Fontys Hogeschool Techniek en Logistiek

Software Engineering

DAMI

Research Report

Domenik Irrgang (2391295)
December 15, 2017

Summary

This document is written for the subject DAMI during the computer science study course in semester 7 at Fontys Venlo. It is supposed to describe the process of a research regarding data analysis and machine learning. It should describe the problem, a research question that is derived from that problem and results made during said research. The research was conducted during a 4 week period at the end of the semester. The research is going to be about the possibility of transforming hand written notes into ASCII characters.

Contents

List of Figures

1 Introduction

Everybody has had the problem before, it is not possible to read someones hand writing or even your own. This kind of situation either happens when you need to take notes very rapidly like in a lesson or an important meeting. The hand writing gets a lot worse because people in meetings and lessons tend to talk very fast or have rapid discussions between each other, which makes it difficult to maintain a good handwriting while keeping up with the speed of communication. After such a situation it often is required to digitize the hand written notes into "computer letters" to send them to the participants via mail. This is a tedious process, because 1) the handwriting is often bad and 2) you need to write down something you already wrote down. Sending a copy of the note is only going to solve the second problem, because the hand writing might still be unreadable by the recipient.

2 Research Question

This research is going to look into the possibilities of solving the problem described in the last chapter. The resulting research questions therefore is: How can hand written notes be transformed into ASCII characters?. The answer to this question will solve the problem of not wanting to write down the notes twice and also will make it readable for everyone wanting to read it.

3 Analysis

This chapter is going to describe how the research question has been tried to answer.

3.1 Model

As the problem at hand requires to distinguish between different letters or words this is a classification problem. Classification problems can be solved by neural networks. Now it needs to be decided which model to use for training. In this case a softmax regression model has been chosen, because it is a model fitting the classification problem, that is easy to use (project is time constrained), but will still provide a solution to the problem, even if a higher accuracy can probably be reached with a different model. The library used in this research for doing this is called "Tensorflow".

3.2 Dataset

The problem at hand shall be solved using a dataset that is tailored for the persons handwriting that it is trying to read. Therefor for every person that wants to use this a new dataset needs to be created. Because of this the model should work without much training data. If the model would be used to classify complete words or sentences this would quickly get out of hand. This suggests that the model should be reading individual letters instead of a composed image of letters. That way the user only needs to provide some images of his written individual letters. The model now should need a lot less images compared to the approach with words and sentences. Also two different datasets are needed during this research. One is used to train the model and another one is for testing or verifying the functionality of the trained model. Both of these datasets need to be different from each other to actually verify the result.

3.3 Separation

As the model is only going to be able to classify characters and not whole words or sentences the note needs to be split up into different parts.

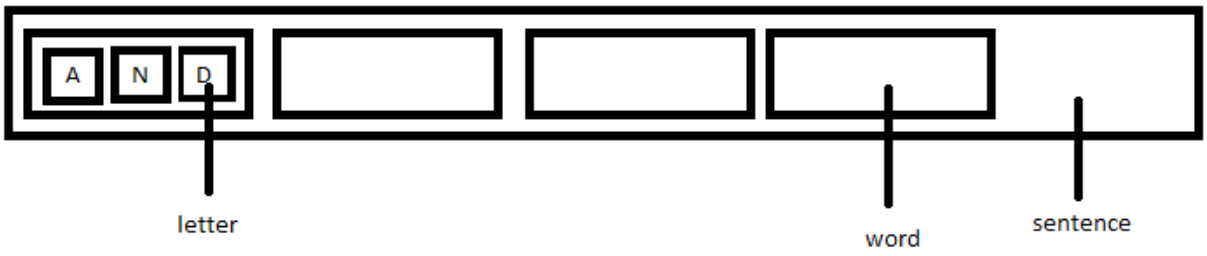

Figure 1: Structure of a note

As shown in the figure above, a note is composed of three different types of elements: sentences, words and letters. For this model to work we need some other application doing the work of separating the sentences from each other, one for separating the words in a sentence from each other and another one for separating letters of a words of each other. This is not going to be part of this research. It will focus on classifying the letters.

4 Implementation

This chapter is going to describe how the model has been implemented.

4.1 Loading the dataset

Before any training or data analysis can be done the dataset first needs to be loaded. The datasets reside in the projects root folder in the Letter folder. There is one folder called "Training" which contains the dataset for training and one folder called "Testing" which contains the dataset for verifying the results. Inside the dataset every letter has its own folder for itself. These folders contain only images of that specific letter. The name of the folder is going to get used as the label for the letter during training. To be able to assign the labels to ASCII characters the name of the folder needs to be the same as the corresponding ASCII code -65, because thats where the letters in the ASCII table start.

```
def loadDataSet(directory):
    directories = [d for d in os.listdir(directory)
                   if os.path.isdir(os.path.join(directory, d))]
    letterLabels = []
    letterImages = []
    for d in directories:
        labelDirectory = os.path.join(directory, d)
        file_names = [os.path.join(labelDirectory, f)
                      for f in os.listdir(labelDirectory)]
        for f in file_names:
            letterImages.append(data.imread(f))
            letterLabels.append(int(d))
    return letterImages, letterLabels
```

Figure 2: Function to load dataset

This function is loading a dataset in the format that got described above from a given directory. This way the training and testing dataset can be loaded using the same function. It splits up the name of the labels and the actual image data of all images and folders it finds in the dataset into two different lists and returns them. Now the function is used to load the training data.

```
ROOT_PATH = "/home/domenik/Documents/DAMI"
train_data_directory = os.path.join(ROOT_PATH, "Letters/Training")
test_data_directory = os.path.join(ROOT_PATH, "Letters/Testing")
images, labels = loadDataSet(train_data_directory)
testImages, testLabels = loadDataSet(test_data_directory)
```

Figure 3: Loading the data

First the directory of the datasets need to be defined which is "¡Project-Root¿/Letters/¡DataSetname¿". Both the training and testing dataset gets loaded already.

4.2 Transform Data

The pictures and labels now need to be transformed into the correct format for using the model. But first some stats of the data needs to be calculated or defined.

```
images = np.array(images)
labels = np.array(labels)
unique_labels = set(labels)
imageHeight = 56
imageWidth = 56
imagePixelCount = imageHeight * imageWidth * 4
outputCount = len(unique_labels)
imageCount = int(images.size / imagePixelCount)
```

Figure 4: Calculate/Define stats of input data

The images and labels are now stored in a numpy array. The width and height for all images is the same namely 56x56 pixels. Because the model needs to know how many input values it has we need to calculate the amount of pixel which is just the height times the width of the image. Because one pixel is composed of four values (Red-Green-Blue-Alpha) this number needs to be multiplied by four. The size of the output of the model is dependent on how many different unique letters it will get trained for. The number of images that are in the set can now be calculated by dividing the arrays size (size in bytes) by imagePixelCount. That works because one color value is between 0-255 which is one byte. After the stats of the training set have been calculated the data needs to be prepared into the correct format.

```
convertedImageList = [images[x].ravel() for x in range(0, imageCount)]
imageArray = np.array(convertedImageList)
convertedLabelList = [[ 0 for x in range(0, outputCount)]
        for y in range(0, labels.size)]
labelArray = np.array(convertedLabelList)
for x in range(0, labels.size):
    labelArray[x][labels[x]] = 1
```

Figure 5: Converting dataset into format to feet to neural network

First the images are converted to an numpy array in which the format of the data looks like this: [[color value, color value, color value, color value] [next picture] ...]. The labels are converted to match the output size of the neural network into the format of: [[0, 0, 1], [1, 0, 0]]. In this case the outputCount would have been three.

4.3 Setting up neural network

Now that the data is prepared the only thing left to do is to setup the neural network for training.

```
x = tf.placeholder(tf.float32, [None, imagePixelCount])
W = tf.Variable(tf.zeros([imagePixelCount, outputCount]))
b = tf.Variable(tf.zeros([outputCount]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
y_ = tf.placeholder(tf.float32, [None, outputCount])
```

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indic
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
```

Figure 6: Setting up neural network

The input and output layer get the proper sizes depending on the training set used, which the values that got calculate earlier.

4.4 Training the model

The model is now trained using the loaded training data. "epochs" being the number of times the training should be repeated.

```
for _ in range(epochs):
  sess.run(train_step, feed_dict={x: imageArray, y_: labelArray})
```

5 Conclusion

Now that the model is able to read individual letters, an application that can cut a word into its individual pieces needs to be done. Those individual letters are then fed to the model. Once that is done an application splitting a note into its individual words needs to be implemented. After that a note can be converted into its corresponding ASCII characters. Using a neural network with Tensorflow was a good choice for solving this problem as it was pretty easy to use and had a good documentation online and achieved the goal of this project.