DATA STANDARDIZATION USING DEEP LEARNING FOR HEALTHCARE INSURANCE CLAIMS

Kaelan Renault and Amr R. Abdel-Dayem

Bharti School of Engineering and Computer Science,
Laurentian University, Sudbury, Ontario, Canada

ABSTRACT

This paper presents a deep learning model that can be used for data standardization tasks. With applications such as insurance processing, accounting, and government forms processing – being able to standardise data that is presented in a nonstandard format would be impactful across many industries. Restaurant receipt images from CORD dataset were used to build the model. These images were previously processed with OCR then pre-processed to create JSON files that contain the OCR'ed data and metadata of the information on those images. The main challenge in building the model is that the size of the data set is very small (1,000 images). In order to overcome this challenge, an augmentation stage was employed to generate more training samples out of the existing ones. While this standardization problem can be modelled as a classification task, it has been decided to attempt using a regression model that can predict the total on a receipt.

KEYWORDS

Deep Learning, Machine Learning, Data Standardisation, Regression Model, Data Augmentation, Healthcare Insurance.

1. INTRODUCTION AND PROBLEM DEFINITION

This article will focus on applying deep learning techniques in a new application and show that the task of standardizing common data currently presented in nonstandard formats can be performed by a computer system rather than done manually. An example of this type of problem that seems like one of the easiest to convey the potential impact of this research is the process of adjudicating a health insurance claim. When a health insurance claim is submitted to a health insurance provider there are several methods for submitting the claim. The most popular claim submission type for drug claims is the pharmacy submitting the claim on the patient's behalf and immediately knowing how much of the claim will be paid by the health insurance provider. If the patient elects to submit the claim themselves they can fill out forms online, they can mail the provider paper claims, they can fax claims to the provider, or they can submit a "photo claim". Photo claims allow the patient to take a picture of their receipt and upload it directly to the healthcare insurance provider [1]. These types of claims are becoming more popular with the increase in mobile devices and easy access to the internet; they are especially gaining traction for health insurance claims that cannot be submitted by the practitioner as easily, such as with extended health care (EHC) treatments [2]. When a health insurance provider receives a photo claim the person who adjudicates the claim must process the claim by examining the image and filling in the information to the in-house claims adjudication software. The adjudicator uses the software and their own knowledge of how the patient's health insurance plan is calibrated to determine how much, if any, of the claim will be paid by the health insurance provider. The reason that an adjudicator needs to manually examine and enter the data from the receipt is that

the receipts of all healthcare practitioners across Canada are not standardised. They all have the same basic information needed to adjudicate the claim, but the information is organised differently across the receipt; worded differently for values that have the same meaning (Final Total, Grand Total, Total, etc.); values are differently associated to their labels, some might be directly adjacent to the label, some directly below, some across the page but organised in a table, some across the page in a list. Due to these factors, it is not trivial to automatically adjudicate these types of claims so companies must pay hundreds of individuals salaries to adjudicate claims which they can only do as quickly and as accurately as their human capacities allow. As a result, health insurance companies do not guarantee 100% accuracy and tend to offer repayment in terms of multiple days or even weeks [3][4][5]. Replacing these adjudicators with automatic software driven adjudication will allow these companies to adjudicate claims in real time and with close to 100% accuracy. The only difference between pharmacies submitting a claim on the patient's behalf and the patient submitting the claim themselves via photo claim is data standardisation. Pharmacies use software like Kroll [6] or Propel Rx [7] to send the data in a standard format to any health insurance provider. If software can read the same information that is already present on the photo claims that patients submit, the data can be processed by the same automatic adjudication engine that instantly processes claims submitted by pharmacies.

2. BACKGROUND

The problem at hand is data standardisation. The practice of converting images of text into data representations of text has been solved in one way or another since 1914 when Emanuel Goldberg developed a machine that converted printed characters into telegraph code [8]. This practice, known as Optical Character Recognition (OCR), has been used many times for a wide variety of applications including traffic sign recognition in automated driving applications [9]; assistive technology for the visually impaired [10]; and even in traffic cameras for automatic licence plate recognition [11]. OCR, at the time of this writing, is well understood. Every OCR application currently involves data presented in a standard format. Written language has a certain format and explicit rules that must be followed, licence plates have specific formats, and even data entry tasks that have been automated rely on the specific application having external knowledge of the format the data will take for their specific task [12]. This research seeks to go further with OCR and create an application that can take the text from the image and standardise it. The examples will all have the same data, but the information will be presented in wildly different ways from one another. The goal of this standardisation is to be able to automate more processes such as business expense calculations, insurance claims, and wider data entry applications.

Originally the sample data intended to be used for this work was health insurance claims. Since working with a health insurance provider would yield an abundance of claim data, and all the data would have already been labelled as part of the work those companies are already doing. After considering the potential hurdles of working with a private company, and especially of using confidential healthcare information, the example data this research will utilise consists of restaurant receipts. Restaurant receipts have common information on them, but each restaurant (or chain) will organise and present the information differently. The actual organisation of the receipts will differ (item name above, price below vs. item name left, price right) and the labels will differ (TOTAL, FINAL TOTAL, GRAND TOTAL, AMOUNT DUE, etc. are all possible labels with the same meaning) [13]. Instead of tracking provider number, service types, procedure codes, etc. from a health insurance claim; the restaurant receipts have total and subtotal common across all receipts. The interesting thing about these two features is that subtotal is not always explicitly marked on the receipt. If a receipt does not have a subtotal field or a tax field, the implication is that the subtotal must be equal to the total.

Receipts are complicated, yet people are often able to interpret the important information on a receipt within seconds of glancing at one. Aspects of the text like font, size, location, spacing, etc. are critical to human understanding of receipts. Consider the receipt on the left of figure 1 with the same information on the right without formatting, how long does it take you to identify the total in each case?

DAN KEANEY AND ASSOCIATES IN. 1191 LANSIGN AVE UNIT 3 SUDBURY ON CARD **********7638 CARD TYPE VISA DATE 2022 / 04 / 06 TIME 0376 18:03:29 RECEIPT NUMBER H84008377 -001-001-585-0 PURCHASE TOTAL $135.00 VISA CREDIT 10000000031010 1E1011B303A290B3 000000000000 - APPROVED AUTH# 98937 01-027 THANK YOU NO SIGNATURE REQUIRED CARDHOLDER COPY IMPORTANT - RETAIN THIS COPY FOR YOUR RECORDS

Figure 1. Receipt image (left), same information but without formatting (right).

For many people the formatting of the receipt in Figure 1 is important to easily identify important information such as the total on the receipt [14]. Based on these observations, it seems like a good starting point to consider that positional and formatting data might be integral to designing a machine that can successfully interpret receipt information. Metadata (data about the data) must be included with the receipt text in order to facilitate training and give the system the same advantages humans have in identifying important information on receipts.

An additional consideration for researchers when starting out on this problem is how to approach the idea of data standardisation on something like a shopping receipt. The first distinction is whether to treat this as a classification problem or a regression problem. Classification problems suffer from needing predefined categories to classify training examples into - which means there must be a category for every total if that's the important information being identified. Since receipts vary widely in totals, even for a conservative model that worked only for most receipts instead of all, it would require thousands of categories [15]. Not only are the number of categories potentially problematic, but a classification solution would require many examples of the same category, so many distinct examples for every supported total on the receipt. Regression problems, on the other hand, suffer from other challenges, mainly that they produce approximated solutions that are accepted when within a certain error tolerance [15]. We decided

treating this as a regression problem was more feasible in industry, as being accurate within $0.01 is negligible, and a larger error margin may still be acceptable if the system was a cost-saving measure in other ways like efficiency and accuracy over human workers.

Given the decision to treat this as a regression problem, the next hurdle was deciding on the type of machine learning model to use. Nonstandard data would depend on contextual clues surrounding it, and so a series of datapoints would need to be passed into the model for it to be able to understand the context of each token being processed. Since the order of the sequence was important to preserve, the decision was to use a recurrent neural network (RNN) for training. An RNN can process new information incrementally while maintaining an internal model of what it is processing, built from past information and constantly updated as new information comes in much like how humans process information [16]. The idea is that the neural network will take in each piece of information about the receipt (data on the receipt as well as metadata about each of those datapoints) and keep track of the context as it reads through. That way – the total should be identifiable once the entire receipt has been read regardless of the formatting of that particular receipt. After training on a sufficient number of examples, the model should learn to identify the total on any receipt – even those whose formats have not been seen during training.

3. EXPERIMENTAL SETUP

A publicly available dataset that contained common data in a non-standardised format is the Consolidated Receipt Dataset for post-OCR parsing (CORD) [17] It contains 1,000 sample images of food receipts local to Indonesia. The dataset also contains 1,000 matched JSON files that contain the OCR'ed data and metadata of the information on those images. Clova AI assembled the CORD dataset, including both the images and the OCR-produced JSON files. Clova AI has seen fit to obscure some of the information present on the receipts in order to avoid legal issues with the Indonesian government [18]. Figure 2 is an example of paired image and JSON files from the dataset. Many of the images are imperfect from the point of view of image processing and OCR, but one of the reasons the CORD dataset seemed ideal to researchers is the OCR was performed on images that are similar to what users would submit to an insurance company or an accounting firm. Users will submit imperfect images, so it's best if the process from image to standardised data works on less-than-ideal images. Figure 2 shows a matched set of an image and a JSON file, both provided as part of the CORD dataset. To understand how they work, note the word "Extra" in the image. In the JSON file the line *"text": "Extra"* refers to that token recognised by the OCR during processing. The X and Y values in the lines above are the coordinates of the bounding box of that token on the receipt, identified during the OCR processing. (x1, y1) are the coordinates of the top left corner of the bounding box, (x2, y2) are the coordinates of the top right corner, (x3, y3) the bottom right corner, and (x4, y4) the bottom left corner. Those coordinates confer the information of the position of the text, but also the size of the text. Both of those factors might be important metadata in determining what the total of the receipt is.

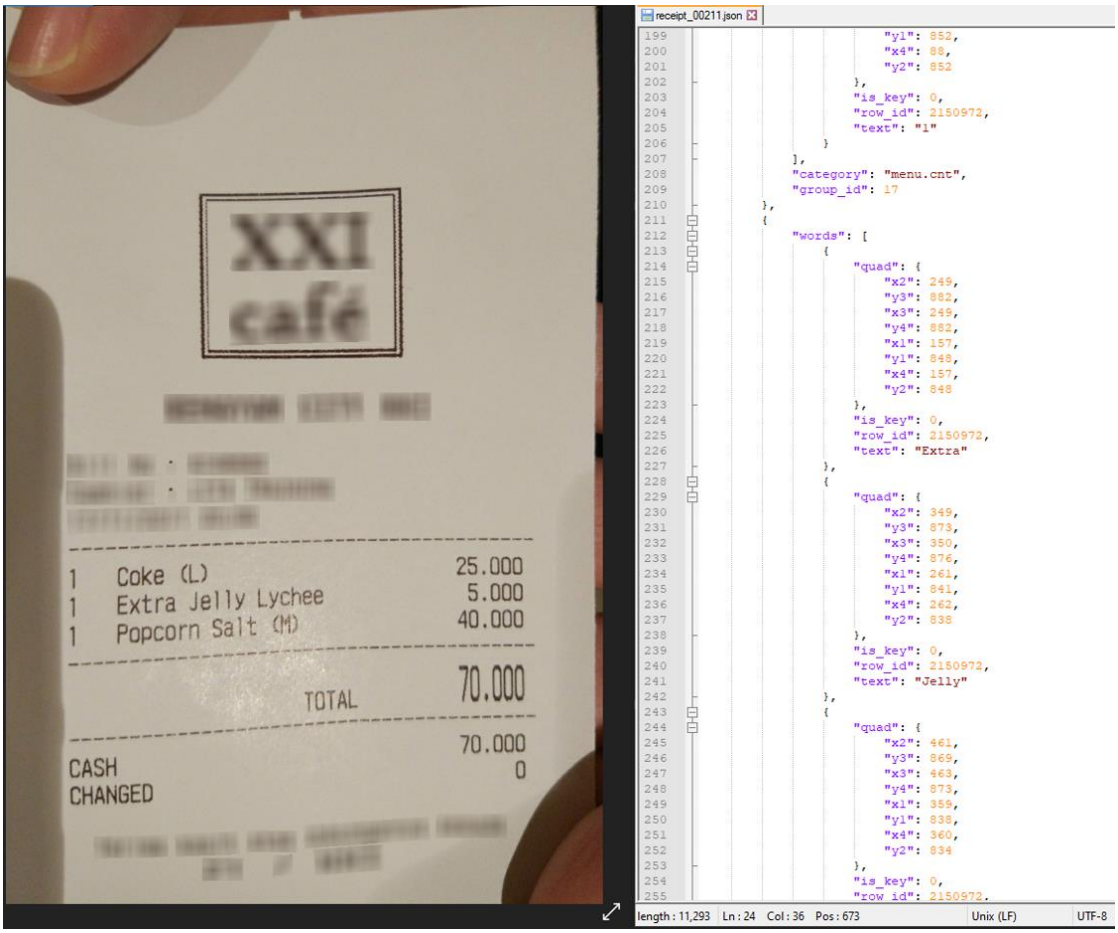


Figure 2. CORD receipt image (left) and associated CORD JSON file (right)

As useful as the CORD dataset is as a repository of common data with non-standard formatting, there were some issues in the data that needed to be pre-processed out. First, the OCR is only as good as the original receipts. After a number of unsuccessful attempts at extracting the total from the JSON files, we realised that the label they had manually assigned to an example wasn't always the same as the CORD JSON files, and the labels all needed to be adjusted to match the JSON version of the total. That was determined to be preferable to adjusting the CORD data to match the actual totals on the images, as it might introduce some unforeseen error in the data and would take significantly longer. As an example, the OCR total on the receipt pictured in figure 3 should be 23,000 Indonesian rupiahs, which is worth approximately 2 Canadian dollars. However, since the receipt separated the digits with a period, the OCR and the machine learning model were reading the total as 23.0 Indonesian rupiahs, about a fifth of a Canadian cent. These types of issues were identified in over 200 of the original 1,000 CORD examples – all but 18 were able to be corrected, with the outstanding examples being left out of the dataset for training.

Figure 3. Example CORD image that did not have matching OCR total to manually identified total.

The next most obvious problem to correct was that the CORD JSON files were much longer than necessary. The first pass at pairing down the JSON files reduced the length of the files by approximately 95%, regarding the number of lines. Each line in the new "trimmed" JSON files consisted of eight integers each separated by a comma and space, ending in a string. The string is some text identified on the image and the eight numbers are coordinates of the bounding box of that text on the receipt image. As a result, each line had the format "X1, X2, X3, X4, Y1, Y2, Y3, Y4, TEXT" where the x and y values are directly transcribed from the coordinates of the corners of the bounding box identified in the original CORD JSON file. Keeping the important metadata, while getting rid of all the unnecessary formatting was essential to bringing the files down to manageable lengths. Instead of 388 total lines in the JSON file pictured in figure 4, that particular example was reduced to only 20 lines; with each line corresponding to a single token identified on the receipt. 8 integers which represent bounding box coordinates and a string token.

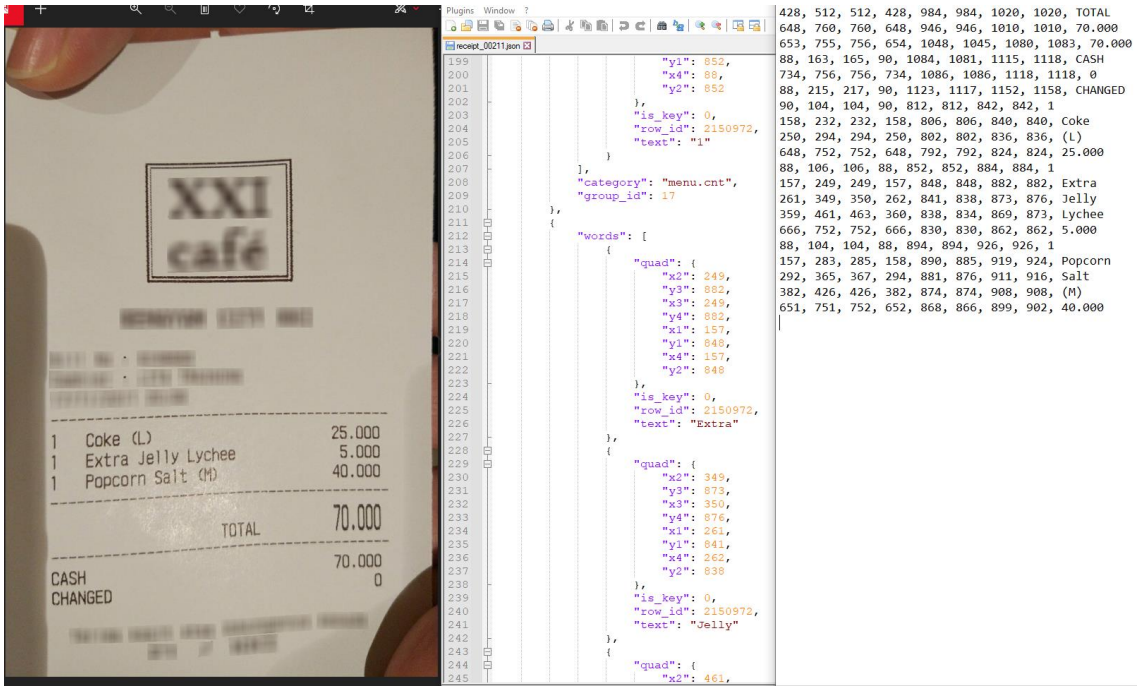
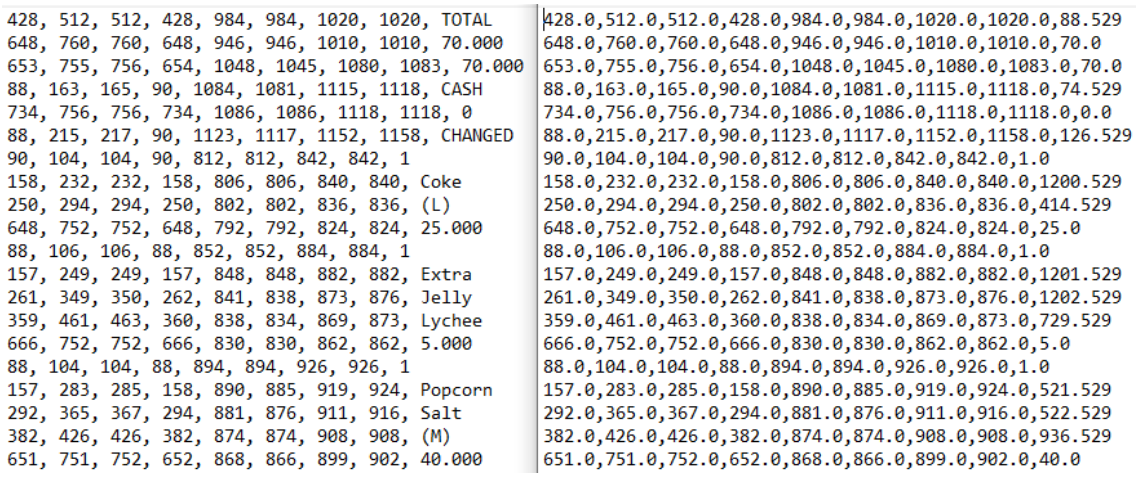


Figure 4. CORD receipt image (left), associated CORD JSON file (centre), associated trimmed JSON file (right)

Other formats were attempted in order to get examples which were as short as possible without losing any information. Firstly, the string values on the receipts were mapped to integer values which correspond to an index in a dictionary created as files were being processed. This is because a regression model will only operate on numerical values. Since the string information being translated into numerical data was only valuable to provide context for the actual numbers on the receipts, it was decided to leave the numbers on the receipts untransformed. To make a clear distinction between untransformed numerical data on the receipts and strings converted into numbers using the dictionary indices, each transformed integer was converted to a floating-point number with an arbitrary ".529" appended to the number, which holds no meaning other than it is never found on the actual numerical data in any of the receipts. These modifications change the file from the rightmost file in figure 4 to the file in figure 5, which both correspond to the same sample file from the original CORD dataset. The coordinates are unchanged, though have been converted to floating point values, and the data from the receipts have either been converted to floating point numbers from strings, which end in ".529" or have remained unchanged. The dictionary values are arbitrary, in that they don't signify anything other than the order tokens were identified in the pre-processing step; however, each word is only tokenized once, meaning every subsequent instance of that same string token will be converted into the same numeric value.

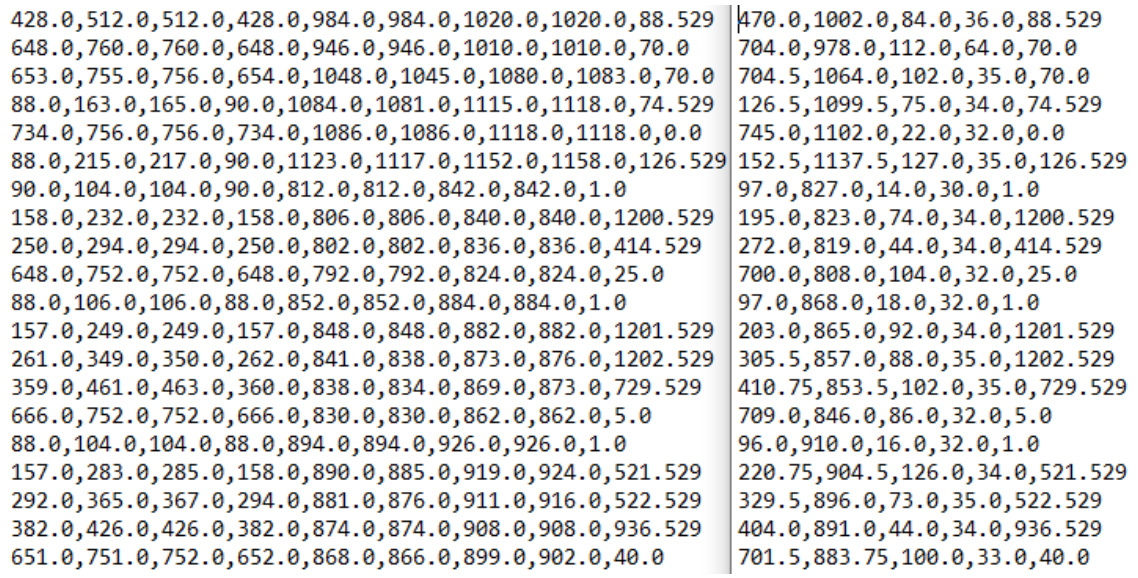


```
428, 512, 512, 428, 984, 984, 1020, 1020, TOTAL
648, 760, 760, 648, 946, 946, 1010, 1010, 70.000
653, 755, 756, 654, 1048, 1045, 1080, 1083, 70.000
88, 163, 165, 90, 1084, 1081, 1115, 1118, CASH
734, 756, 756, 734, 1086, 1086, 1118, 1118, 0
88, 215, 217, 90, 1123, 1117, 1152, 1158, CHANGED
90, 104, 104, 90, 812, 812, 842, 842, 1
158, 232, 232, 158, 806, 806, 840, 840, Coke
250, 294, 294, 250, 802, 802, 836, 836, (L)
648, 752, 752, 648, 792, 792, 824, 824, 25.000
88, 106, 106, 88, 852, 852, 884, 884, 1
157, 249, 249, 157, 848, 848, 882, 882, Extra
261, 349, 350, 262, 841, 838, 873, 876, Jelly
359, 461, 463, 360, 838, 834, 869, 873, Lychee
666, 752, 752, 666, 830, 830, 862, 862, 5.000
88, 104, 104, 88, 894, 894, 926, 926, 1
157, 283, 285, 158, 890, 885, 919, 924, Popcorn
292, 365, 367, 294, 881, 876, 911, 916, Salt
382, 426, 426, 382, 874, 874, 908, 908, (M)
651, 751, 752, 652, 868, 866, 899, 902, 40.000
```

```
428.0,512.0,512.0,428.0,984.0,984.0,1020.0,1020.0,88.529
648.0,760.0,760.0,648.0,946.0,946.0,1010.0,1010.0,70.0
653.0,755.0,756.0,654.0,1048.0,1045.0,1080.0,1083.0,70.0
88.0,163.0,165.0,90.0,1084.0,1081.0,1115.0,1118.0,74.529
734.0,756.0,756.0,734.0,1086.0,1086.0,1118.0,1118.0,0.0
88.0,215.0,217.0,90.0,1123.0,1117.0,1152.0,1158.0,126.529
90.0,104.0,104.0,90.0,812.0,812.0,842.0,842.0,1.0
158.0,232.0,232.0,158.0,806.0,806.0,840.0,840.0,1200.529
250.0,294.0,294.0,250.0,802.0,802.0,836.0,836.0,414.529
648.0,752.0,752.0,648.0,792.0,792.0,824.0,824.0,25.0
88.0,106.0,106.0,88.0,852.0,852.0,884.0,884.0,1.0
157.0,249.0,249.0,157.0,848.0,848.0,882.0,882.0,1201.529
261.0,349.0,350.0,262.0,841.0,838.0,873.0,876.0,1202.529
359.0,461.0,463.0,360.0,838.0,834.0,869.0,873.0,729.529
666.0,752.0,752.0,666.0,830.0,830.0,862.0,862.0,5.0
88.0,104.0,104.0,88.0,894.0,894.0,926.0,926.0,1.0
157.0,283.0,285.0,158.0,890.0,885.0,919.0,924.0,521.529
292.0,365.0,367.0,294.0,881.0,876.0,911.0,916.0,522.529
382.0,426.0,426.0,382.0,874.0,874.0,908.0,908.0,936.529
651.0,751.0,752.0,652.0,868.0,866.0,899.0,902.0,40.0
```

Figure 5. Trimmed CORD JSON file (left), associated integer-translated file (right)

The last major shift to the original CORD JSON files was an attempt to reduce the file lengths, in case long sequences were causing vanishing gradient issues that caused training to be unsuccessful. Instead of including the coordinates of the four corners of the bounding box for each piece of data identified in the receipts, we identified the centre of the bounding box with coordinates, followed by the height and the width of the data. This preserves the metadata of the size and position of the text but reduces the number of tokens in the sequences by almost half in every file. An example is shown in figure 6.



```
428.0,512.0,512.0,428.0,984.0,984.0,1020.0,1020.0,88.529
648.0,760.0,760.0,648.0,946.0,946.0,1010.0,1010.0,70.0
653.0,755.0,756.0,654.0,1048.0,1045.0,1080.0,1083.0,70.0
88.0,163.0,165.0,90.0,1084.0,1081.0,1115.0,1118.0,74.529
734.0,756.0,756.0,734.0,1086.0,1086.0,1118.0,1118.0,0.0
88.0,215.0,217.0,90.0,1123.0,1117.0,1152.0,1158.0,126.529
90.0,104.0,104.0,90.0,812.0,812.0,842.0,842.0,1.0
158.0,232.0,232.0,158.0,806.0,806.0,840.0,840.0,1200.529
250.0,294.0,294.0,250.0,802.0,802.0,836.0,836.0,414.529
648.0,752.0,752.0,648.0,792.0,792.0,824.0,824.0,25.0
88.0,106.0,106.0,88.0,852.0,852.0,884.0,884.0,1.0
157.0,249.0,249.0,157.0,848.0,848.0,882.0,882.0,1201.529
261.0,349.0,350.0,262.0,841.0,838.0,873.0,876.0,1202.529
359.0,461.0,463.0,360.0,838.0,834.0,869.0,873.0,729.529
666.0,752.0,752.0,666.0,830.0,830.0,862.0,862.0,5.0
88.0,104.0,104.0,88.0,894.0,894.0,926.0,926.0,1.0
157.0,283.0,285.0,158.0,890.0,885.0,919.0,924.0,521.529
292.0,365.0,367.0,294.0,881.0,876.0,911.0,916.0,522.529
382.0,426.0,426.0,382.0,874.0,874.0,908.0,908.0,936.529
651.0,751.0,752.0,652.0,868.0,866.0,899.0,902.0,40.0
```

```
470.0,1002.0,84.0,36.0,88.529
704.0,978.0,112.0,64.0,70.0
704.5,1064.0,102.0,35.0,70.0
126.5,1099.5,75.0,34.0,74.529
745.0,1102.0,22.0,32.0,0.0
152.5,1137.5,127.0,35.0,126.529
97.0,827.0,14.0,30.0,1.0
195.0,823.0,74.0,34.0,1200.529
272.0,819.0,44.0,34.0,414.529
700.0,808.0,104.0,32.0,25.0
97.0,868.0,18.0,32.0,1.0
203.0,865.0,92.0,34.0,1201.529
305.5,857.0,88.0,35.0,1202.529
410.75,853.5,102.0,35.0,729.529
709.0,846.0,86.0,32.0,5.0
96.0,910.0,16.0,32.0,1.0
220.75,904.5,126.0,34.0,521.529
329.5,896.0,73.0,35.0,522.529
404.0,891.0,44.0,34.0,936.529
701.5,883.75,100.0,33.0,40.0
```

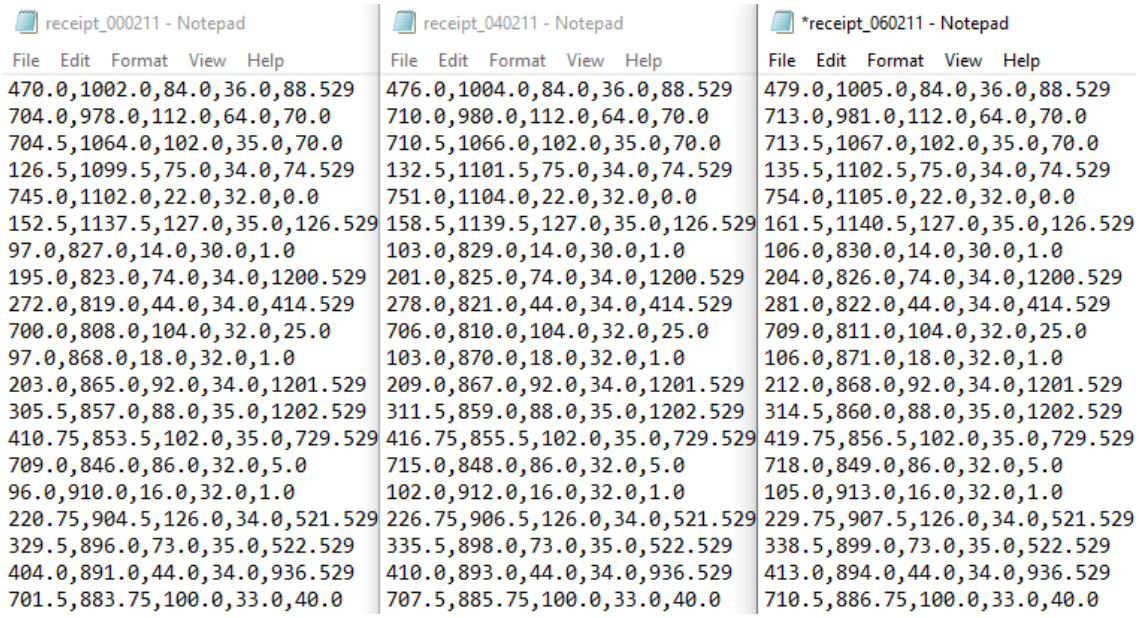
Figure 6. CORD Integer-translated file (left), associated shortened file (right)

4. EXPERIMENTAL WORK AND RESULTS

We found it reasonable to start by overfitting an algorithm to the sample data, and work on reducing overfitting as much as possible from that point. With that in mind, and with an open mind on what specific model setup might achieve the best results for this uncommon machine learning task, a python script was developed that was malleable at runtime in order to populate the model with many different varieties of layers, activation functions, computation nodes, loss

functions, optimizers, epochs, batch sizes, learning rate, dropout, and file changes. The script pulls the appropriate pre-processed files and creates the model specified by user input, then trains for the given number of epochs. After training, the model makes predictions on a small number of examples, some from the training data and some from holdout data not used for training, and we evaluate how the predictions line up with expectations. One would expect predictions to be near perfect for the training examples, and be wildly inaccurate for the holdout data, at least initially when overfitting was taking place as expected. Unfortunately, the model never moved past making the same static prediction after every run. The number would be different every time a new model was created and trained, even if all the parameters were the same as a previous model, but the model would always output some static number as its prediction of the total on a receipt regardless of the training parameters.

As a result of this, hundreds of experiments were performed, with many changes to the model's structure in an attempt to find the right configuration to produce some reasonable results, since overfitting to the training data was expected. After many tests without significant improvement, it was determined that the small number of available examples might be holding back the model. The original 1,000 CORD examples were the most abundant dataset we could find for files that contained common data in non-standardised presentation, but 1,000 training examples is not an abundance of data from the point of view of machine learning in general. To overcome this problem, examples were augmented in a very careful way, because the relative position of labels and corresponding numerical data was considered important to the model's training, and size of the text on the receipts was considered important to the model's training. So that meant we couldn't reasonably modify the data on the receipt, nor the height / width of some elements of the receipt and not others, nor the position of some elements of the receipt and not others. The choice was made to augment the data by modifying the positions of every datapoint on a given receipt in the same way. Relative positions are protected, as is everything else that was deemed important to the training but having the positions change should be a significant enough change that the model will be able to use the new example and learn from it, not so similar that we're just showing the same example multiple times.



receipt_000211 - Notepad

File Edit Format View Help

```
470.0,1002.0,84.0,36.0,88.529
704.0,978.0,112.0,64.0,70.0
704.5,1064.0,102.0,35.0,70.0
126.5,1099.5,75.0,34.0,74.529
745.0,1102.0,22.0,32.0,0.0
152.5,1137.5,127.0,35.0,126.529
97.0,827.0,14.0,30.0,1.0
195.0,823.0,74.0,34.0,1200.529
272.0,819.0,44.0,34.0,414.529
700.0,808.0,104.0,32.0,25.0
97.0,868.0,18.0,32.0,1.0
203.0,865.0,92.0,34.0,1201.529
305.5,857.0,88.0,35.0,1202.529
410.75,853.5,102.0,35.0,729.529
709.0,846.0,86.0,32.0,5.0
96.0,910.0,16.0,32.0,1.0
220.75,904.5,126.0,34.0,521.529
329.5,896.0,73.0,35.0,522.529
404.0,891.0,44.0,34.0,936.529
701.5,883.75,100.0,33.0,40.0
```

receipt_040211 - Notepad

File Edit Format View Help

```
476.0,1004.0,84.0,36.0,88.529
710.0,980.0,112.0,64.0,70.0
710.5,1066.0,102.0,35.0,70.0
132.5,1101.5,75.0,34.0,74.529
751.0,1104.0,22.0,32.0,0.0
158.5,1139.5,127.0,35.0,126.529
103.0,829.0,14.0,30.0,1.0
201.0,825.0,74.0,34.0,1200.529
278.0,821.0,44.0,34.0,414.529
706.0,810.0,104.0,32.0,25.0
103.0,870.0,18.0,32.0,1.0
209.0,867.0,92.0,34.0,1201.529
311.5,859.0,88.0,35.0,1202.529
416.75,855.5,102.0,35.0,729.529
715.0,848.0,86.0,32.0,5.0
102.0,912.0,16.0,32.0,1.0
226.75,906.5,126.0,34.0,521.529
335.5,898.0,73.0,35.0,522.529
410.0,893.0,44.0,34.0,936.529
707.5,885.75,100.0,33.0,40.0
```

*receipt_060211 - Notepad

File Edit Format View Help

```
479.0,1005.0,84.0,36.0,88.529
713.0,981.0,112.0,64.0,70.0
713.5,1067.0,102.0,35.0,70.0
135.5,1102.5,75.0,34.0,74.529
754.0,1105.0,22.0,32.0,0.0
161.5,1140.5,127.0,35.0,126.529
106.0,830.0,14.0,30.0,1.0
204.0,826.0,74.0,34.0,1200.529
281.0,822.0,44.0,34.0,414.529
709.0,811.0,104.0,32.0,25.0
106.0,871.0,18.0,32.0,1.0
212.0,868.0,92.0,34.0,1201.529
314.5,860.0,88.0,35.0,1202.529
419.75,856.5,102.0,35.0,729.529
718.0,849.0,86.0,32.0,5.0
105.0,913.0,16.0,32.0,1.0
229.75,907.5,126.0,34.0,521.529
338.5,899.0,73.0,35.0,522.529
413.0,894.0,44.0,34.0,936.529
710.5,886.75,100.0,33.0,40.0
```

Figure 7. CORD shortened file, augmented by increasing X and Y

Since the coordinates didn't contain the total area of the original image, there was no upper bound on the coordinates for the data within each example. By incrementing the x, and y coordinates - the original 1,000 training examples was augmented up to 80,000 examples originally, then 500,000 examples for good measure, to ensure that the amount of training examples was not a limiting factor in the model training.

The following is an example model that demonstrates the type of training undertaken for this research. Tensorflow [19] is used to train models, which uses underlying the Keras [20] architecture. This particular model was trained with three Long-Short Term Memory (LSTM) [21] layers, then thirty Dense layers with rectified linear unit activation functions. Each LSTM layer had 512 computing nodes and each Dense layer had 256 nodes. The model used the Adam optimizer and used mean absolute error to measure loss. The learning rate was constrained to 0.0001 to prevent overshooting the solution during optimization, and dropout of 0.33 was applied in between each layer to increase variance and attempt to prevent the model from falling into a local minimum during optimization. 20 Epochs were chosen as being a sufficient amount of training time to determine if the model would learn anything – measured by observing if the loss function plateaued or continued to drop as epochs continued. Most tests started plateauing after only 10 epochs, so if a model didn't show any plateauing after 20 it was a good candidate for further study. The input shape of the model is (135, 1) and between each layer the shape changes to (135, 256) to allow the model many variables to adjust that might affect the overall learning. The very last Dense layer of the model has a linear activation function and output shape of (1), since we're only measuring the total in this experiment – no other variables. Following training, which is all logged to track changes in loss, predictions are made on twenty examples, ten from the training dataset and ten from a holdout set not used on training. Accuracy is measured by how close the predicted total is to the actual total on the label.

5. ANALYSIS OF RESULTS

After the data augmentation and studious retesting with ambitious model configurations, the obtained results were not satisfactory to be used in real commercial applications. It is believed that the size of the training data is very small to properly train a deep neural network. Even though the data augmentation stage has created 80,000 samples, the decision to keep relative position and size consistent, as well as keeping receipt data unmodified, may have resulted in augmented data being too similar to the original data. As a result, the model has been trying to train on only 1,000 training examples - clearly too few examples for most machine learning algorithms. In order for the proposed model to provide satisfactory results, it is essential to increase the number of independent training samples to give the neural network an opportunity to learn. Our main concern at that point was whether it would be possible to train the network within feasible time limits. Therefore, we conducted a set of experiments to identify the relationship between the number of training samples and the average training time per epoch. During these experiments, the augmented data, as described in Section 4, was used. It is believed that the augmented data will have the same influence on the average training time per epoch as the true data. The conducted experiments have demonstrated that the relationship between the number of training samples and the average training time per epoch has a low rate of growth. This is evident from the fact that increasing the number of training samples from 1,000 actual data to 80,000 augmented data has resulted in an increase of the training time by a factor of approximately 43. Figure 8 presents the relationship between the number of training examples and the average training time per epoch. Upon the availability of a larger database of receipts or health insurance claims, the model can produce promising results.

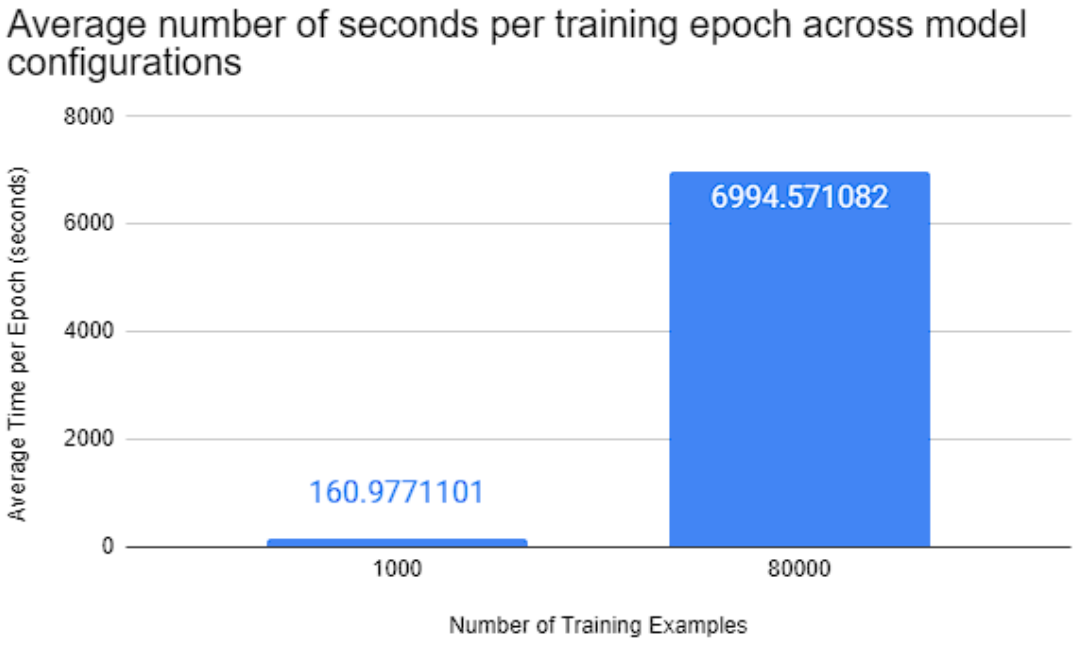


Figure 8. Comparison of the average time per training epoch (in seconds), for experiments run with 1,000 and 80,000 training examples.

6. CONCLUSION AND FUTURE WORK

In conclusion, automatic data standardisation would benefit many industries such as insurance claims processing, accounting, or government documents. The concept of machine learning-enabled automatic data standardisation of physical documents such as receipts seems possible given currently available technologies. The paper presents a general framework for data standardization that can be employed in various applications. As a proof of concept, a deep neural network was built and trained using restaurant receipts images from CORD dataset. A regression model was used to predict the total value on a receipt. Due to the relatively limited number of samples, a data augmentation stage was proposed. The experimental work has demonstrated that with a less restrictive dataset, promising results might be accomplished. Future work in this area includes trying to find a larger data set that can provide better training to the model; approaching this machine learning task as a Natural Language Processing (NLP) task [22], specifically a Named-Entity Recognition task for such applications as finding a total on a receipt; and approaching this as an object-detection (OD) task given that OCR will allow for easy collection of the bounding boxes of all information on the receipt. Work on NLP or OD applications can occur simultaneously with attempting more configurations on the regression version of the task.

REFERENCES

[1] ClaimSecure. (n.d.). PhotoClaims : easiest way to file a claim …. “Quick as a Selfie!” https://www.claimsecure.com/products/Video.aspx?type=photoclaims&lang=en

[2] BodyShop Business Staff Writers. (2017). “500 Allstate Claims Adjusters Being Replaced by Mobile Photo App.” BodyShop Business. https://www.bodyshopbusiness.com/allstate-500-claims-adjusters-replaced-quickfoto-app/.

[3] ClaimSecure. (n.d.). Plan Administrator Guide. ClaimSecure. https://www.claimsecure.com/content/pdfs/en-CA/Forms/admin_hd_en.pdf

[4] Canada Life. (n.d.). Claims procedures for customers with benefits through their employer. Canada Life. https://www.canadalife.com/content/dam/canadalife/documents/forms/you-and-your-family/claims-procedures/en/claims-procedures-for-customers-with-group-benefits.pdf
[5] Manulife. (n.d.). Individual insurance FAQs - Support. Manulife. https://www.manulife.ca/personal/support/insurance/individual-insurance/faqs.html
[6] Telus Health. (n.d.). Kroll Pharmacy Management Solution. TELUS. https://www.telus.com/en/health/health-professionals/pharmacies/kroll
[7] McKesson Canada. (n.d.). Propel Rx. Pharmacy Technology Solutions. https://www.pharmacytechnologysolutions.ca/propel-rx
[8] Dhavale. (2017). Advanced Image-based Spam Detection and Filtering Techniques. IGI Global.
[9] Chauhan, Luthra, & Ansari. (2019). "Road Sign Detection Using Camera for Automated Driving Assistance System." Proceedings of the International Conference on Advances in Electronics. https://ssrn.com/abstract=3573876
[10] American Foundation for the Blind, "Optical Character Recognition Systems.", https://www.afb.org/node/16207/optical-character-recognition-systems#:~:text=Optical%20character%20recognition%20(OCR)%20systems,%2C%20recognition%2C%20and%20reading%20text.
[11] Rosebrock. (2020). "OpenCV: Automatic License/Number Plate Recognition (ANPR) with Python." PyImageSearch. https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/
[12] Lombardi, A. (2020, May 19). How Do Cheque Scanners Work? (Updated 2020). CashTech Currency Products Inc. https://www.cashtechcurrency.com/blog/how-do-cheque-scanners-work
[13] QuickBooks Canada Team. (2018, November 13). Everything You Need to Know About Sales Receipts | QuickBooks Canada. QuickBooks. https://quickbooks.intuit.com/ca/resources/invoicing/sales-receipts/
[14] ConnectsUs HR. (n.d.). Why is proper formatting of a document important? ConnectsUs HR. https://connectsus.com/help/faq/why-proper-formatting-document-important
[15] Goodfellow, Bengio, & Courville. (2016). Deep Learning. MIT Press.
[16] Chollet. (2018). Deep Learning with Python. Manning.
[17] NAVER Corp & Clova AI. (2019, December 26). CORD: A Consolidated Receipt Dataset for Post-OCR Parsing. GitHub. https://github.com/clovaai/cord
[18] Park, et. al. (2019). CORD: A Consolidated Receipt Dataset for Post-OCR Parsing. Conference on Neural Information Processing Systems, 33. https://openreview.net/pdf?id=SJl3z659UH
[19] TensorFlow. (n.d.). Why TensorFlow. TensorFlow. https://www.tensorflow.org/about
[20] Keras. (n.d.). Getting started. Keras. https://keras.io/getting_started/
[21] Brownlee. (2017, May 24). A Gentle Introduction to Long Short-Term Memory Networks by the Experts. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
[22] IBM Cloud Education. (2020, July 2). What is Natural Language Processing? IBM. https://www.ibm.com/cloud/learn/natural-language-processing