DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Using Natural Language Processing to extract information from receipt text

MARKO LAZIC

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Using Natural Language Processing to extract information from receipt text

MARKO LAZIC

Master in Computer Science
Date: June 11, 2020
Supervisor: Mårten Björkman
Examiner: Danica Kragic Jensfelt
School of Electrical Engineering and Computer Science
Host company: Bontouch

Abstract

The ability to automatically read, recognize, and extract different information from unstructured text is of key importance to many areas. Most research in this area has been focused on scanned invoices. This thesis investigates the feasibility of using natural language processing to extract information from receipt text. Three different machine learning models, BiLSTM, GCN, and BERT, were trained to extract a total of 7 different data points from a dataset consisting of 790 receipts. In addition, a simple rule-based model is built to serve as a baseline. These four models were then compered on how well they perform on different data points. The best performing machine learning model was BERT with an overall F1 score of 0.455. The second best machine learning model was BiLSTM with the F1 score of 0.278 and GCN had the F1 score of 0.167. These F1 scores are highly affected by the low performance on the product list which was observed with all three models. BERT showed promising results on vendor name, date, tax rate, price, and currency. However, a simple rule-based method was able to outperform the BERT model on all data points except vendor name and tax rate. Receipt images from the dataset were often blurred, rotated, and crumbled which introduced a high OCR error. This error then propagated through all of the steps and was most likely the main reason why machine learning models, especially BERT were not able to perform. It is concluded that there is potential in using natural language processing for the problem of information extraction. However, further research is needed if it is going to outperform the rule-based models.

Sammanfattning

Förmågan att automatiskt läsa, känna igen och utvinna information från ostrukturerad text har en avgörande betydelse för många områden. Majoriteten av den forskning som gjorts inom området har varit inriktad på inskannade fakturor. Detta examensarbete undersöker huruvida språkteknologi kan användas för att utvinna information från kvittotext. Tre olika maskininlärningsmodeller, BiLSTM, GCN och BERT, tränades på att utvinna totalt 7 olika datapunkter från ett dataset bestående av 790 kvitton. Dessutom byggdes en enkel regelbaserad modell som en referens. Dessa fyra modeller har sedan jämförts på hur väl de presterat på de olika datapunkterna. Modellen som gav bäst resultat bland maskininlärningsmodellerna var BERT med F1-resultatet 0.455. Den näst bästa modellen var BiLSTM med F1-resultatet 0.278 medan GCN hade F1-resultat 0.167. Dessa resultat påverkas starkt av den låga prestandan på produktlistan som observerades med alla tre modellerna. BERT visade lovande resultat på leverantörens namn, datum, moms, pris och valuta. Dock hade den regelbaserade modellen bättre resultat på alla datapunkter förutom leverantörens namn och moms. Kvittobilder från datasetet är ofta suddiga, roterade och innehåller skrynkliga kvitton, vilket resulterar i ett högt fel hos maskinläsningverktyget. Detta fel propagerades sedan genom alla steg och var troligen den främsta anledningen till att maskininlärningsmodellerna, särskilt BERT, inte kunde prestera. Sammanfattningsvis kan slutsatsen dras att användandet av språkteknologi för att utvinna information från kvittotext har potential. Ytterligare forskning behövs dock om det ska användas istället för regelbaserade modeller.

Contents

Chapter 1

Introduction

Document analysis and recognition is a very interesting and active research field. The ability to automatically read, recognize, and extract key information in different types of documents is of key importance to many areas. Many businesses deal with a lot of paper documents like receipts or invoices on a daily basis [1]. Most of these documents have to be processed manually which is both time consuming and expensive [2]. Much research has been done to investigate how to extract the most important fields like date, name, and the total price from scanned invoices. Some, but not as much research has been done on receipts. That research exclusively includes fields like date, address, and total price. It would be of great interest to investigate if some of these methods can be used to recognize and extract the information about the products bought with their respective price and amount.

1.1 Problem definition

This degree project will cover the comparison of several different approaches to solving the problem of information extraction from an image of a receipt. There will be a number of different features that are going to be extracted like vendor name, date, total price, address, currency, tax rate, and product list with respective amounts and prices. The project will also include some suggestions on how these methods can be improved in order to boost their performance. There are several challenges involved. One of them is that the receipts have a great variability when it comes to placement and form of different features. This trend is especially present in the product list section of the receipt as some of them may also include information about the discounts in addition to the price. Another challenge is that the images are usually not perfect as the

receipts may be blurred, rotated, or crumpled.

1.2 Scope

Figure 1.1 shows the overview of this thesis from the raw receipt image to the final data extraction. The yellow color indicates the parts that are included in the thesis and the blue color indicated the parts that are the main area of focus. Scanning the receipts, binarizing the images, and reading the text are parts that are necessary for the final results, but are not the part of this research. The parts that are included, but less important are filtering the result from the optical character recognition (OCR) engine and rule-based model (steps 3 and 4d in figure 1.1). The rule-based model is implemented to serve as a baseline comparison. The main focus of the thesis is the comparison of three different machine learning methods for data extraction from receipt text.

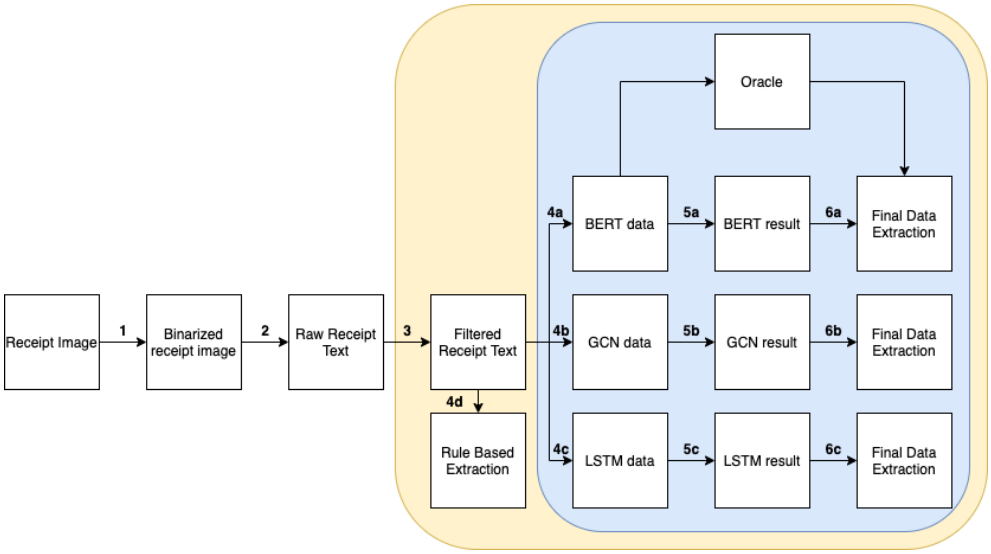


Figure 1.1: Overview of the thesis. Yellow represents parts included in the thesis and blue represents the main area of focus.

1.3 Research Questions

The aim of this thesis is to answer the following questions:

- How well do the BiLSTM, BERT, and GCN perform on data extraction from receipt text in terms of precision, recall, and F1 score?

- Which of these methods is best suited for the extraction of different data fields?
- Can these methods be used to extract information from receipts instead of a rule-based method?

Chapter 2

Background

In this chapter, the theory and background information needed to understand this thesis is presented. This includes three different machine learning models, BiLSTM, BERT, and GCN as well as the measurement used to evaluate the method. Furthermore, the work related to this thesis is presented.

2.1 Natural Language Processing

Natural Language Processing (NLP) is an area of research and application that is focused on exploring how computers can be used to understand and manipulate natural language in text or speech [3]. Applications of NLP include a wide variety of problems like machine translation, natural language text processing, summarization, speech recognition, question answering and named entity recognition and classification.

Named Entity Recognition and Classification is an important subtask of Information Extraction and NLP that aims to classify the extracted named entities like a person, address, date, currency, location, and similar from unstructured text [4].

2.1.1 Recurrent Neural Network

In the area of the NLP the data is very often different in terms of dimensionality. The models have to deal with the documents, text snippets, or sentences of arbitrary sizes. Recurrent Neural Network is a class of Artificial Neural Network which is based on work by Rumelhart et al. [5] and that utilizes its hidden layers to process variable-length sequence of inputs. Figure 2.1 shows a simple model of an RNN network. At each time-step t the hidden layer for

the input x_t is calculated by the formula 2.1. Each hidden layer x_t is calculated based on the current input and the signal propagated from the beginning. RNN uses backpropagation learning rule but when the error signal propagates back in time it tends to blow up or vanish [6].

$$h_t = A(h_{t-1}, x_t) \tag{2.1}$$

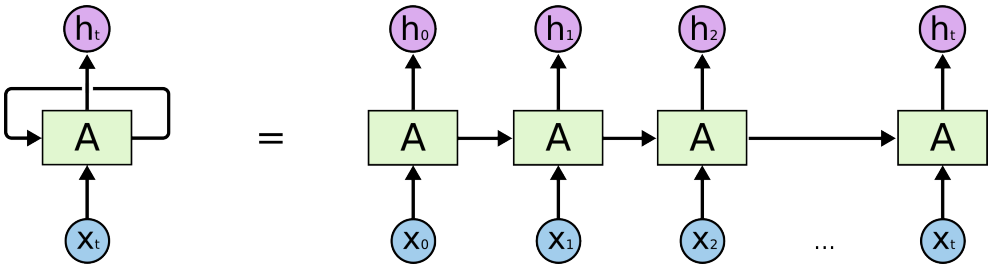


Figure 2.1: The graphical representation of a RNN model [7].

2.1.2 Long Short Term Memory

Long Short Term Memory (LSTM) is a variant of an RNN that solves the problem of blowing and vanishing gradients by introducing a memory cell that is able to preserve the state over long periods of time. Each cell state is controlled by a number of gates that regulate the amount of information the network keeps or discards over the entire input series. This gives the ability to maintain the constant error signal that propagates back over time [6]. Figure 2.2 shows the simple model of an LSTM architecture.

Bidirectional Long Short Term Memory

Standard LSTM processes information in a temporal order. However, it is often beneficial to have access to the future as well as past context [8]. A variant of LSTM called Bidirectional Long Short Term Memory (BiLSTM) allows just that. BiLSTM introduces a second hidden layer where information flows in the opposite temporal order. Figure 2.3 shows a simple 2 layered BiLSTM model. Information flows in two directions at the same time and the outputs from two layers are aggregated at each step containing both the previous and future context.

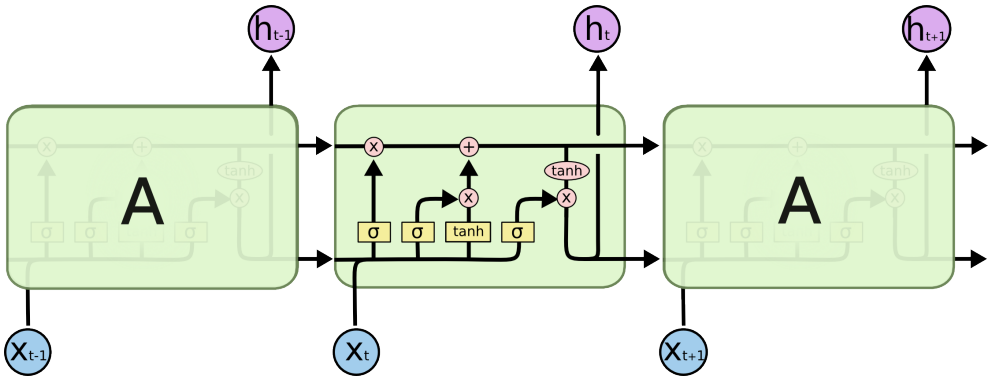


Figure 2.2: The graphical representation of a LSTM model [7].

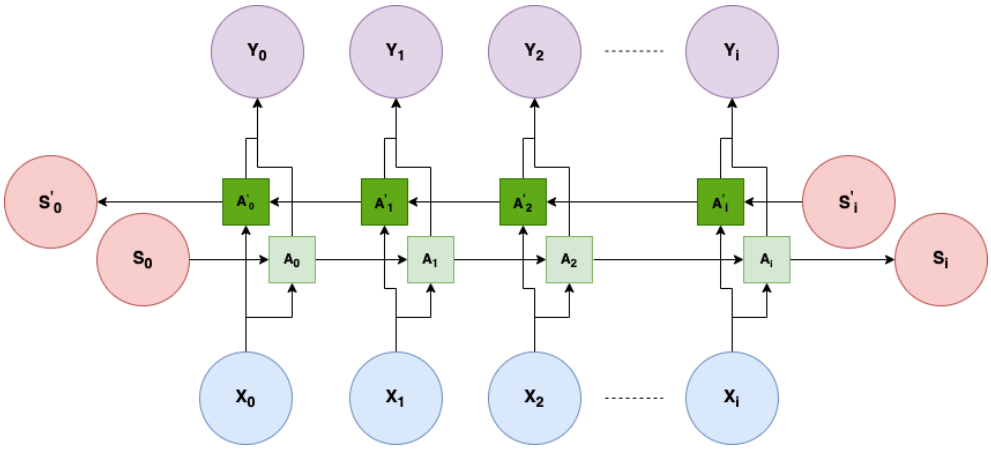


Figure 2.3: The graphical representation of a BiLSTM model.

2.1.3 Sequence To Sequence Models

Sequence to sequence models are introduced for the first time by Sutskever et al. [9]. Their aim was to solve the problem where the input sequence length is not the same as the output sequence length and the traditional LSTM can not be used. These models usually consist of two RNNs; an encoder and a decoder. The encoder takes the input sequences and outputs an encoded vector that aims to encapsulate the information from the input sequence. The decoder takes this input vector and outputs the resulting sequence that does not have to be the same size as the input. Figure 2.4 shows the simple model of an Encoder-Decoder architecture. However, the study by Cho et al. [10] showed that this type of architecture suffers from the problem of not being able to model the long-term dependencies into the encoded vector. It often tends to

forget the important features of the first part of the input as it processes the whole sequence.

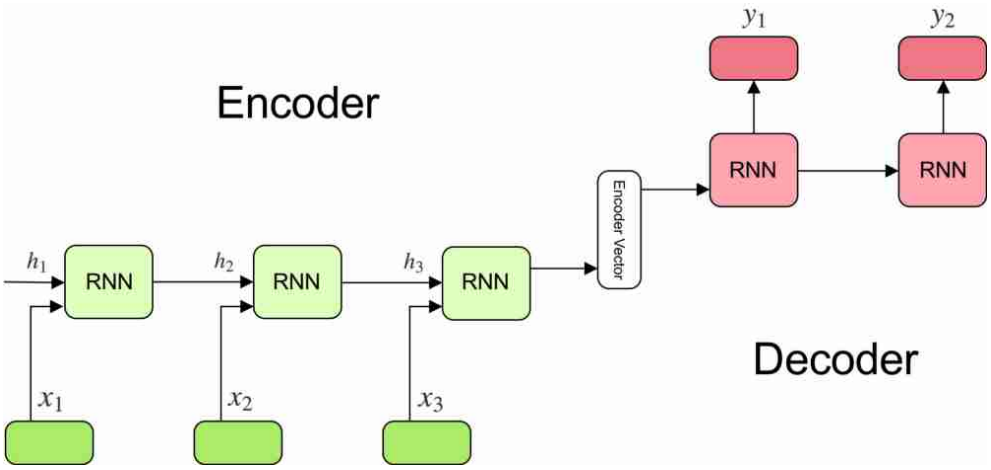


Figure 2.4: The example of an Encoder-Decoder architecture [11].

2.1.4 Attention Mechanisms

Bahdanau et al. [12] introduced the idea of attention mechanism in order to overcome the problem with the traditional encoder-decoder architectures. The attention mechanism allows the decoder to look into the relevant parts of the original sequence when at each generating step instead of relying only on the fixed size encoded vector. This is accomplished by adding an attention layer between the encoder and the decoder which allows the decoder to decide which parts of the input are more relevant for the next prediction.

2.1.5 Transformer

The Transformer is the neural network architecture presented in Vaswani et al. [13] that is solely based on the idea of attention mechanisms. Instead of using the recurrent cells whose value depends on the previous calculation, the whole sequence is fed to the model at the same time. This enables the utilization of parallel computations which are not available for the regular RNNs. In the encoder phase, each element in the input sequence gets a corresponding attention vector which is calculated by the following formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

Vectors Q, K and V (Query, Key, and Value) are calculated using the corresponding weight matrices W_Q, W_K, and W_V which are learnable parts of the encoder. The dimension of queries and values are represented by d_k. The attention vector for a single element in the input sequence contains the information about which are the most relevant parts of the input when looking into this particular element. This particular attention is called Scaled Dot-Product Attention.

However, the work in Vaswani et al. [13] shows that it is beneficial to perform multiple attention operations with different W_Q, W_K, and W_V matrices. This type of attention is called Multi-Head attention. The output of the attention layer is then fed to the regular feed-forward layer which gives the final encoder output.

The decoder part consist of two different attention layers. Firstly, there is a Masked Multi-Head self Attention layer. Since the whole sequence is available to the decoder during the training phase the model has to mask the part of the input so that it can not look into the future words. Otherwise, the model could end up only learning to copy the decoder input. This is then followed by another Multi-Head Attention layer which gets its Query vector from the previous decoder layer while the Key and Value are fetched from the encoder output. Similar to the encoder this is followed by a feed-forward layer. The model of this encoder-decoder architecture is shown in figure 2.5

Positional Encoding

Since the model does not contain any recurrence or convolution the information about the relative and absolute position of each element in the sequence has to be incorporated in some other way. This is accomplished by adding a positional encoding to the input embeddings of both encoder and decoder. Positional encodings have the same dimension as the input embeddings and are calculated by the following formula:

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{2.3}$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 * 2\pi$ [13]. This allows the model to encode the sequence lengths longer than encountered in the training phase.

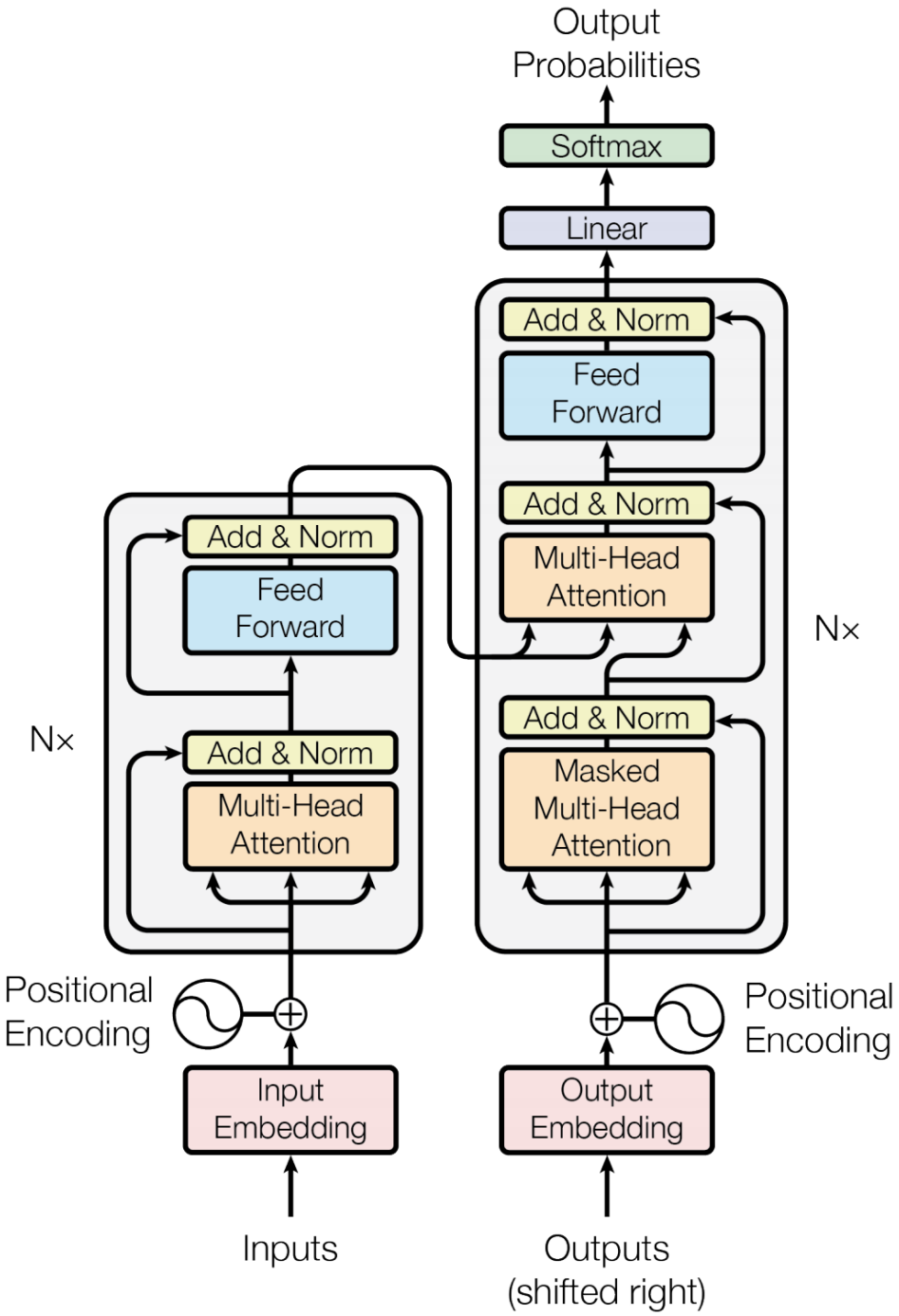


Figure 2.5: Transformer - model architecture [13].

2.1.6 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model proposed by Devlin et al. [14]. BERT is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both left and right context in all layers. It is based on the transformer architecture presented in Vaswani et al. [13]. Figure 2.6 shows the simple model of the BERT architecture consisting of a number of bi-directionally connected transformers. BERT is pre-trained on BooksCorpus (800M words) and English Wikipedia (2.5B words) on two different unsupervised tasks. In the first task, 15% of all word tokens in each sequence are masked at random and the network is then trained to predict them. For the second task, the model is learned to understand the relationship between a two sentences. During the training phase, the model is fed by two sentences A and B, where 50% of the time the B was the actual sentence that followed the sentence A. The model is then learned to decide if the sentence B is a logical successor of the sentence A.

By utilizing the power of transfer learning BERT can be fine-tuned to obtain state-of-the-art results on 11 different NLP tasks [14]. Figure 2.6 shows how the model can be adapted to perform the NER task.

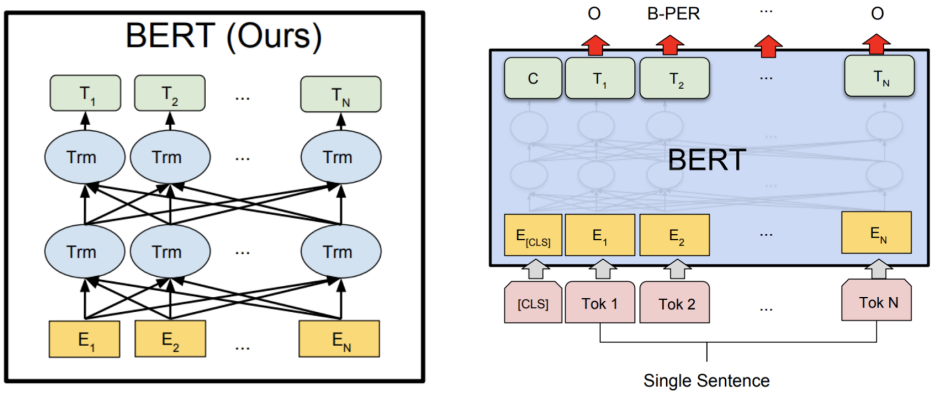


Figure 2.6: Bert model architecture [14].

2.2 Graph Convolution Network

The graph convolution network is a neural network that works directly on graph-structured data. The network's goal is to learn a function that takes as an input a feature vector X_i for every node i and some sort of a structural

description of a graph, usually an adjacency matrix A. The model then produces a node-level output, meaning that it has an output for each node in a graph. Given that a each graph convolution layer is described as a function of a previous one and a adjacency matrix A:

$$H^{l+1} = f(H^l, A) \tag{2.4}$$

then the function $f(H^l, A)$ can be described as:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l) \tag{2.5}$$

with $\hat{A} = A + I$, where I i the identity matrix, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, σ is a non linear activation function and W^l is a weight matrix [15].

2.3 F1 score

To properly evaluate methods some kind of measure is needed to determine how well they perform with respect to some given ground truth. A measure that is commonly used in literature and will also be used here is the F1 score. It is a measure of the test's accuracy used in the statistical analyses of binary classification. The F1 score is the harmonic mean of precision p and recall r. Precision p is the number of correct positive results divided by the number of all positive results returned by the classifier. Recall r is the number of correct positive results divided by the number of all relevant samples [16]. Figure 2.7 shows a visual representation of precision and recall. Equation 2.6 shows the mathematical definition of F1 score.

$$F_1 = 2 * \frac{p * r}{p + r} \tag{2.6}$$

F1 score can be extended for evaluating results for multi-class classification as well. This can be done by either taking an arithmetic mean of each class F1 score (macro averaging) or by simply looking at classes as one and summing all their true positives, actual positives, and false negatives to calculate precision, recall and F1 score (micro averaging).

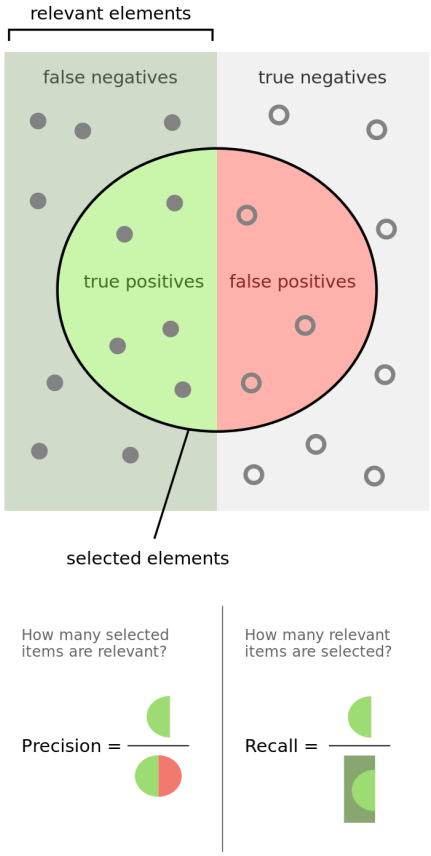


Figure 2.7: Graphical explanation of recall and precision [16].

2.4 Related Work

Extracting information from billing documents is something that has been researched a lot lately. However, the majority of the work has had its focus on invoices rather than receipts. Much of the work that has been applied to invoices can be applied to the receipts as well with a few additional challenges. One of the challenges that receipts bring is that images are often incomplete, blurred, or receipts are crumbled, damaged, or folded. Suponenkovs et al. [17] presents several methods for image pre-possessing that are useful for reading the receipt text and improving the accuracy of an OCR. Another challenge is identifying and cropping the receipt from the image background. A deep learning based method for receipt detection, cropping, and brand logo detection on images taken by an smartphone is proposed by Raoui-Outach et al. [18]. The biggest challenge is the lack of a large publicly available dataset as

the receipts may contain some private information that can not be shared due to GDPR regulations.

The literature on information extraction from receipts and invoices can roughly be divided into three different kinds of approaches. The first one is a template-based approach that analyses the geometrical dependencies between different parts of receipts. This approach tries to utilize the fact that the different key features usually occupy a different specific part of the receipt. The example of this is that the vendor name is usually placed on the top of the receipts while the total price is at the bottom. The second one is based on Named Entity Recognition (NER). This approach tries to understand the semantics of the receipt text and in that way to classify each of its characters or words separately. The third approach is a hybrid solution and a mixture of the first two.

2.4.1 Template structure based models

Many methods for invoice information extraction are based on template classification like [19], [20], [21] and [22]. The method presented by Schuster et al. [21] starts with a k-nearest neighbors (kNN) classifier that finds the nearest matching template from a database of already seen templates. Then, it classifies each field in a new receipt by matching it with a template found in the database. A similar case-based reasoning approach is presented by Hamza et al. [22]. Each invoice is represented by a graph and then compared to the graphs that are already stored in a database to find a most similar one. When a similar graph is found the solution from the stored one in only mapped on the new one. A bit different approach is presented by Rusiñol et al. [20] and d'Andecy et al. [19]. Instead of keeping a database of all different templates seen so far, one structural model that generalizes all of the templates is kept and adjusted each time a new receipt comes in. Figure 2.8 shows an overview of a template-based system. However, all of these methods require that the same or similar template has been seen before and therefore require a large amount of data to be processed or/and stored.

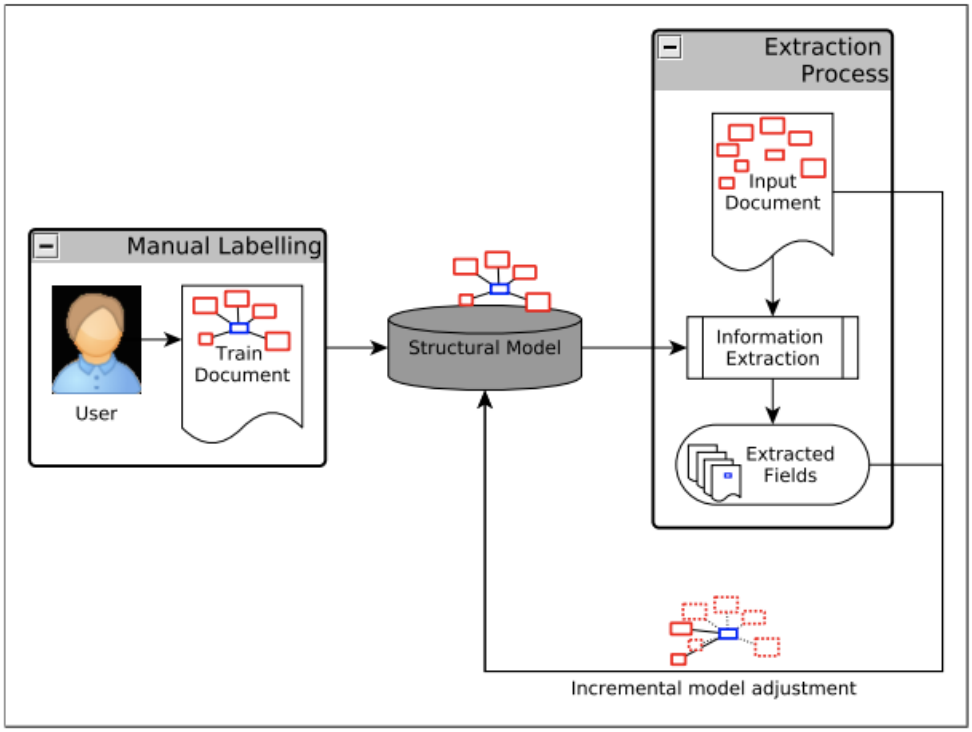


Figure 2.8: An overview of an template based system used by d'Andecy et al. [19]

2.4.2 NER based models

Palm et al. [23] uses a LSTM model for a word classification problem. For each word, a set of features from 3 different categories is calculated: textual, numerical and boolean. Then, a 6-layered RNN that can be seen in Figure 2.9 is used with words from left to right as input. Each word is labeled using IOB labeling and the output is a probability vector of size 65 (32 different classes which can either be marked as "beginning" or "inside" and one "outside" class). The model is then trained on a dataset consisting of 326471 invoices and the overall F1 score was 0.891. The paper also introduces an oracle model that knows all of the ground truth labels but still can not achieve 100% accuracy due to imperfections in OCR.

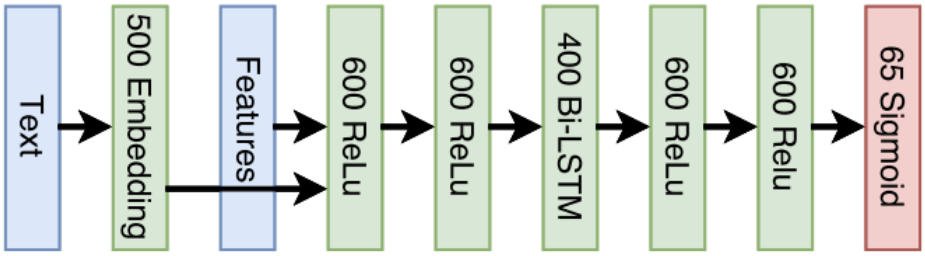


Figure 2.9: The LSTM model used by Palm et al. [23].

2.4.3 Hybrid models

Lohani et al. [24] uses a Graph Convolutional Network (GCN) to solve the classification problem of identifying the different key parts of the scanned invoices. Each word is represented by 3 different feature vectors. The first feature vector is a 13-dimensional boolean vector calculated based on a set of predefined features like isDate, isNumber, isCurrency and similar. The second feature vector is a numeric one whose calculation is based on the relative distance to the nearest neighbors. The third feature vector is a word embedding based on Byte-Pair Encoding (BPE). Word embedding is a process of mapping a variable-sized string representation of words or phrases into a fix sized vector of real numbers so that it can be processed by a computer. Each invoice then gets its own graph where each node represents a single word with its feature vectors and each edge represents the nearest neighbor word in one of the four linear directions. Figure 2.10 shows the resulting graph for a sample invoice. A GCN is then used to label each node of the graph with one of the 28 existing classes. This method is then applied to a dataset consisting of 3100 scanned invoiced and the overall F1 score of 0.93 is obtained.

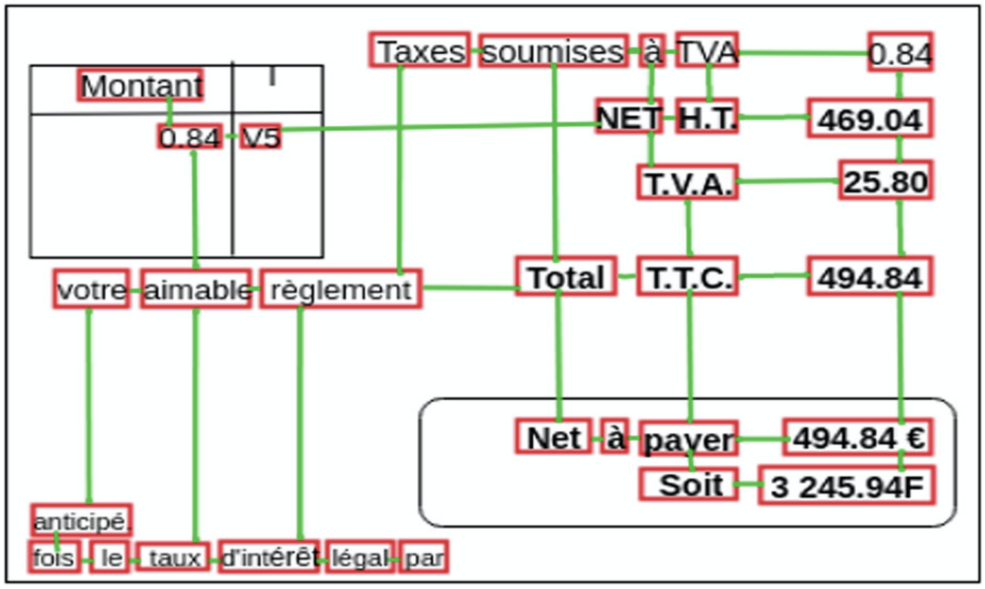


Figure 2.10: An example of graph for a sample invoice used by Lohani et al. [24].

Gal et al. [25] presents a very interesting method that tries to solve the problem of receipt field recognition by taking into account both the semantic and geometrical aspects of the receipt text. Firstly, they train their own model in an unsupervised manner to perform the Char2Vec embedding which is a technique widely used to mapp characters to vectors. As the input to their model, they for each word in the receipt create a word pair consisting of that word and 4 of their nearest neighbors in 4 different directions. Then, for each word in the receipt they encode the output of Char2Vec to a 32-bit long vector and color the bounding box of that word with the result of that vector. Figure 2.11 shows the resulting image after the steps described above. The heatmap is then used as an input to a U-net that tries to learn how to classify each pixel of the colored part of the image. This method is then applied to a dataset consisting of 5094 manually labeled receipts that is extended to a total number of 22670 images by various synthetic image transformations. The final result is only evaluated on the total price field and the best-resulting accuracy was 87.86% with the best dice score of 0.6367.

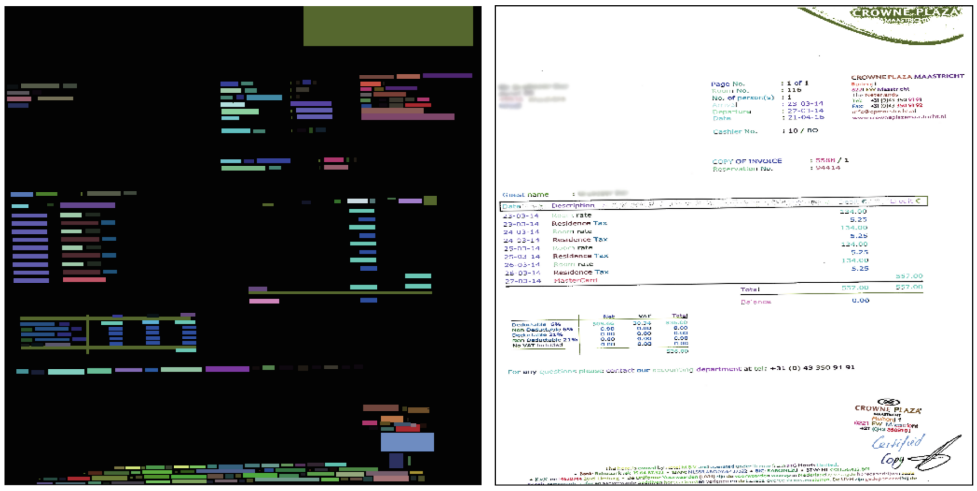


Figure 2.11: Example of the heatmap got from the Char2vec embedding (left) and its mapping on the real image (right) [25].

Chapter 3

Methods

In this chapter, the methods used in this thesis are presented. Firstly, datasets and prepossessing techniques are presented. Next, the rule-based model and the data labeling algorithm is described. Lastly, the three different models and the experiments performed are detailed.

3.1 Dataset

The dataset consists of 790 receipt images taken by a smartphone camera. Some examples of the receipts can be seen in figure 3.1. The majority of receipts are from Sweden. However, there are a small number of outliers that are not. All images have a corresponding list of ground truth labels connected with it which includes: vendor name, date, address, total price, tax rate, currency, and a list of products with its corresponding name, price, and amount. However, not all images contains all labels. From the total number of receipts 90 of them are selected at random and put in the test set.

Figure 3.1: Examples of receipts from the dataset

3.2 Data pre-processing

The focus of this research lies on the receipt text rather than on the images. The following steps describe the process of getting the text from the images. This process is necessary, but it is not part of the research. At step 1 in figure 1.1 all images were binarized using the adaptive thresholding method proposed by Sauvola and Pietikäinen [26]. Image Binarization is a process of transferring a gray-scaled image to a binary form. A binary image is an image where each pixel is either black or white, hence the name binary. The thresholding window size used is 25 pixels. The result after this step can be seen in figure 3.3. The binarizad images are then sent to the OCR engine provided in the Apples Vision framework. The version of the framework is 1.0 and the detection languages are set to Swedish and English in that priority. The result from the OCR engine is a list of text boxes with its coordinates relative to the receipt image and the text inside. This is represented by step 2 in figure 1.1. In step 3 the raw receipt text is filtered from the raw OCR result with a set of empirically predetermined rules. All text boxes which match one of the following rules are removed:

- The length of the textbox is ≤ 0

- The length of the text is ≤ 0 or ≥ 40
- Text contains 4 or more of the same characters in a row.
- The text vowel ratio is < 0.1 or > 0.9

After removing the unwanted OCR result all text boxes are grouped by if they appear on the same line and then all the lines are sorted from the top to the bottom. So the final result after step 3 is the line by line text of the receipt. Figure 3.2 shows the example of the text sorted by lines.

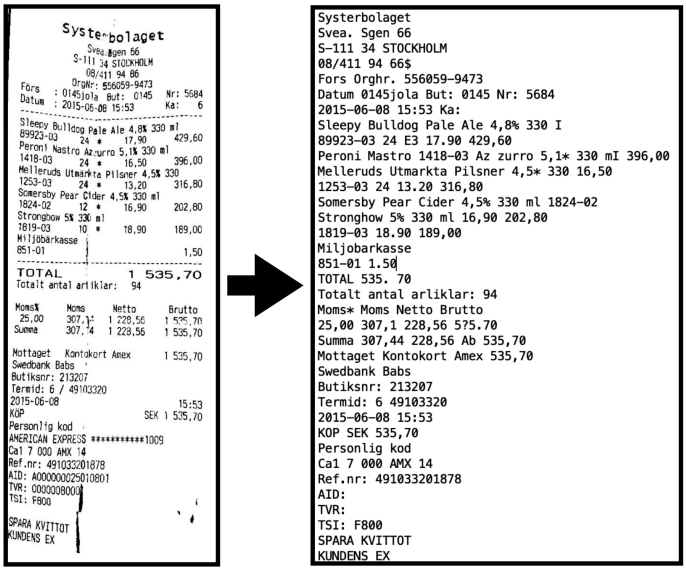


Figure 3.2: From binarized receipt image to sorted text.

Figure 3.3: Receipt images before (left) and after (right) the adaptive thresholding has been applied

3.3 Rule-based

The simple rule-based method for data extraction is created to have a baseline to compare to. This is accomplished by having a set of rules for each specific data field. Each of these rules is applied to the result from step 3 in figure 1.1. All of the rule sets are implemented using the behavior trees where each data class is represented by a single tree.

Vendor name

The text form the textbox at the top is taken unless it contains the Swedish word for receipt 'kvitto' or the text length is less than 2, then the next one is taken instead.

Date

A set of most common date formats is kept and then the receipt text is searched for these formats using regex.

Address

Similar to the date a set of most common address formats is kept and then the receipt text is searched for these formats using regex.

Currency

The set of most common currency acronyms and abbreviations is kept and then the receipt text is searched using regex.

Tax rate

Firstly a receipt text is searched for numbers with percent sign $\%$ at the end. Then, for each result of the first, another search on the same line or the line above is performed looking for keywords like 'moms' or 'vat'. If these keywords are found then the number is considered to represent a tax rate. If no match is found the same process is then performed for all numbers, not only ones with percent sign at the end.

Total price

The total price is extracted by 3 different rules performed in priority order. Firstly the receipt text is searched for a number coming directly after a currency sign. If no match found then the text is searched for a numbers coming after keywords 'total', 'belopp', 'summa', 'betala', 'tot', 'kontokort', 'amount' and 'net'. If no match is found then the text is searched for all the numbers in decimal format and the largest of them is considered to be the total price.

Product list

Each product is represented by three different classes; product name, product price, and product amount. The product's list is extracted by matching the consecutive lines of text with the format for price, amount, and name.

3.4 NER data creator

The result after step 3 is a text string for each of the receipts. In addition to that, a JSON file containing the extracted classes is also available. However, all of the used models require a label for each of the inputting tokens (words or characters) and that had to be created separately. This creation is represented by step 4 in figure 1.1. Figure 3.2 shows an example of text after the step 3. For this particular example, we know that a vendor name is 'Systembolaget' and that the address is 'Sveavägen 66 S-111 34 STOCKHOLM'. However, neither the vendor name of the address string is present in the text due to the reading error of the OCR engine. In order to mitigate this the Levenshtein distance instead or regular string comparison is used when looking for strings in receipt text. The Levenshtein distance is a metric for measuring the distance between two sequences. In can also be used on strings and it represents a minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word into the other [27]. Levenshtein distance by itself is not sufficient for the problem since some words in the text can be perceived as two by the OCR engine as it can be seen in figure 3.2 where 'Sveavägen' became 'Svea. Sgen'. Instead of comparing each word in the text with the ground truth class, the list of all possible sub-strings of various lengths is created from the receipt text. Given that the length of the current ground truth class (example 'Systembolaget') is l then the list will include all sub-strings of lengths $l-2, l-1, l, l+1$ and $l+2$. Each of these words is in the list will then be compared with the ground truth class and the best match is found. If the best match Levenshtein similarity is greater then 75% then this sub-string is marked with the current class label. The threshold of 75% is produced by an empirical study. All of the text words that are not marked with some of the ground truth labels are marked with label 'O'. Algorithm 1 shows how the labels are created in more detail.

Algorithm 1 Labeling NER data

for class c in receipts ground truth **do**
    $l = len(c)$
    $bestMatch = None$
    for i in range(-2, -1, 0, 1, 2) **do**
        $substrings = allSubstingsOfLength(l + i)$ ▷ From receipt text
        for substring in substrings **do**
            $LevenshteinDistance = Levenshtein(c, substring)$
            if LevenshteinDistance is less than best **then**
                $bestMatch = substring$
    if LevenshteinSimilarity(c, bestMatch) > 75 **then**
        Mark substring as label c

3.4.1 Synthetic data

At this point, each receipt consists of a text where each word has its corresponding label. However, by inspecting the data some imbalances inside the classes are detected. Table 3.1 shows the number of times the 5 most frequent vendors and addresses occur. The 5 most common vendors are represented in almost 1/3 of all receipts. A similar trend is seen with addresses where the top 5 most frequent addresses are found in about 1/5 of all receipts. In order to increase the variation inside the classes and get more data a two synthetic datasets are created, **synthetic1000** and **synthetic10000**. Synthetic data is created by taking one of the real receipts texts at random and replacing its classes by following rules:

Vendor name

The vendor name is replaced by a random one from a set of vendor names created from all real receipts.

Address

The address is replaced by a new one created from a random combination of a street name, street number, postal code, and city, all of which are contained in a small database of Swedish addresses.

Date

The date is replaced by a new one with a random date format taken from a small database of date formats. All synthetic generated dates are between the first of January 2010 and the first of January 2020.

Total price

The total price is replaced by a random one from a normal distribution with a mean of 100 and a standard deviation of 30.

Product name

Each product name is replaced with a random one from a set of already existing product names.

Currency, tax rate, product price, amount and the text that does not belongs to any known class is not replaced. Both datasets are created on the same way with **synthetic1000** having 1000 synthetic receipts in addition to the real ones while **synthetic10000** has 10000.

Address	**Number of Occurrences**
Stockholm - Kungsgatan 41	52
Drottninggatan 53 111 21	21
Barnhusgatan 1 11123 Stockholm	18
Sveavägen 66 S-111 34 Stockholm	13
Gamla Brogatan 27 111 20 Stockholm	13
Total	117
Vendor	**Number of Occurrences**
Kjell & Company	63
Clas Ohlson	56
Systembolaget	47
Webhallen	46
Hemköp	24
Total	236

Table 3.1: Most common addresses and vendor names from the dataset

3.5 Oracle model

As seen in the previous chapter, errors in the OCR result can produce difficulties in creating labeled NER data. Algorithm 1 mitigates a part of them, but it is only a heuristic without any guarantees that will always work. In order to measure the impact of OCR error and end errors that may be caused by wrongly labeling the data an oracle model is introduced. Oracle is a model that knows all of the right labels in NER data and uses that to produce the final result. In a case of perfect OCR and perfect labeling algorithm the accuracy of this model would be 1. However, since the OCR engine is not perfect the result from the oracle model can be used to see what proportion or final result error is caused by OCR and data labeling algorithm and to set the upper limit of the other three networks.

3.6 LSTM model

For this model, a 2-layered bi-directional LSTM followed by a linear layer is used for a character-wise classification. At step 4c in figure 1.1 the NER data is split to characters and one-hot encoded using the vocabulary consisted of all ASCIIs characters, digits, and punctuation. All characters are capitalized prior to one-hot encoding meaning that the network is case insensitive. By an empirical study, a number of hidden nodes are set to 256 with the Adam optimizer and the loss function used is Cross-Entropy Loss. The 20% of train data is used for a validation set and the network is trained for a 2000 epoch with the batch size 16. The model that achieved the least loss on the validation set is saved as a final model. This is represented as a step 5c in figure 1.1. At this point, the result is a list of all characters for a receipt with their respective labels. At step 6c this is converted to the final result. With an exception for class 'O', all consecutive characters that belong to the same class are concatenated to form the final data extraction result.

3.7 BERT model

For this model, a pre-trained BERT-Base, uncased model with 12-layers, 768 hidden nodes, 12 heads, and 110M parameters is used. NER data is tokenized using WordPiece, and a vocabulary used is constructed from all of the words from the train set. BERTs pre-trained model is fine-tuned to classify each word token into one of the 10 classes. The 20% of the train set is used as a validation

set and the model is trained for 30 epoch with batch size 16. The model with the best loss on the validation set is saved as a final model. This is represented by step 6a in figure 1.1. To form the final result each consecutive word that belongs to the same class, with the exception for class 'O', is concatenated. In case that the several groups of the same class are created for a single receipt then the final result is decided by a majority vote. If all of the groups are different then the one that comes first is taken.

3.8 GCN model

To create data for the GCN model a graph had to be constructed from NER data. This is represented by step 5b in figure 1.1. The graph is constructed in the same way as proposed by Lohani et al. [24]. Figure 3.4 shows an example of how one such graph looks. Each word represents a node in the graph and the data extraction problem is tackled as a node classification problem. Each node (word) is being classified as one of the 10 classes. The links are created between the nearest words in four main directions, up, down, left, and right. For neighbors in directions left and right only those words that are on the same line are considered, while for the directions up and down only the words in previous and next lines are considered. If two ore more words are directly beneath a third one, then the one with the biggest vertical overlap is selected. This way it is made sure that a word has a maximum of one link in a single direction. An example of this can be seen in figure 3.4 where the word 'Rosenlundsgatan' is above both '11853' and 'Stockholm' but since its overlap with 'Stockholm' is bigger they are the only ones connected.

Feature Creation

Each node (word) is represented by 312 dimensional feature vector. The features are divided into three categories boolean, numeric and textual.
Boolean features outputs a 8 dimensional vector and are calculated as follows:

(i) **isDate**: a parser that looks if a word can be a date.

(ii) **isKnownCity**: looks if word matches some known city in a small database.

(iii) **isKnownCountry**: looks if word matches some known country in a small database.

(iv) **isAlphaNumeric**, **isAlphabetic**, **isNumeric**, **isNumberwithDecimal** and **isCurrency** looks for specific word format.

Numeric features outputs a 4 dimensional numeric vector with relative distances to 4 neighbours in the graph. If a node does not have a neighbour in some direction a value is set to a 1.
Textual features outputs a 300 dimensional vector as a result of Byte Pair Encoding. The implementation of BPE used is BPEmb, recommended by Heinzerling and Strube [28]. The version used is a BPEmp multi-lingual model with vocabulary size of 1000000.
The final 312 dimensional vector is constructed by concatenating the output of these three calculated vectors. These calculation are implemented, only with small adaptations, as suggested by Lohani et al. [24].
The model itself is a 5 layer GCN architecture. The receipt graph is passed through 4 hidden layers, each followed by a ReLu activation function. The last layer uses the softmax to classify each node into one of the classes. Figure 3.5 shows the model architecture. With each layer, the number of filters nf is increased by a factor of 2, and the initial number of filters is 16. The model is trained for 1000 epochs with the initial learning rate of 0.001 and the L2 regularization factor $5e^{-4}$. 20% of training data is used as a validation set and the model that achieves the best validation loss is saved. The model is trained only on real data since the graph constructing algorithm needs the information about the width, length and relative position of each text box adding, removing, or altering text was not possible.

3.9 Comparison method

In the sections 3.6, 3.7 and 3.8 the process of extracting key data fields with different models were explained. All of the extracted fields are of type String. The simple method of comparing these extracted fields with ground truth using string equality can lead to wrong results. The two price strings '99.99' and '99,99' are not the same even if they obviously represent the same price. The reason why this occurs is that the decimal number on the receipt is represented with both '.' and ',' while in ground truth the '.' is always used. The same applies to two date strings '17-01-95' and '17-01-1995'. The string is not equal, even if the date they represent is. Some receipts have multiple different date formats for the same date while ground truth always has only one. Due to problems mentioned above the simple equality can not always be used. The total price is always converted to a number before comparison. Date strings are always parsed and checked if they represent the same date no matter if the

strings differ. Tax rates come both with and without the % sign so only the numerical parts of the tax rate strings are compared. However, for the vendor, currency, and address the simple string comparison is used. For a product to be counted as correctly extracted all three parts: name, price, and the amount have to match.

3.10 Experimental setup

All experiments are implemented using Python, Google Colab, and the PyTorch open-source machine learning library. All training is done using the Tesla K80 GPU with 12GB GDDR5 VRAM and 2496 CUDA cores.

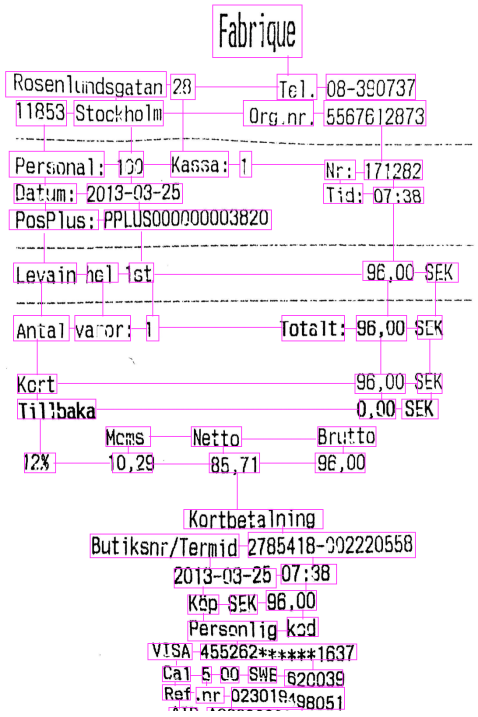


Figure 3.4: A graph constructed with words from receipt

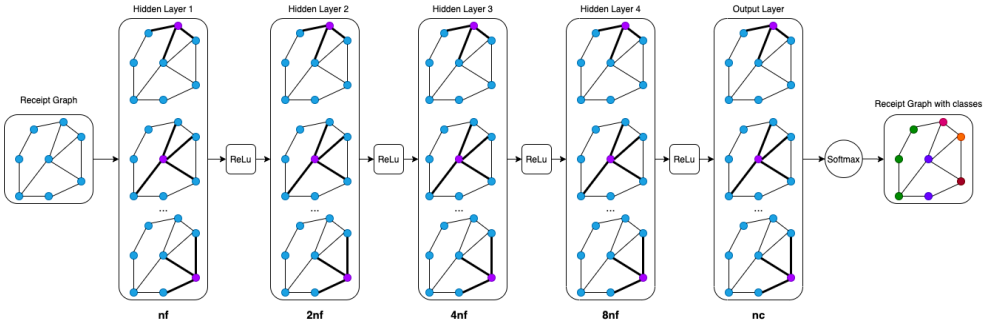


Figure 3.5: A model of GCN architecture used in this thesis.

Chapter 4

Results

In this section, results from the rule based, oracle, and the three different machine learning models are presented. At the beginning of this section the results for each model in the form of F1 scores are presented. After that, an overview of results for each separate data field is given. The chapter concludes with a comparison between all the models.

4.1 Oracle

Table 4.1 shows the result of the oracle model. Even though the model classifies each token correctly the final result is not 1. This depends on two things. Firstly, the OCR engine is not perfect which results in the errors in text scanning and secondly, the algorithm for creating the NER data is only a heuristic and some data can be lost in transition. Date, price, tax rate, and currency have an F1 score above, or near 0.9. The vendor has an F1 score of 0.791, while the scores for address and products are significantly lower. The table 4.2 gives a bit of the perspective for these three data fields. Instead of comparing the strings as they are, a small tolerance in terms of Levenshtein distance is allowed. The F1 score for the vendor increased from 0.791 to 0.985 when the tolerance in Levenshtein distance is increased to 3. Under the same conditions, the F1 score of address increased from 0.574 to 0.869. For the products, the tolerance is introduced only for the product name and it resulted into the increase from 0.404 to 0.515. The micro average for the oracle model is 0.660 in the F1 score and it gives an estimate of how big impact the OCR and the errors in NER data creation have on the final results. From the 90 receipts in the test set, only 9 of them had all the data fields correctly extracted. If the product list is omitted, then this number rises to 40. This model also gives a

good baseline for comparison with the other models since it shows how good the model that predicts all tokes correctly is.

Class	Precision	Recall	F1
VENDOR	0.791	0.791	0.791
DATE	0.916	0.916	0.916
ADDRESS	0.593	0.556	0.574
TAX RATE	0.985	0.985	0.985
PRICE	0.900	0.900	0.900
CURRENCY	0.895	0.895	0.895
PRODUCTS	0.413	0.395	0.404
MICRO AVG	0.668	0.652	0.660
MACRO AVG	0.785	0.777	0.781

Table 4.1: The final result of the oracle model.

Class	Levenshtein Distance	Precision	Recall	F1
VENDOR	0	0.791	0.791	0.791
	1	0.884	0.884	0.884
	2	0.916	0.916	0.916
	3	0.985	0.985	0.985
ADDRESS	0	0.593	0.556	0.574
	1	0.695	0.651	0.672
	2	0.864	0.810	0.836
	3	0.899	0.841	0.869
PRODUCTS	0	0.413	0.395	0.404
	1	0.511	0.487	0.499
	2	0.521	0.497	0.508
	3	0.526	0.503	0.515

Table 4.2: The development of the precision, recall and F1 score for the oracle model when the Levenshtein distance tolerance is introduced.

4.2 Rule Based

Table 4.3 shows the result of the rule-based method. The best results are achieved on date and currency where the F1 scores are 0.923 respective 0.926. The F1 score for the price field is 0.833 and for the tax rate, that score is

0.627. Vendor and address both have an F1 score slightly under 0.5. The worst achieved performance is for the products which have an F1 score of 0.127. The overall performance of the rule base method is 0.515 in micro average and 0.710 in the macro average. From the 90 receipts in the test set, only 4 of them have all the data fields extracted correctly, and that number rises to 27 if the product list is omitted. Table 4.4 shows however how the scores change if the tolerance for correct answers is increased for vendor, address and products fields. The F1 score for the vendor is increased from 0.455 to 0.693 when a Levenshtein distance tolerance is 3 and under the same conditions, the score for address is increased from 0.427 to 0.565. The F1 score for products is almost doubled when tolerance for product name is increased with the increase from 0.127 to 0.228. However, the product price and amount still have to be a perfect match for a product to be considered as correct.

Class	**Precision**	**Recall**	**F1**
VENDOR	0.444	0.465	0.455
DATE	0.907	0.940	0.923
ADDRESS	0.412	0.444	0.427
TAX RATE	0.600	0.831	0.697
PRICE	0.833	0.833	0.833
CURRENCY	0.980	0.877	0.926
PRODUCTS	0.234	0.087	0.127
MICRO AVG	0.591	0.456	0.515
MACRO AVG	0.715	0.705	0.710

Table 4.3: The final result of the rule based model.

Class	Levenshtein Distance	Precision	Recall	F1
VENDOR	0	0.444	0.465	0.455
	1	0.533	0.558	0.545
	2	0.611	0.640	0.625
	3	0.678	0.709	0.693
ADDRESS	0	0.412	0.444	0.427
	1	0.471	0.508	0.489
	2	0.515	0.555	0.535
	3	0.544	0.587	0.565
PRODUCTS	0	0.234	0.087	0.127
	1	0.282	0.105	0.154
	2	0.387	0.145	0.211
	3	0.419	0.156	0.228

Table 4.4: The development of the precision, recall and F1 score for the rule based model when the Levenshtein distance tolerance is introduced.

4.3 LSTM

Figure 4.1 shows the training and validation loss of the LSTM model on three different datasets. Validation loss on real data reaches its minimum around epoch 750 and then it gets bigger and bigger as the model overfits for the training data. A similar trend is seen with both synthetic1000 and synthetic10000 where their loss achieves its minimum around epoch 850 respective 1400. Even if the validation loss is more stable with these two datasets, overfitting is still an issue. However, all three datasets achieve very similar minimum loss, little over 0.6 with synthetic1000 being the lowest. Table 4.5 shows the result of LSTM model on these three datasets. The result matches the loss plot showing that all three models have quite similar micro average scores with synthetic1000 being the best with the F1 score of 0.278. All three models show promising results on the date, tax rate, price, and currency. On the other hand, the result for vendor, address, and product all have an F1 score lower than 0.1. None of the models have managed to correctly extract all data fields from a single receipt.

Table 4.6 shows how the scores change for the model trained on synthetic1000 when the Levenshtein tolerance is introduced. The F1 score for the vendor increased linearly from 0.078 to 0.170 and the same score for products increased from 0.037 to 0.088. The change for the address was not that high increasing

from 0.060 to 0.075 on the first step and then staying unchanged.

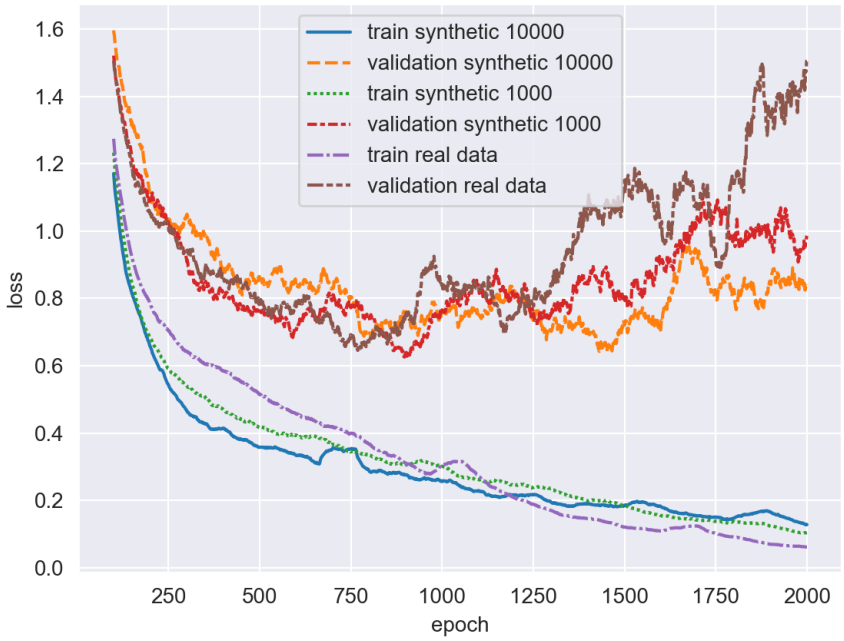


Figure 4.1: Train and validation loss of the LSTM model on three different datasets.

Class/Data	**Real Data**	**Synthetic1000**	**Synthetic10000**
VENDOR	Precision: 0.013 Recall: 0.012 F1: 0.012	Precision: 0.090 Recall: 0.070 F1: **0.078**	Precision:0.090 Recall: 0.058 F1: 0.071
DATE	Precision: 0.743 Recall: 0.699 F1: 0.720	Precision: 0.810 Recall: 0.820 F1: 0.814	Precision: 0.812 Recall: 0.831 F1: **0.821**
ADDRESS	Precision: 0.057 Recall: 0.063 F1: 0.060	Precision: 0.056 Recall: 0.063 F1: 0.060	Precision: 0.055 Recall: 0.063 F1: 0.059
TAX RATE	Precision: 0.686 Recall: 0.738 F1: 0.711	Precision: 0.750 Recall: 0.877 F1: **0.801**	Precision: 0.064 Recall: 0.800 F1: 0.712
PRICE	Precision: 0.602 Recall: 0.489 F1: 0.540	Precision: 0.691 Recall: 0.622 F1: 0.655	Precision: 0.741 Recall: 0.700 F1: **0.720**
CURRENCY	Precision: 0.959 Recall: 0.825 F1: **0.887**	Precision: 0.891 Recall: 0.860 F1:0.875	Precision: 0.877 Recall: 0.877 F1: 0.877
PRODUCTS	Precision: 0.011 Recall: 0.018 F1: 0.014	Precision: 0.028 Recall: 0.054 F1: **0.037**	Precision:0.024 Recall: 0.049 F1: 0.032
MICRO AVG	Precision: 0.215 Recall: 0.269 F1: 0.239	Precision:0.239 Recall: 0.333 F1: **0.278**	Precision: 0.232 Recall: 0.333 F1: 0.274
MACRO AVG	Precision: 0.470 Recall: 0.445 F1: 0.457	Precision: 0.510 Recall: 0.531 F1: **0.520**	Precision: 0.496 Recall: 0.530 F1: 0.512

Table 4.5: The final result of the LSTM model.

Class	Levenshtein Distance	Precision	Recall	F1
VENDOR	0	0.090	0.070	0.078
	1	0.119	0.093	0.105
	2	0.164	0.128	0.144
	3	0.194	0.151	0.170
ADDRESS	0	0.056	0.063	0.060
	1	0.070	0.079	0.075
	2	0.070	0.079	0.075
	3	0.070	0.079	0.075
PRODUCTS	0	0.028	0.054	0.037
	1	0.037	0.072	0.049
	2	0.052	0.102	0.069
	3	0.066	0.130	0.088

Table 4.6: The development of the precision, recall and F1 score for the LSTM model when the Levenshtein distance tolerance is introduced

4.4 BERT

Figure 4.2 shows the train and validation loss for the three datasets. Real data reaches its minimum loss of 0.4 on the validation set around epoch 6 and then loss only gets bigger. Synthetic1000 reaches its minimum loss of 0.2 around epoch 11 and stays stable around that value for the rest of the training. Validation loss for the synthetic10000 dataset reaches its minimum around epoch 5 and stays stable after that being only slightly higher than the training loss. Table 4.7 shows that the synthetic10000 achieves good results on the test set as well having the micro average F1 score of 0.961. This score represents the result of the BERT model on the token classification task. Table 4.8 shows the final result for the BERT model after the data has been extracted. It shows that the synthetic10000 has the best final result as well, with the micro average score of 0.445. Synthetic10000 gave the best F1 scores for all fields except for an address where it shares the best F1 of 0.344 with the real data model. The address is the data field, with the exception of products that has the lowest F1 score. The best scores are achieved on the tax rate, date, price, and currency with F1 scores of 0.885 respective 0.842, 0.818, and 0.885. The data field with the lowest score is the products with an F1 score of 0.027. Even though BERT was able to achieve high results of classifying the product name, price and amount tokens, the final F1 score after the data are extracted is very low. Synthetic1000 has been able to extract all data fields correctly from only 1 receipt.

If the product list is omitted from the data fields then that number rises to 23. Table 4.9 shows how the scores are changed for the synthetic10000 model is the Levenshtein distance tolerance is added when comparing the results. The F1 score for the vendor data field has increased from 0.667 to 0.845 when the distance tolerance is 3. In that same condition, the F1 score for the address has increased from 0.356 to 0.661. The scores for products almost doubled, but are still very low with the best F1 score of 0.44. As explained in section 3.9 for a product to be counted as correctly extracted, all three components price, name and amount have to match. If only product names are compared then the F1 score for products increases to 0.502.

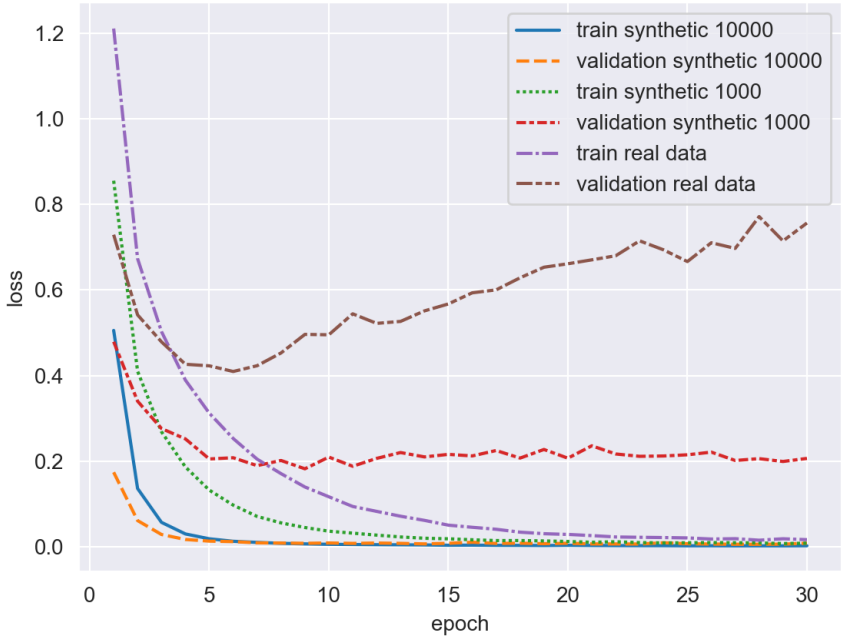


Figure 4.2: Train and validation loss of the BERT model on three different datasets.

Class	Precision	Recall	F1	Support
VENDOR	0.901	0.863	0.882	95
DATE	0.951	0.942	0.946	103
ADDRESS	0.705	0.729	0.717	59
TAX RATE	1	1	1	225
PRICE	0.996	1	0.998	261
CURRENCY	1	1	1	133
PRODUCT NAME	0.953	0.953	0.953	317
PRODUCT PRICE	0.989	0.992	0.990	261
PRODUCT AMOUNT	0.944	1	0.714	17
MICRO AVG	0.958	0.965	0.961	1471
MACRO AVG	0.965	0.965	0.965	1471

Table 4.7: Token classification result of BERT model.

Class/Data	Real data	Synthetic1000	Synthetic10000
VENDOR	Precision: 0.500 Recall: 0.430 F1: 0.462	Precision: 0.530 Recall: 0.512 F1: 0.521	Precision: 0.683 Recall: 0.651 F1: **0.667**
DATE	Precision: 0.746 Recall: 0.602 F1: 0.667	Precision: 0.838 Recall: 0.687 F1: 0.755	Precision: 0.928 Recall: 0.771 F1: **0.842**
ADDRESS	Precision: 0.356 Recall: 0.333 F1: **0.344**	Precision: 0.321 Recall: 0.286 F1: 0.303	Precision: 0.356 Recall: 0.333 F1: **0.344**
TAX RATE	Precision: 0.984 Recall: 0.954 F1: 0.969	Precision: 0.899 Recall: 0.954 F1: 0.925	Precision: 0.985 Recall: 0.985 F1: **0.985**
PRICE	Precision: 0.765 Recall: 0.433 F1: 0.553	Precision: 0.792 Recall: 0.678 F1: 0.731	Precision: 0.837 Recall: 0.800 F1: **0.818**
CURRENCY	Precision: 0.923 Recall: 0.842 F1: 0.881	Precision: 0.904 Recall: 0.825 F1: 0.862	Precision: 0.893 Recall: 0.877 F1: **0.885**
PRODUCTS	Precision: 0.019 Recall: 0.021 F1: 0.020	Precision: 0.028 Recall: 0.025 F1: 0.026	Precision: 0.028 Recall: 0.027 F1: **0.027**
MICRO AVG	Precision: 0.361 Recall: 0.335 F1: 0.348	Precision: 0.393 Recall: 0.384 F1: 0.388	Precision: 0.458 Recall: 0.433 F1: **0.445**
MACRO AVG	Precision: 0.610 Recall: 0.508 F1: 0.554	Precision: 0.614 Recall: 0.565 F1: 0.589	Precision: 0.673 Recall: 0.635 F1: **0.653**

Table 4.8: The final result of the BERT model.

Class	Levenshtein Distance	Precision	Recall	F1
VENDOR	0	0.683	0.651	0.667
	1	0.756	0.721	0.738
	2	0.841	0.801	0.821
	3	0.866	0.826	0.845
ADDRESS	0	0.356	0.333	0.344
	1	0.475	0.444	0.459
	2	0.627	0.587	0.607
	3	0.661	0.619	0.637
PRODUCTS	0	0.027	0.028	0.027
	1	0.035	0.033	0.034
	2	0.044	0.042	0.043
	3	0.044	0.042	0.043

Table 4.9: The development of the precision, recall and F1 score for the BERT model when the Levenshtein distance tolerance is introduced.

4.5 GCN

Figure 4.3 shows the plot of train and validation loss on real data. Minimum loss on the validation set is reached around epoch 100 and it stays stable after that. Table 4.10 shows the final result of the GCN model. The best F1 score is achieved on currency 0.682 and the worst on price 0.022. The model has high precision on the date, tax rate, price, and currency, but it has a lot of true negatives which reflects on recall score. The micro average F1 score of the model is 0.167 and the macro average is 0.305. Table 4.11 shows how the scores of the vendor, address, and product change when the Levenshtein distance tolerance is introduced. The F1 score for the vendor name is increased from 0.222 to 0.245 and the F1 score for products is increased from 0.067 to 0.089 while the scores for address stayed unchanged.

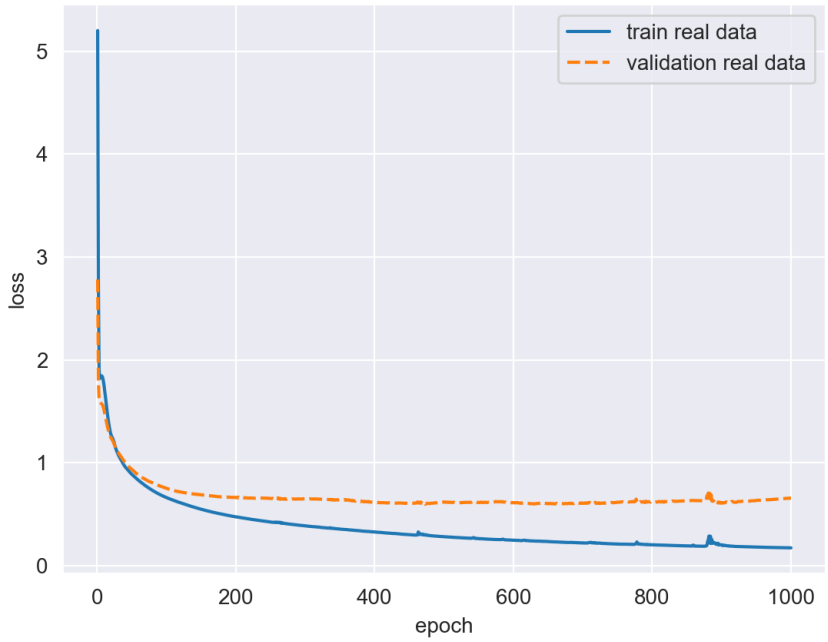


Figure 4.3: Train and validation loss of GCN model on real data.

Class	Precision	Recall	F1
VENDOR	0.350	0.163	0.222
DATE	0.833	0.181	0.297
ADDRESS	0.083	0.063	0.072
TAX RATE	0.783	0.277	0.410
PRICE	1.0	0.011	0.022
CURRENCY	0.968	0.523	0.682
PRODUCTS	0.071	0.063	0.067
MICRO AVG	0.226	0.133	0.167
MACRO AVG	0.616	0.202	0.305

Table 4.10: The final result of the GCN model.

Class	Levenshtein Distance	Precision	Recall	F1
VENDOR	0	0.350	0.163	0.222
	1	0.375	0.174	0.238
	2	0.400	0.180	0.254
	3	0.400	0.180	0.245
ADDRESS	0	0.083	0.063	0.072
	1	0.083	0.063	0.072
	2	0.083	0.063	0.072
	3	0.083	0.063	0.072
PRODUCTS	0	0.071	0.063	0.067
	1	0.071	0.063	0.067
	2	0.091	0.081	0.086
	3	0.095	0.084	0.089

Table 4.11: The development of the precision, recall and F1 score for the GCN model when the Levenshtein distance tolerance is introduced.

4.6 Result comparison

Tables A.3, A.4, A.5, A.6, A.7 and A.8 shows some example of the extracted fields by different models.

Table 4.12 shows the comparison of all models with their best results. The overall best result is 0.515 in micro average and 0.710 in the macro average both achieved by the rule-based model. The rule-based model achieved the best F1 score in all data fields except vendor and tax rate where the best result is attained by the BERT.

4.6.1 Vendor

The best F1 score on the vendor name data field is attained by the BERT model. The F1 score achieved represents 86% of the oracle model F1 score which is the highest score this model could have achieved. It was superior over all other models.

4.6.2 Date

Rule-based achieved the best F1 score for the date. However, both LSTM and BERT have comparable results. The best machine learning model and the

second-best overall, BERT had reached 92% of the oracle model F1 score. The GCN model showed to be inferior to the other models on this task.

4.6.3 Address

The best F1 score of address is obtained by the rule-based model. The second-best result not so far behind is achieved by the BERT model. The BERT model managed, however, to reach only 58% of the Oracle score. Both LSTM and GCN have significantly lower scores.

4.6.4 Tax rate

The tax rate is the field with the single best F1 score across all data fields and models. BERT has scored 100% of the oracle model. The second-best result here was achieved by LSTM. The rule-based model and GCN have not achieved comparable results.

4.6.5 Price

The rule-based model was best at extracting the price as well. The BERT model was second best with a score that represents the 91% of the oracle model. The GCN model was inferior with a very low score.

4.6.6 Currency

The best F1 score on currency is achieved by the rule-based model. LSTM and BERT have very similar scores which represent around 98% of the oracle model. The least successful model was GCN.

4.6.7 Products

The product list is a field with the lowest score for all models except GCN. The best F1 score is achieved by rule-based. The second best is GCN, however, that score is very low. Both LSTM and BERT have scored less than 10% of the oracle model with LSTM being slightly better.

Class/Model	Rule Based	LSTM	BERT	GCN	Oracle
VENDOR	0.455	0.078	**0.677**	0.222	0.791
DATE	**0.923**	0.814	0.842	0.297	0.916
ADDRESS	**0.427**	0.075	0.344	0.072	0.574
TAX RATE	0.697	0.801	**0.985**	0.410	0.985
PRICE	**0.833**	0.655	0.818	0.022	0.900
CURRENCY	**0.926**	0.875	0.885	0.628	0.895
PRODUCTS	**0.127**	0.037	0.027	0.067	0.404
MICRO AVG	**0.515**	0.278	0.455	0.167	0.660
MACRO AVG	**0.710**	0.520	0.653	0.305	0.781

Table 4.12: Comparison of the F1 scores of final results for all models.

Chapter 5

Discussion

In this chapter, the analysis and the discussion about the results is presented.

5.1 Problems

One of the main problems that affect all models including the rule-based one is the errors produced by the OCR engine. Since all of the final extraction results are exactly compared to the ground truth, errors in the OCR can make this comparison appear false even if the extraction is done correctly. Levenshtein distance tolerance is included in the result analysis in order to see how much of the error depends on this. This problem is even greater for the machine learning models as the OCR error propagates through all of the steps 2-6 in the figure 1.1. Erroneously read words can lead to even bigger errors in the data creation algorithm 1 in steps 4a, 4b, and 4c. That can potentially lead that some tokens are falsely labeled during this process which leads to information loss. This is confirmed by the results of the oracle model. The micro average of the oracle model is only 0.660 even though this models correctly predict all of the tokens. Adding the Levenshtein distance tolerance increases its F1 score for vendor name up to 0.985. Doing the same for the address the F1 score is increased up to 0.896. Neither of these numbers is 1 despite the Levenshtein distance tolerance, which means that additional information is being lost during the token creation and labeling.

An additional problem is an ambiguity in the receipt data fields. Figure A.9 shows one such example where both 180.00 and 179.90 could be interpreted as the total price. The ground truth of the data goes in favor of the larger one and the model that predicts 179.90 would be considered wrong.

5.2 Machine learning models

Vendor name

The best performing model on the vendor name extractions was BERT. It attained the F1 score of 0.677, which is much higher than the LSTMs 0.078 and GCN's 0.222. The complexity of the BERT and the fact that it had a pre-trained knowledge about the language were making a difference. Table 4.9 shows that the BERTs F1 score is increased by 0.178 when Levensthein tolerance is 3 which means that a fair amount of the error is caused due to OCR. Under the same conditions, the LSTMs F1 score more than doubled and reached 0.170. The score is still very low meaning that the potential OCR error is not the only issue and that LSTM is not able to learn to classify the vendor name. The same applies to the GCN, where the F1 score for vendor increased only slightly meaning that the error is more systematic.

Date and Currency

BERT had the highest F1 score for both date and currency. However, the difference is not that high and the BERT only slightly outperforms the LSTM. These two fields can be extracted with high accuracy even with the simple character-wise LSTM. Having the pre-trained knowledge of the language with BERT or the knowledge of the underlying receipt structure with GCN is not that important for date or currency. How big proportion of error is caused by OCR is not clear in this case since it is not reasonable to allow Levenshtein distance for date or currency due to fact that even a small difference in the text can make a huge difference in value.

Address

The address is the data field that together with the product list gave the worst results which can partly be attributed to the nature of the data field itself. Table A.1 shows the differences in length and the number of words for the different data fields. The average length of the address in the test dataset is around 31 and the average number of words is 4.86. This means that the LSTM has to classify 31 characters correctly on average compared to 3 for currency or 9.65 for data. The margin of error is much greater both for the OCR engine to make a mistake while reading the word and for the model to misclassify some part of the address. This can also be seen in the oracle models score which had scored only 0.574 in F1 despite knowing all the right classes. The same

applies to BERT and GCN which both have to correctly classify 4.86 words on average. If one of these words is missed, then the result is classified as incorrect even if the rest of the words are correctly extracted.

Tax rate and Price

The tax rate is the field where the BERT model showed a great advantage over the others managing to score 0.985 in the F1 score or 100% of the oracle models score. The LSTM achieved the F1 score of 0.801 while the GCN has a score of 0.410. Even though the tax rate has a simple format and usually comes with the percentage it is not that trivial to recognize it and it is often confused with discount. Neither LSTM nor GCN have any previous knowledge about the language and were unable to make this distinction. A similar reason applies to the total price as well where the BERT had the best F1 score of 0.818 followed by LSTM with 0.655 and GCN with 0.022. The format is simple, but it is often mistaken with the other prices.

Products

The product list gave the worst results across all models. The main reason behind this is that a single product is actually consisting of 3 different fields, product name, product price, and product amount. All of these three fields have to match in order to count a single product correctly extracted. The average number of words in a product name is 2.51 and the average length is 15.52 characters. This together with product price and product amount makes a margin or error much bigger compared to other data fields. As explained in section 5.1 even a small error in OCR gets bigger as it propagates through steps 3, 4, 5, and 6 in 1.1. Oracle model achieves only 0.404 in the F1 score. Even when 3 steps in Levenshtein distance for the name are allowed the F1 score reaches only 0.515. Another challenge for products is the final data extraction at step 6 from the result given by the models. At this step product name, product price, and product amount that belong to the same product must be grouped together. Is some of these are missed then the whole product list might be corrupted. This can be seen in table A.7. The prices for products given by LSTM are all shifted by one and thus all incorrect. The best performing model here is GCN with the F1 score of 0.067. The best performing model overall BERT performed worst here with the score 0.027.

BERT was the model that gave the best score among the machine learning methods. It showed that having the pre-trained knowledge about the language

is more important than knowing the underlying geometrical position like GCN does. The experiment with different amounts of synthetic data showed that the model is very good at generalizing when more data is provided. Figure 4.2 shows that BERT tends to overfit when the dataset is small. Increasing the size of the data reduced the validation loss by a huge margin. The synthetic10000 dataset was able to achieve the validation loss nearly as low as training loss. Table 4.7 shows the true potential of the BERT model. The micro average F1 score of BERT on the token classification task is 0.9611, much greater than the final F1 micro average of 0.455. This shows that a lot of error is introduced by the OCR and the labeling algorithm, rather than the BERT model itself.

GCN model performed much worse than in the work by Lohani et al. [24]. There are two main reasons for this. Firstly, they used a dataset of invoices which tend to be much more structured and less diverse than the receipts. Secondly, they have used 28 different classes compared to only 10 used in this thesis. This means that the much greater percentage of the nodes is marked with class '0' compared to work by Lohani et al. [24]. Table A.2 shows the probability of a randomly chosen node with a given label to be connected to nodes with other labels. It shows that a graph is very sparse in the sense that most of the nodes are labeled with label '0'. The probability for a date node to be connected to something else then '0' is less than 10% which means that GCN is not able to utilize its advantage of knowing the neighbor nodes, but rather has to focus on the node value. Similar low probabilities are noticed with the tax rate, currency, and total price. The only data fields where the probability of a connection with the label '0' is low are product name, price, and amount. There exist more connections between these three fields so GCN is able to utilize its propagation rule much better. The product list is the only field where GCN has managed to make an advantage over the other two models.

LSTM model proved to be much better on the numeric data fields comparing to the textual ones which is quite reasonable considering the complexity of the model. The experiment with the different synthetic datasets showed that the size of the data does not have as big impacts as on BERT. The model that performed the best is the one trained on synthetic1000, but the difference is not that significant meaning that the problem is not overfitting but rather that the underlying function is too complex to be learned by this model.

5.3 Rule Based

The results showed that the simple rule-based model outperforms all three of the machine learning methods. The rule-based method achieved the best result on all of the data fields except vendor name and tax rate. The reason behind this is that most of the extracted field have somehow strict format. The best example of it is the currency. Keeping the simple list of the known currencies and their abbreviations the rule-based method was able to reach the F1 score of 0.926. The machine learning methods were not too far behind on this task with the BERT models having the F1 score of 0.885 just slightly better than the LSTM with a score of 0.875. However, since the list of currencies is of fairly limited size, the rule-based method seems like a logical approach. A similar approach can be used on both date and address. Keeping the list of date and address formats and then matching it to the receipt text gave the F1 scores of 0.923 respective 0.427. Both scores top all of the other methods. When it comes to the total price, taking the largest number that matches the price format yielded into F1 score of 0.833. This score is only slightly better than the BERTs 0.818. Matching the price format is a straight forward task, but taking the largest such number is not always the right solution. Image A.2 shows that a receipt can sometimes have two prices, both before and after the discount.

The rule-based method was not able to beat BERT when it comes to the vendor name. This data field does not follow any specific pattern and it comes in various sizes and places in a receipt. The best score of the rule base model is achieved when taking the first text box that does not match the pattern for any other data field. This works in only roughly 50% of the time. BERT was, on the other hand, able to learn to infer this information from the surrounding text.

Another field where BERT proved to be better is the tax rate. This field comes most often together with the percent sign. However, the rule-based method was not able to differentiate between the tax rate and other data like discount which also can contain the percent sign. BERT, on the other hand, was able to do so almost perfectly having the F1 score of 0.985.

5.4 Societal and Ethical Aspects

As mention in chapter 1, the work in this thesis can be used to automatize data extraction from receipts or similar documents with unstructured text. In

the ideal world, this would mean that the human resources that are now spent on this could be used elsewhere. However, some companies might decide that human resources are not needed anymore and it could lead to a loss of jobs. This work can also be extended to create a tool for helping a person with tracking its economy by, for example taking a photo of each receipt and saving the extracted information in digital form. This would mean that the information would become easily accessible by the individual and could have a positive impact on its economy. However, if this would mean using a third-party app then there is a huge potential risk for exploitation. By taking a photo of receipts one is giving out all the information about its shopping habits. In addition to that, based on the information that can be extracted from receipts one could possibly infer other habits, as well as the home and work address and possibly even the income level. All this information is highly valued today.

5.5 Sustainability

Work by Sarenmalm [29] investigates the environmental impact of the usage of paper receipts in Sweden. It showed that the receipts are often coated with chemicals hazardous to humans and the environment and that they are non-recyclable due to their high contamination risk and that around 60000 trees are used annually to create 1.5 billion paper receipts that are issued in Sweden. The same work also assess the possible substitute for paper receipts in the form of a digital one. As mentioned in the previous section, the work in this thesis can be extended for usage on an individual level. If such a thing would become popular then it can potentially slow down or prevent transformation toward digital receipts which would leave a big environmental footprint.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, three different machine learning algorithms were compared in the task of information extraction from a receipt text. The values that were extracted are vendor name, date, address, total price, tax rate, currency, and the product list. The experiment showed that the best performing machine learning model is BERT achieving 0.455 in micro average F1 score, followed by BiLSTM with the micro average of 0.278. The model with the least micro average was GCN with a score of 0.167. The BERT model had the best F1 score among the machine learning models on all individual fields except the product list where the GCN was better. It was able to outperform the ruled based model on vendor name and tax rate extraction. However, the simple rule-based model was better on all other date fields and had a better micro average F1 score of 0.515 as well. Several reasons for this were identified. One of the main reasons is the OCR error that propagates over the rest of the steps and causes imperfections in data labeling heuristic. Another reason is that most of the data fields in receipts have a well-defined format and the simple rules were sufficient to extract them. Although the BERT model did not manage to beat the rule-based model it showed great potential and it would be interesting to look at in the future work.

6.2 Future work and improvements

There are several aspects of the work that would be interesting to investigate further. Since it is showed that the majority of the error in the BERT model depends on the OCR error and the labeling algorithm the logical step would

be to try to mitigate this. One approach to minimize the OCR is to create a language model from the receipts. OCR used in this thesis uses a language model to infer and correct characters where the image is not perfect and the probability of different characters is similar. However, the language model used in the receipt might be different than one used in written language otherwise. Another approach would be to try to use a completely different OCR. When it comes to mitigating the errors caused by labeling algorithm the best solution would be to manually label all of the text, or to use some other more advanced methods then Levenshtein distance.
It would also be interesting to use some of the other BERT models like BERT-LARGE or compare Cased to Uncased variants of BERT [30]. When it comes to GCN model one improvement would be to include more classes, rather 10 that are used in this thesis to see if the graph with more known classes would improve the performance. Another interesting experiment would be to see if the rule-based method could be combined with BERT to boost the overall accuracy. This hybrid method would try to utilize the advantages of both models, by using the rules to search for some key formats, but use BERT where no such match is found.
Since the rule-based model showed to be the best for the task it would be interesting to see if some of its strengths could be used to increase the performance of the machine learning models. The majority of data points from receipts come in well-defined formats. This could be utilized by the machine learning models as well by adding some constraints that incorporate domain knowledge. One possibility would be to add a constraint optimization layer on the top of the neural network. This layer would then make sure that the constraints are satisfied. For example, one such constraint would be that address token never comes as a single word, but rather as a group of neighboring words. One could also utilize the fact that product name, price, and amount are also closely connected. Another similar approach to incorporate rules into the machine learning models is to use constraint violation penalty. This can be accomplished by adding an extra loss as a penalty if some of the constraints are not satisfied.

Bibliography

[1] Abigail J Sellen and Richard HR Harper. *The myth of the paperless office*. MIT press, 2003.

[2] Bertin Klein, Stevan Agne, and Andreas Dengel. Results of a study on invoice-reading systems in germany. In *International workshop on document analysis systems*, pages 451–462. Springer, 2004.

[3] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[4] Archana Goyal, Vishal Gupta, and Manish Kumar. Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, 29:21–43, 2018.

[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Understanding lstm networks – colah's blog. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. (Accessed on 05/04/2020).

[8] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, pages 207–212, 2016.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[11] Understanding encoder-decoder sequence to sequence model. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af (Accessed on 05/04/2020).

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[16] F1 score - wikipedia. https://en.wikipedia.org/wiki/F1_score. (Accessed on 04/08/2020).

[17] A. Suponenkovs, A. Sisojevs, G. Mosāns, J. Kampars, K. Pinka, J. Grabis, A. Locmelis, and R. Taranovs. Application of image recognition and machine learning technologies for payment data processing review and challenges. In *2017 5th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pages 1–6, Nov 2017. doi: 10.1109/AIEEE.2017.8270536.

[18] Rizlene Raoui-Outach, Cecile Million-Rousseau, Alexandre Benoit, and Patrick Lambert. Deep learning for automatic sale receipt understanding. *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Nov 2017. doi: 10.1109/ipta.2017.8310088. URL http://dx.doi.org/10.1109/IPTA.2017.8310088.

[19] V. P. d'Andecy, E. Hartmann, and M. Rusiñol. Field extraction by hybrid incremental and a-priori structural templates. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 251–256, April 2018. doi: 10.1109/DAS.2018.29.

[20] M. Rusiñol, T. Benkhelfallah, and V. P. dAndecy. Field extraction from administrative documents by incremental structural templates. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1100–1104, Aug 2013. doi: 10.1109/ICDAR.2013.223.

[21] D. Schuster, K. Muthmann, D. Esser, A. Schill, M. Berger, C. Weidling, K. Aliyev, and A. Hofmeier. Intellix – end-user trained information extraction for document archiving. In *2013 12th International Conference on Document Analysis and Recognition*, pages 101–105, Aug 2013. doi: 10.1109/ICDAR.2013.28.

[22] H. Hamza, Y. Belaid, and A. Belaid. A case-based reasoning approach for invoice structure extraction. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 327–331, Sep. 2007. doi: 10.1109/ICDAR.2007.4378726.

[23] Rasmus Berg Palm, Ole Winther, and Florian Laws. Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 406–413. IEEE, 2017.

[24] D. Lohani, A. Belaïd, and Y. Belaïd. An invoice reading system using a graph convolutional network. In Gustavo Carneiro and Shaodi You, editors, *Computer Vision – ACCV 2018 Workshops*, pages 144–158, Cham, 2019. Springer International Publishing. ISBN 978-3-030-21074-8.

[25] Rinon Gal, Nimrod Morag, and Roy Shilkrot. Visual-linguistic methods for receipt field recognition. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 542–557, Cham, 2019. Springer International Publishing. ISBN 978-3-030-20890-5.

[26] J Sauvola and M Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000. ISSN 0031-3203.

[27] Levenshtein distance - wikipedia. `https://en.wikipedia.org/wiki/Levenshtein_distance`. (Accessed on 04/08/2020).

[28] Benjamin Heinzerling and Michael Strube. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. *arXiv preprint arXiv:1710.02187*, 2017.

[29] Isabel Sarenmalm. Would you like your receipt?: Sustainability perspectives of consumer paper receipts, 2016.

[30] Google Research. Bert. `https://github.com/google-research/bert`, 2018.

Appendices

Appendix A

In this appendix, the sample of data and data extraction results from different models is presented.

Figure A.1: Example receipts after image binarization.

Figure A.2: Example receipt from original image to sorted text.

	RULE BASED	LSTM	BERT	GCN
Vendor	Class Ohlson	.. 121 0 36	Class Ohlson	None
Date	2013-04-17	None	None	2013-04-17
Address	None	None	None	Class Ohlson
Tax rate	25 %	25 %	25 %	25 %
Price	315	None	315	None
Currency	None	None	None	None
Product	None	**Name**: KABELKANAL 38X16M 5 @ **Price**: 315 **Amount**: 1	**Name**: KABELKANAL 38X16M **Price**: 315 **Amount**: 5	**Name**: kabelkanal **Price**: 029579 **Amount**: 1

Figure A.3: Data extraction results for bottom left receipt in figure A.1.

	RULE BASED	LSTM	BERT	GCN
Vendor	Non Solo Bar	N S B R	Non Solo Bar	None
Date	2017-01-21	2017-01-21	2017-01-21	2017-01-21
Address	Rorstrandsgatan 113 40 Stockholm	32 22 41 IO 3011968 B K N 33830 D	Rorstrandsgatan 113 40 Stockholm	None
Tax rate	12 %	12 %	12 %	None
Price	120	120	120	None
Currency	SEK	SEK	SEK	None
Product 1	**Name**: farskpressad juice **Price**: 49 **Amount**: 1	**Name**: 4 S 2117 14:52 S1 S F J **Price**: 49 **Amount**: 1	**Name**: farskpressad juice **Price**: 0 **Amount**: 1	**Name**: farskpressad **Price**: 49 **Amount**: 1
Product 2	**Name**: cappuccino liten **Price**: 33 **Amount**: 1	**Name**: C L **Price**: 33 **Amount**: 1	**Name**: cappucino liten **Price**: 0 **Amount**: 1	**Name**: cappuccino **Price**: 33 **Amount**: 1
Product 3	**Name**: latte stor **Price**: 38 **Amount**: 1	**Name**: L S **Price**: 38 **Amount**: 1	**Name**: latte stor **Price**: 0 **Amount**: 1	**Name**: latte stor **Price**: 38 **Amount**: 1

Figure A.4: Data extraction results for top right receipt in figure A.1.

	RULE BASED	LSTM	BERT	GCN
Vendor	Ramen	R G B	Ai Ramen	Ramen Gamla
Date	2020-02-18	2020-02-18	2020-02-18	2015-06-09
Address	Gamla Brogatan 27 111 20 Stockholm	1 L 1.00 S	Gamla Brogatan 27 11120	20 Stockholm
Tax rate	12 %	12 %	12 %	None
Price	155	155	155	None
Currency	SEK	SEK	SEK	SEK
Product	None	**Name**: M **Price**: 0 **Amount**: 1	**Name**: spicy miso **Price**: 155 **Amount**: 1	None

Figure A.5: Data extraction results for bottom right receipt in figure A.1.

	RULE BASED	LSTM	BERT	GCN
Vendor	RESTAURANT	RESTAURANT TETE DE SOLAISE B.P	Restaurant tete de solaise	None
Date	21-02-2018	21-02-2018	None	None
Address	None	31 73150 VAL D'ISERE FRANCE 06	73150 VAL D'ISERE FRANCE	None
Tax rate	12 %	None	None	None
Price	154.2	154.2	154.2	None
Currency	None	None	None	None
Products	Figure A7	Figure A7	Figure A7	Figure A7

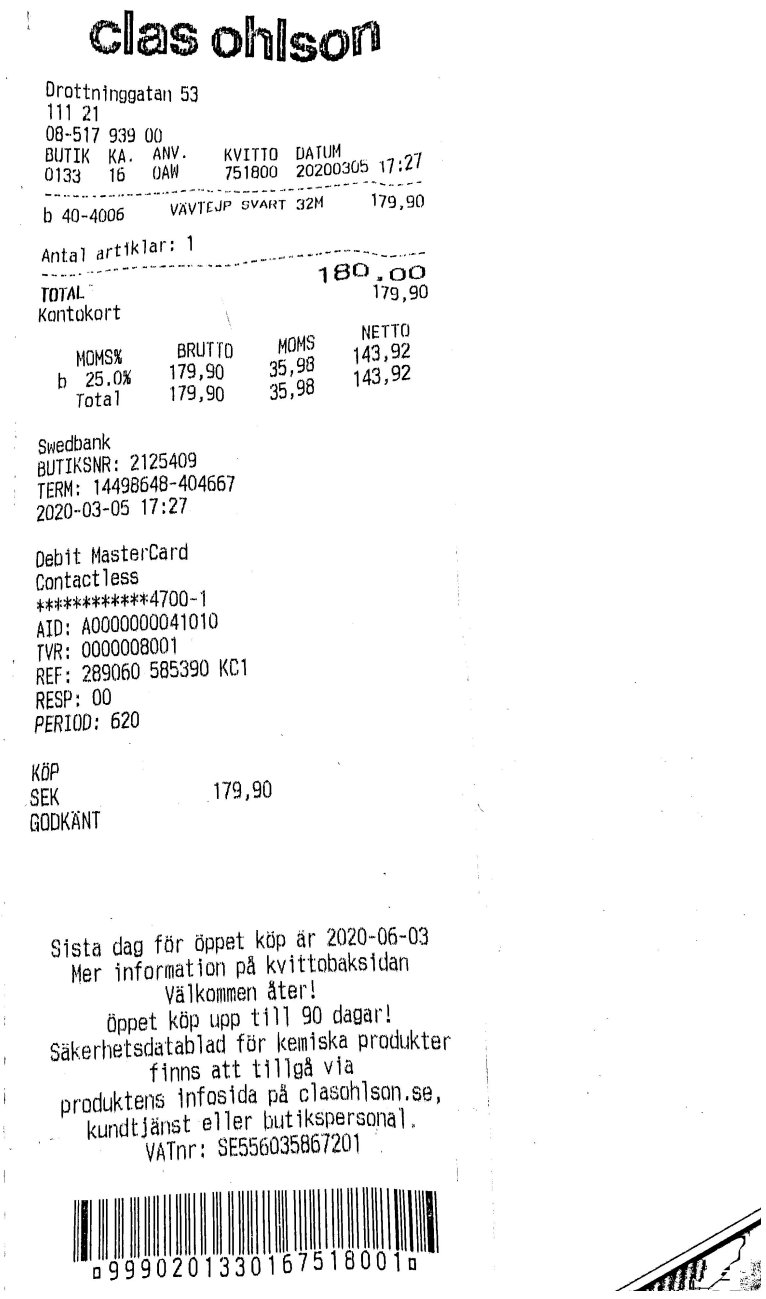
Figure A.6: Data extraction results for top left receipt in figure A.1.

	RULE BASED	LSTM	BERT	GCN
Products	None	**Name**: JULIEN PATES AU HOMARD H OERS POL SONE **Price**: 33 **Amount**: 1	**Name**: pates au homard **Price**: 00 **Amount**: 1	**Name**: b **Price**: 1 **Amount**: 1
	None	None	None	**Name**: oers **Price**: 8 **Amount**: 1
	None	**Name**: lasagne **Price**: 8 **Amount**: 1	**Name**: lasagne **Price**: 00 **Amount**: 1	**Name**: lasagne **Price**: 21 **Amount**: 1
	None	**Name**: pate bolognaise **Price**: 21 **Amount**: 1	**Name**: pate bolognaise **Price**: 50 **Amount**: 1	**Name**: pate bolognaise **Price**: 1 **Amount**: 1
	None	**Name**: burger solarise **Price**: 19.50 **Amount**: 1	**Name**: burger solarise **Price**: 50 **Amount**: 1	**Name**: burger solarise **Price**: 22.50 **Amount**: 1
	None	**Name**: coc **Price**: 22.5 **Amount**: 1	**Name**: coca **Price**: 50 **Amount**: 1	**Name**: coca **Price**: 4.50 **Amount**: 1
	None	**Name**: pates carbonara **Price**: 4.50 **Amount**: 1	**Name**: pates carbonara **Price**: 50 **Amount**: 1	**Name**: pates carbonara **Price**: 19.50 **Amount**: 1
	Name: choco viennois **Price**: 15.60 **Amount**: 1	**Name**: choco viennois **Price**: 19.50 **Amount**: 1	**Name**: choco viennois **Price**: 60 **Amount**: 1	**Name**: choco viennois 5.20 **Price**: 15.60 **Amount**: 1
	Name: expresso forte **Price**: 5.50 **Amount**: 1	**Name**: expresso **Price**: 15.60 **Amount**: 1	**Name**: expresso forte **Price**: 50 **Amount**: 1	**Name**: expresso forte **Price**: 5.50 **Amount**: 1
	None	**Name**:80 5.50 C 1 EARL GREY **Price**: 5 **Amount**: 1	**Name**: earl grey **Price**: 00 **Amount**: 1	**Name**: earl grey **Price**: 5 **Amount**: 1

Figure A.7: Products list results for top left receipt in figure A.1.

	RULE BASED	LSTM	BERT	GCN
Vendor	Tommy Hilfiger Man Sth	None	Tommy Hilfiger Man Sth	None
Date	2019-07-23	2019-07-23	2019-07-23	None
Address	Hamngatan 18-20 11177 Stockholm	HAMNGATAN 18-20 S-111 77 STOCKHOLM	Hamngatan 18 - 20 11177 Stockholm	None
Tax rate	0 %	None	None	None
Price	700	None	700	None
Currency	SEK	SEK	SEK	SEK
Product	**Name**: kortbetalning **Price**: 700	**Name**: hilfiger logo zip mock **Price**: 400	**Name**: hilfiger logo zip mock **Amount**: 1	None

Figure A.8: Data extraction results for receipt in figure A.2.



clas ohlson

Drottninggatan 53
111 21
08-517 939 00

BUTIK	KA.	ANV.	KVITTO	DATUM
0133	16	OAW	751800	20200305 17:27

b 40-4006 VÄVTEJP SVART 32M 179,90

Antal artiklar: 1

TOTAL 180,00
Kontokort 179,90

MOMS%	BRUTTO	MOMS	NETTO
b 25.0%	179,90	35,98	143,92
Total	179,90	35,98	143,92

Swedbank
BUTIKSNR: 2125409
TERM: 14498648-404667
2020-03-05 17:27

Debit MasterCard
Contactless
**********4700-1
AID: A0000000041010
TVR: 0000008001
REF: 289060 585390 KC1
RESP: 00
PERIOD: 620

KÖP
SEK 179,90
GODKÄNT

Sista dag för öppet köp är 2020-06-03
Mer information på kvittobaksidan
Välkommen åter!
Öppet köp upp till 90 dagar!
Säkerhetsdatablad för kemiska produkter
finns att tillgå via
produktens infosida på clasohlson.se,
kundtjänst eller butikspersonal.
VATnr: SE556035867201

¤99902013301675180011

Figure A.9: An example of the receipt from the dataset with an ambiguous total price.

Class	Average length	Average words number
VENDOR	12.16	1.92
DATE	9.65	1.07
ADDRESS	31.05	4.86
TAX RATE	3.23	1
PRICE	4.33	1
CURRENCY	3	1
PRODUCT NAME	15.42	2.51

Table A.1: Average length and average word number for different ground truth classes in the test dataset.

Class	V.	D.	A.	T.R	P.	C.	P.N	P.P	P.A.	O.
V.	0.4	0.001	0.2	0	0	0	0	0	0	0.4
D.	0.004	0.04	0.004	0	0	0	0.04	0	0	0.91
A.	0.09	0.02	0.57	0	0	0	0.004	0	0	0.33
T.R.	0	0	0	0	0	0	0.01	0	0	0.99
P.	0	0	0	0	0	0.06	0.06	0	0	0.88
C.	0	0	0	0	0.06	0	0.03	0	0	0.91
P. N.	0	0.005	0.007	0.0003	0.003	0.002	0.49	0.24	0.15	0.1
P. P.	0	0	0	0	0	0	0.3	0.3	0.3	0.1
P.A.	0	0	0	0	0	0	0.3	0.3	0.3	0.1
O.	0.007	0.01	0.01	0.006	0.007	0.006	0.05	0.005	0.005	0.87

Table A.2: Probability for a node with a given label to be connected to another node with another label. (V. - Vendor, D.- Date, A.- Address, T.R- Tax Rate, P. - Price, C. - Currency, P.N. - Product Name, P.P. Product Price, P.A. Product Amount, O.- Other).

TRITA EECS-EX-2020:247