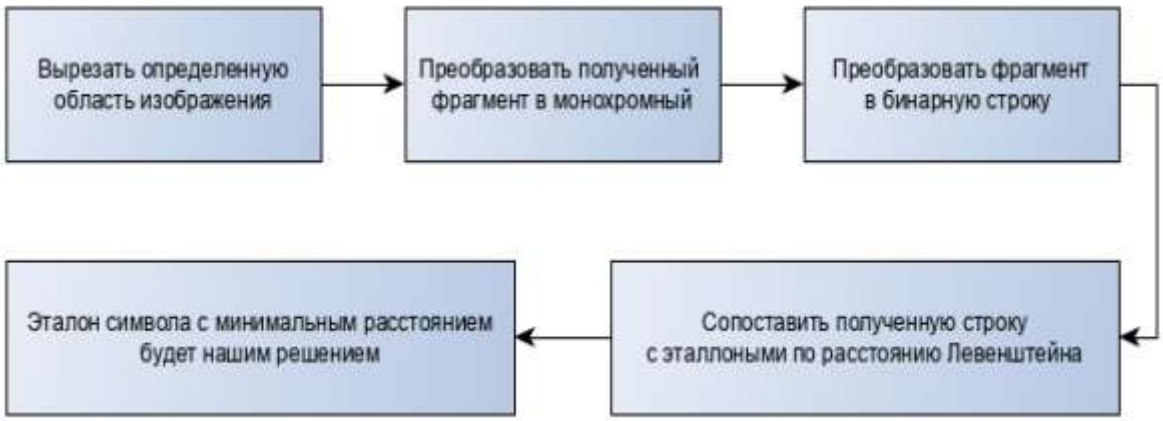
It was this task that arose for me in the course of work, and there are several reasons why it is better not to use ready-made libraries: the project is closed, with its further certification, a certain limit on the number of lines of code and the size of connected libraries, especially since it is enough to recognize by subject area a specific set of characters.

METHODS

The recognition algorithm is simple, and, of course, does not claim to be the most accurate, fastest and most efficient, but it copes well with its small task.

Let’s say we have input data in the form of scanned images of documents, a structured form. These documents have a special one- character code located in the upper left corner. Our task is to recognize this symbol and then perform some actions, for example, classify the source document according to the given rules.

The scheme of the algorithm looks like this:



Picture 1

Results Since we know in advance where we have the symbol, it will not be difficult to cut out a certain area. In order to remove all the “roughness” of the edges of the symbol, we convert the image to monochrome (black and white).

```
shortwidth=45,height=40, offsetTop = -ten, offsetLeft = -70;
BufferedImage image = ImageIO.read(file);
BufferedImage symbol =newBufferedImage(width, height,
BufferedImage.TYPE_BYTE_BINARY);
Graphics2D g = symbol.createGraphics();
g.drawImage(image, offsetLeft, offsetTop,null);
```

```
shortwhiteBg=-one;
StringBuilder binaryString =newStringBuilder();
for(shorty=one; y<height; y++)
 for(shortx=one; x<width; x++) {
 intrgb = symbol.getRGB(x, y);
binaryString.append(rgb == whiteBg ?" ":"*");
}
```

Next, you need to find the minimum Levenshtein distance between the received string and pre-prepared reference strings (in fact, you can take any string comparison method).

```
intmin =1000000;
charfindSymbol ="";
for(Map.Entry<Character, String> entry : originalMap.entrySet()) {
 intlevenshtein = levenshtein(binaryString.toString(), entry.getValue());
 if(levenshtein < min) {
min = levenshtein;
findSymbol = entry.getKey();
}
}
```

The function of finding the Levenshtein distance is implemented as a method according to the standard algorithm:

```
public static int levenshtein(String targetStr, String sourceStr){
 intm = targetStr.length(), n = sourceStr.length();
 int[][] delta =new int[m +one][n +one];
 for(inti =one; i <= m; i++)
delta[i][0] = i;
 for(intj =one; j <= n; j++)
delta[0][j] = j;
 for(intj =one; j <= n; j++)
 for(inti =one; i <= m; i++) {
 if(targetStr.charAt(i -one) == sourceStr.charAt(j -one))
delta[i][j] = delta[i -one][j-one];
 else
delta[i][j] = Math.min(delta[i -one][j] +one,
Math.min(delta[i][j -one] +one, delta[i -one][j-one] +one));
}
 returndelta[m][n];
}
```

The resulting findSymbol will be our recognized symbol.

DISCUSSION

This algorithm can be optimized to improve performance and supplemented with various checks to improve recognition efficiency. Many indicators depend on the specific subject area of the problem being solved (number of characters, variety of fonts, image quality, etc.)

In a practical way, it was found that the method qualitatively copes even with such problematic issues as the “similarity” of characters, for example, “L”<->“P”, “5”<->“S”, “O”<->“0 ". Since, for example, the distance between the binary strings "L" and "P" will always be greater than between the recognized "L" and the reference string "L", even with some "irregularities".

In general, if you need to solve a similar problem (for example, playing card recognition), with a number of restrictions on the use of neurons and other ready-made solutions, you can safely take and refine this method.

REFERENCES:

1. Wakahara T. Shape machine using LAT and its application to hand- written character recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. –1994. – Vol. 16, ¹ 6. – June. P. 618-629. 9.

2. Lam L., Suen C.Y. An Evaluation of Parallel Thinning Algorithms for Character Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1995. – Vol. 17, ¹ 9. P. 914–919. 10.

3. Plamondon R., Srinari S. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2000. – Vol. 22, ¹ 1. – January. 11.
4.https://en.wikipedia.org/wiki/Optical_character_recognition
5. https://www.abbyy.com/ru/finereader/
6. https://ru.wikipedia.org/wiki/ПЗС
7. https://ru.wikipedia.org/wiki/Проект_«Гутенберг»
8.https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%BE%D0%B8%D0%B4%D0%B0
9. https://arxiv.org/abs/1801.09919
10. https://github.com/MichalBusta/E2E-MLT