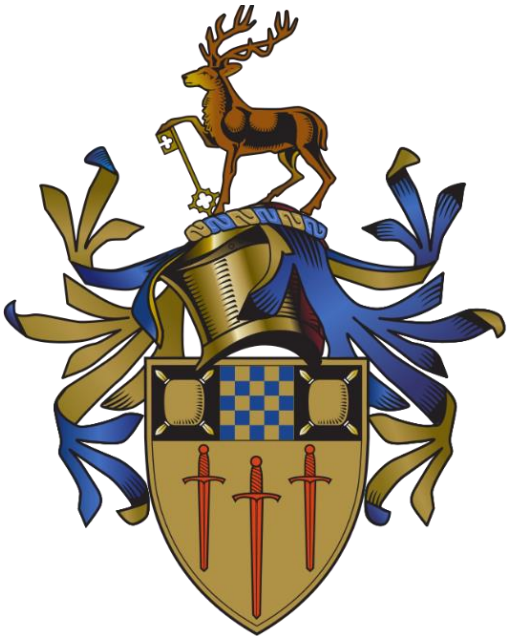
Automatic Product Extraction, Classification, and Analysis of Receipt Data

University of Surrey

Faculty of Engineering & Physical Sciences

Department of Computer Science



2021-2022

Grace Yingtong Lin

Supervisor: Professor Helen Treharne

URN: 6461871

Declaration of Originality

"I confirm that the submitted work is my own work. No element has been previously submitted for assessment, or where it has, it has been correctly referenced. I have clearly identified and fully acknowledged all material that is entitled to be attributed to others (whether published or unpublished) using the referencing system set out in the programme handbook.

I agree that the University may submit my work to means of checking this, such as the plagiarism detection service Turnitin® UK and the Turnitin® Authorship Investigate service. I confirm that I understand that assessed work that has been shown to have been plagiarised will be penalised."

Abstract

Optical Character Recognition is a field of research that has seen many advancements, such that the current state of OCR for printed text is sometimes referred to as a "solved" problem. However, there are still many areas of improvement that exist, as reflected in the current research space, and specifically in the field of receipt and invoice processing. Whilst there are significant innovations in the OCR space for invoice and receipt processing, post-OCR data analysis and receipt parsing research is beginning to receive more attention, with research focused on integrating emergent or newer technologies such as those involving transformer models. The two tasks that form the basis of receipt understanding, whilst invariably linked, have typically been treated as separate problems. Key information extraction research is rooted in the field of Natural Language Processing and is closely related to text classification problems. This project seeks to combine research from both fields (OCR and NLP) into a method to extract key information from scanned receipts and invoices and provide insight and analysis into these documents. This work will focus on data from personal sales receipts, rather than in a business context, which is more typical of similar works in this space. Therefore, this work aims to extract an itemised purchase list from a receipt and classify those items into relevant categories. By comparing the latest methods in relevant machine learning research, this project proposes a proof-of-concept for an automated pipeline to extract, classify, and analyse product data from receipts and invoices.

Acknowledgements

I would like to express my gratitude to a few people who have supported me throughout the completion of this project and throughout my time at the University of Surrey. To begin with, I extend a huge thank you to my project supervisor, Helen Treharne, for her guidance and advice throughout the past few months. Her support has significantly helped me in the completion of this project.

Additionally, I would like to thank my friends for their continued support and encouragement throughout the duration of my degree.

Last but not least, I thank my parents, for their sacrifices, commitment, and unconditional love, which without, I would not be completing my degree as a first-generation University student.

Table of Contents

Table of Figures

Table of Tables

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BoW	Bag of words
CNN	Convolutional Neural Networks
CORD	Consolidated Receipt Dataset for Post-OCR Parsing
cuDNN	NVIDIA CUDA Deep Neural Network library
DCNN	Deep Convolutional Neural Network
FinCap Survey	Financial Capability Survey
HBS	Household Budget Survey
HITL	Human-in-the-Loop
ICDAR	International Conference on Document Analysis and Recognition
KIE	Key Information Extraction
LSTM	Long-Short Term Memory
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
OCR	Optical Character Recognition
RNN	Recurrent Neural Network
SROIE	Scanned Receipts OCR and key Information Extraction dataset
TF-IDF	Term frequency-inverse document frequency

1 Introduction

This project will explore text classification methods, OCR (Optical Character Recognition) technology, and key information extraction (KIE) techniques to provide insights into an individual's purchase receipts for easier management and tracking of personal expenses. This will be achieved through creating a proof-of-concept pipeline for a service (either a web or mobile application) in which a user can scan their physical receipts with their smartphone and generate spending breakdowns.

By analysing and experimenting with existing products on the market that allow users to scan receipts, current OCR techniques, data extraction techniques, and text classification techniques, we combine the results to provide a pipeline for financial insights on physical receipts.

1.1 Project Aims

This project aims to develop a way to extract relevant information from physical sales receipts and classify this data for personal financial insights, which will speed up and ease the process of personal budgeting and accounting.

There are two distinct tasks within this project: the extraction of key receipt data from an image of a receipt, and the classification of this data for the creation of basic spending insights. Key data in the context of this project mainly refers to, but is not limited to, the purchased products list on the receipt.

For the first task, provided an image of a receipt or invoice, this project aims to provide a method to extract an itemised list of the products purchased, along with other relevant information such as store name, date, and time of the purchases.

Then, once a list of items is retrieved, the second part of this project aims to use relevant product data to classify the purchases into relevant categories (such as snacks, hygiene, and drinks). This can then be used to provide spending insights such as how much was spent on a given category during a specific timeframe.

Combining the two tasks will complete the overall project goal of providing an end-to-end pipeline for the extraction, classification, and analysis of receipt data into a user-friendly and accessible form so that users can easily access financial overviews without having to manually review receipts.

1.2 Project Motivations

With the move to online banking, financial insights and tracking have become significantly easier to access. However, these solutions, whilst helpful, are not fully comprehensive. The biggest category of spending that digital banks cannot track is cash spending. Items bought with cash are neglected in financial breakdowns as these transactions cannot be tracked by an online banking app due to their physical nature (at most it would show up as an ATM withdrawal).

Additionally, in modern supermarkets, one may be able to purchase electronic goods or clothes, but because the transaction was completed in a supermarket under one receipt or purchase, the amount would be classified under groceries. For an individual tracking their spending, this approach may not be of enough detail to assist with budgeting and would then require manually tracking purchases.

Traditional budgeting involves sorting through potentially dozens of receipts and invoices, and manual data insertion into a spreadsheet or physical budgeting book, to understand where money has been spent. This process is time-consuming and tedious; automation of this would make budgeting easier and more manageable. With most people owning smartphones in the UK, this project proposes the idea of a pipeline that takes a photo of a receipt from a mobile phone, analyses the data in the receipt, and automatically categorises the products purchased on the receipt, in order to provide spending insights to the user.

Whilst invoice and receipt analysis programs and APIs are available for business uses, such as [1], [2], and [3] the technology is not widely available for personal use. In the mentioned APIs and services, the focus of the insights provided are tailored towards business applications, such as processing invoices and streamlining expense claims, and not relevant for an individual's use.

1.3 Project Objectives

The objectives of the project are to:

1. Identify appropriate receipt OCR solutions and methods
2. Identify appropriate methods of machine learning for text classification
3. Design and implement a method for extracting key information from a picture of a receipt
4. Design and implement a text classification system for the text retrieved from invoices and receipts
5. Evaluate the OCR method implemented to extract text from a photo
6. Evaluate the NLP method implemented to classify product items into groups

1.4 Success Criteria

The following criteria will be used to determine the success of the project during the evaluation stage after development and testing:

- A demonstrable working OCR method of extracting text from a picture of a receipt taken on a mobile phone
- A demonstrable working model that can extract an itemised list of purchases from OCR output of a receipt
- A demonstrable working text classification model that can categorise product names with over 80% accuracy

1.5 Report Structure

This project is split into 8 chapters and the outline of the report is as follows

Chapter name	Brief description
1 Introduction	An introduction to the project's aims, objectives, and motivations
2 Literature review	This chapter documents the research conducted for the project into current methodologies and the theory behind them and provides background to the project's motivations and aims
3 Problem Analysis and Design	The problem analysis and design chapter of this report discusses the findings from the Literature Review and justifies the experiments that will be performed during implementation
4 OCR Design, Implementation, & Testing	This chapter of the report describes the process of experimenting with the OCR component of the project, the implementation of the OCR model and results from testing
5 KIE Design, Implementation, & Testing	Chapter 4 describes the design and implementation of the KIE component of the project pipeline, and the results from experimentation
6 Text Classification Design, Implementation, & Testing	This chapter details the experiments conducted, how the text classification system of the project was implemented, and the experimentation and results of exploring text classification techniques
7 Statement of Ethics	The Statement of Ethics discusses the legal, social, ethical, and professional issues that were considered during the project and the effect of these issues, if any, on the progress of the project
8 Project Evaluation	The Project Evaluation reviews the overall project trajectory and examines the success of the project in meeting its goals and objectives

2 Literature Review

This chapter of the report will focus on analysing the domain of the project by reviewing current solutions, academic papers, technical frameworks, and the necessity of this project within the domain. This section also involves research and discussion of the theory behind relevant components of the project, such as convoluted neural networks and other machine learning algorithms. Research from this section will be used to influence decision-making and planning during the problem analysis and design stage of the project.

2.1 UK Finances

The following section examines and discusses the UK's current financial landscape to provide background into the real-life applications of a receipt OCR pipeline. Additionally, this research will serve to provide context to the project's motivations and why a receipt OCR pipeline might be beneficial in the UK.

2.1.1 Financial Literacy

The 2018 Financial Capability Survey (FinCap Survey) is a survey conducted by the UK's Money and Pensions Service (also known as the Money Advice Service) [4]. The survey collected data from almost 6,000 adults across a representative sample of UK residents, and is used to support the Financial Capability Strategy for the UK [5]. it was found that 63% of adults in the UK do not feel like they are in control of their finances and only 49% could last more than 3 months on savings if they lost their main source of income. 21% of the UK population rarely or never save, and 22% have less than £100 in savings. This indicates that a large proportion of the UK cannot (or choose not to) manage their savings.

Saving is more difficult for low-income households, who are more likely to be unable to save. However, this is not the only reason behind struggling to save money; the research found that a person's tendency to save is also linked, in part, to the time that they are willing to make for their finances. Traditional budgeting is a time-consuming process and may be a roadblock for those who would like to save but do not have time for it.

2.1.2 Project Financial Survey

In a survey designed to research the market need for a personal budgeting app, over 60 people responded. Respondents were in the age range 18-34 years old, and 67% of respondents were students. A full list of questions and responses are provided in Appendix A.

Respondents were asked questions about digital banking, their perceived benefits, and mobile budgeting apps. Many of the respondents when prompted about what they like about digital banking answered variations of the answer "easy to use".

When asked whether they had ever been stressed or anxious about personal finances, 73% answered yes. Of those that said yes, only 13% said they didn't believe a budgeting app could help lessen their worries. The majority of respondents (73%) said they already budget their spending and over half of those said they manually budget. Of those who said they do not budget, only a small minority believe that budgeting seems

pointless or that they don't need to budget. This indicates that many of the respondents want to budget, but either find it too stressful or not easy enough.

Not many people had used a budgeting app before and there were mixed feelings about the helpfulness of a budgeting app, with the average helpfulness score out of ten being 5.64. However, out of all respondents, 81% of people were interested in using a mobile application to manage their finances regardless, with the top features requested being spending insights, ability to sync multiple accounts, customisable spending categories, and physical receipt scanning. The main reservations that respondents had about a mobile budgeting app revolved around privacy, ease of use, and being able to sync existing accounts.

From this research, it is apparent that many worry about finances, and would appreciate an easier way to budget, but equally many are wary about using a mobile app – either due to security concerns or previous bad experiences. These findings will be considered when presenting the project outcomes.

2.2 Receipt OCR Overall Landscape

This section of the literature review will look at existing solutions to the presented problem, including any packaged solutions that are on the market as well as any research works that encompass the overall domain of digital receipt scanning, processing, and analysis.

2.2.1 Existing APIs

In the current market, there exists multiple APIs that have a focus on receipt or invoice data extraction. For example, Veryfi and Nanonets both offer receipt OCR in the form of APIs for the purpose of invoice processing. However, these APIs are largely focused on business users and business invoice processing. As mentioned on their website [1], Veryfi's API offers an out-of-the-box solution for businesses and includes extraction of information such as tax categorisation and vendor identification. Such features are unnecessary for this project's use case, which is focused on personal finances and everyday receipts. In terms of the technology behind the services, Nanonets states that they "use AI and machine learning (ML) to auto-extract meaningful data fields" on their website [3]. Their API also allows users to validate data and uses this input to further train the AI behind their model with the customer's custom data. Conversely, Veryfi bypasses human validation completely, and states on their website that they use a model trained on 30 million plus receipts; achieving a 3% error rate and eliminates the need for human verification [6].

Other APIs that offer similar functionalities include TAGGUN [2], klippa [7], and Lucidtech [8], amongst many others. They all offer similar services in that they provide businesses with the option to streamline expense processing - from scanning receipts to extracting the data into a manageable and exportable form. However, none of these APIs are marketed for personal use, nor made available as an app. Additionally, they do not provide statistics on uploaded receipts.

2.2.2 Existing mobile apps

On the Google Playstore, when searching "receipt scanner", an array of different applications appear. The top app with the most downloads is an app named *"QuickBooks Self-Employed"* [9] with over 5 million downloads. The app allows users to take pictures of receipts and log the data extracted as expenses, though the receipt scanner is not the main feature of the app. This app and the next 3 top apps are targeted towards business users, and whilst they allow users to scan and upload receipts and extract data from these receipts, they do not offer insights or any budgeting features.

As mentioned in [10], there are also a few apps that exist on Apple's app store for iOS that offer similar functionalities, as well as an open-source receipt scanner called *"ReceiptManager"* [11] which is available as a mobile app and web app. All the mentioned apps do not provide insights, but mainly store the extracted data from photos of receipts (either on the mobile phone, or a separate database).

Separately, upon searching "budgeting" or "expenses" in the Google Playstore, the most downloaded app is an app named *"Money Manager Expense & Budget"* by *"Realbyte Inc."* [12] with over 10 million downloads as

of February 2022. In their description on the App Store and their website [13] they state the app offers budget management, with different ways to view statistics generated by manually entered data.

Similarly, the second most downloaded app *"Monefy"* [14] offers similar functionalities and the main difference between the top two apps are in its user interface and design.

Across the budgeting apps mentioned and others in the same category, there is no one application that combines the ability to upload and scan receipts and generate insights and track expenses from those receipts for personal usage and budgeting. There are, however, some features that are prevalent across all the apps mentioned thus far such as security features. The similarities, functionalities, and differences are summarised in the comparison table, Table 1, below.

App	OCR receipts	Spending insights	Personal use	Mobile app	Security features
QuickBooks [9]	✓	✗	✗	✓	✓
ReceiptManager [11]	✓	✗	✓	✓	✓
Money Manager [13]	✗	✓	✓	✓	✓
Monefy [14]	✗	✓	✓	✓	✓

Table I - A comparison of existing mobile apps

2.2.3 Receipt classification and OCR research

In the academic space of OCR and text classification specifically for receipts, a handful of research papers on different approaches to the problem are available. Receipt OCR as a topic has ties to both OCR technologies and text classification, which can further branch into deep learning techniques (for example, convolutional neural networks (CNNs) and long short-term memory (LSTM) architectures which are discussed in later sections), and different text processing techniques.

A prevalent topic which forms the basis for receipt scanning for data extraction lies in research for automation for the Household Budget Survey (HBS). The Household Budget Survey is a survey that many countries (specifically EU member countries) ask their citizens to complete to gauge consumer expenditure at a national level [15].

Benedikt et al. [16] explore an automation based approach to speed up the traditionally manual approach of processing 2-weeks' worth of shopping receipts.

One of the main difficulties of text classification of receipt items involves shortened product codes where full words are often abbreviated and hence may result in the learning algorithm returning a prediction with low confidence (as the machine has not seen the new abbreviation before). This also applies to new or rare products, and how well the model can abstract on unseen data can vary. Other difficulties include distinction

between classes – such as unsliced and sliced bread being in different categories, or perhaps drinks and alcohol (this introduces a hierarchical component to the task as alcohol could be classified under drinks); and imbalanced datasets – where everyday items such as *'eggs'* and *'milk'* will appear more often than *'handwash'* for example, which leads to the potential problem of a model performing well on the items that appear more frequently, but less accurately on under-represented items.

To mitigate these difficulties of receipt analysis and product classification that prevent machine learning from reaching 100% accuracy on its own, Benedikt et al. propose a Human-in-the-Loop (HITL) solution [16]. HITL refers to a form of supervised AI models that require human input for training [17]; for example, involving feedback on predictions which are then used to further train the model. HITL implementations often involve a UI for humans to submit label verification. This is similar to how Nanonet's API functions [3]. HITL is more of a framework than a learning algorithm and when to retrain a model is decided by a human admin. The proposed HITL loop as proposed by Benedikt et al. is shown in Fig. 1.

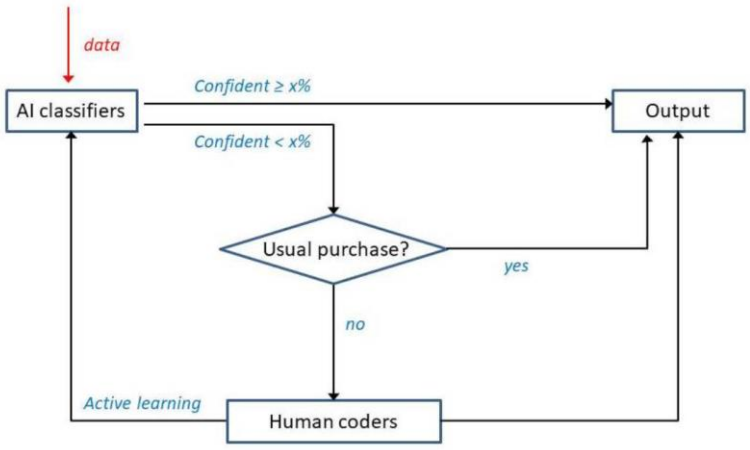


Fig. 1 - The human-in-the-loop architecture proposed in [18]

Another difficulty of receipt classification involves the format of the acquired receipt image. Benedikt et al. detail the benefits of using office scanned photos in [18] which allows for a lot of control in the input, however, this is not accessible for everyday use. The ideal alternative – using photos that anyone can take originating from their smartphones – whilst time-efficient, presents a plethora of difficulties, and results in a significant effect on whether lines of text are missed (or the alternative where extra lines of text are detected). User submitted photos may also be noisy, blurry, distorted, or contain undecipherable text (especially when longer receipts are involved). Raoui-Outach et al. explore extraction of data from pictures of receipts captured from mobile phones and propose the use of Deep Convolutional Neural Networks (DCNNs) for pre-processing and extracting text from an image to determine whether a receipt is present or not [19]. The proposed model of a DCNN coupled with text semantic analysis reached a precision of 99.9% in its purpose of detecting receipts compared to other documents. This was achieved through text analysis performed on related image blocks. As receipt vocabulary is nuanced and domain-specific, they can be distinguished from other documents from the language indicators and positioning of text in the document.

Automatic interpretation of short product labels is especially important in the field of receipt analysis due to the nature of receipt "language". Extracting meaningful product and purchased item data is difficult, as there are often vague and non-meaningful labels that stores will use to depict different products. These will differ between vendors and there is no agreed standard for product codes which adds a level of complexity to the problem.

Janík developed an application to allow users to scan sale receipts and store them in a database [10]. The application uses Microsoft's cloud service Form Recognizer [20] to extract information from receipts to store. The application involves a simple form of item categorisation; the service is tasked with assign an emoji to items from the extracted data; however, this was not a main feature of the developed application, and only reached a 20.8% perceived accuracy in emoji categorisation.

The remaining issues in this domain are linked to overall product classification and providing users with insights as this has yet to be seen as the focus across other packaged implementations or research. Furthermore, limitations in receipt data such as confusing abbreviations (which even humans may struggle to decipher and categorise) and the accuracy of mobile camera OCR contribute to the difficulty of the task. The ideas introduced from previous research in the area provide a few starting points for the important components of the work to follow – including the initial processing of images into text to the classification of the extracted data.

2.3 Supervised deep learning techniques

The overall state of AI theory and working models in the fields of OCR, KIE, and text classification have seen significant advancements in research and theoretical work derived from applying various deep learning techniques. The theory behind modern machine learning techniques and algorithms will be discussed in this section.

2.3.1 Artificial Neural networks

Artificial neural networks (ANNs) are AI models that are modelled on the human brain. ANNs simulate the biological neurons in the brain and consist of layers of nodes or perceptrons representing neurons that are connected together. One node or perceptron typically performs computations on several inputs to produce a singular binary output (a one or zero). Inputs represent features, and each feature is weighted. The weight of an input signifies the importance of the input to the output. The value from the calculations performed by the neuron is then compared against a parameter called the threshold value, where if the value is greater than the threshold then the output is 1, otherwise the node will return 0.

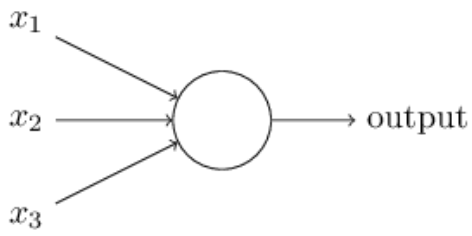


Fig. 3 – An example of a perceptron

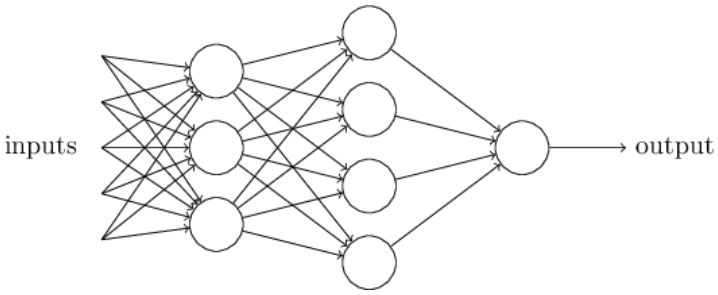


Fig. 2 - A shallow neural network

A single layer in a neural network could consist of one or multiple nodes. The outputs of these nodes can then be connected to other nodes in another layer. This is referred to as a multi-layer perceptron model as depicted in Fig. 3. Any layers between an input layer and an output layer are referred to as a *hidden layer* and the nodes in the layer are referred to as *hidden nodes* [21].

A supervised implementation of a neural network will adjust the weights of any nodes in the model based on passing through training data inputs and considering the effect of weight changes on outputs. Minimising the error of the network on training data can be achieved by updating weights using a gradient-based optimisation method such as stochastic gradient descent (SGD), RMSprop, or Adam optimiser [22]. Ruder states that adaptive-learning rate methods are likely to achieve the best results for sparse input data and suggests that Adam may be the best choice for an optimiser as it performs slightly better towards the end of optimisation.

Shallow neural networks may consist of only one or two layers whereas deep neural networks often employ multiple layers of different types. Deep neural networks can learn detailed and multiple levels of representations of data and can achieve significantly better accuracy than shallow neural networks. However,

training deep learning models can require high computational costs, and are usually non-transferable, as the learned features are dependent on specific applications [23]. Deep neural networks are also difficult to train due to the vanishing gradients problem explained in more detail in [21]. Additionally, depending on the input data, the more complicated a neural network is, the more likely it can be overfit on training data and abstract badly on unseen data.

Different variations of machine learning algorithms, including classical and deep learning models, will be discussed in the next sections.

2.3.2 Traditional classification models

Machine learning algorithms for classification include logistic regression (LR) algorithm, naive Bayes (NB) algorithm, and support vector machines (SVM). All three are simple forms of a neural network used in supervised learning that require labelled data for training.

Logistic regression classifiers can be used for both binary classification and multinomial classification (also known as softmax regression) and is important in machine learning as it is related to neural networks (where a single layer neural network is essentially a logistic regression classifier). Logistic regression learns weights for individual features through fitting data to a sigmoid function (depicted in Fig. 4.). It is a probabilistic model that tries to maximise the probabilities of the data, where the class with the maximum probability is the predicted class.

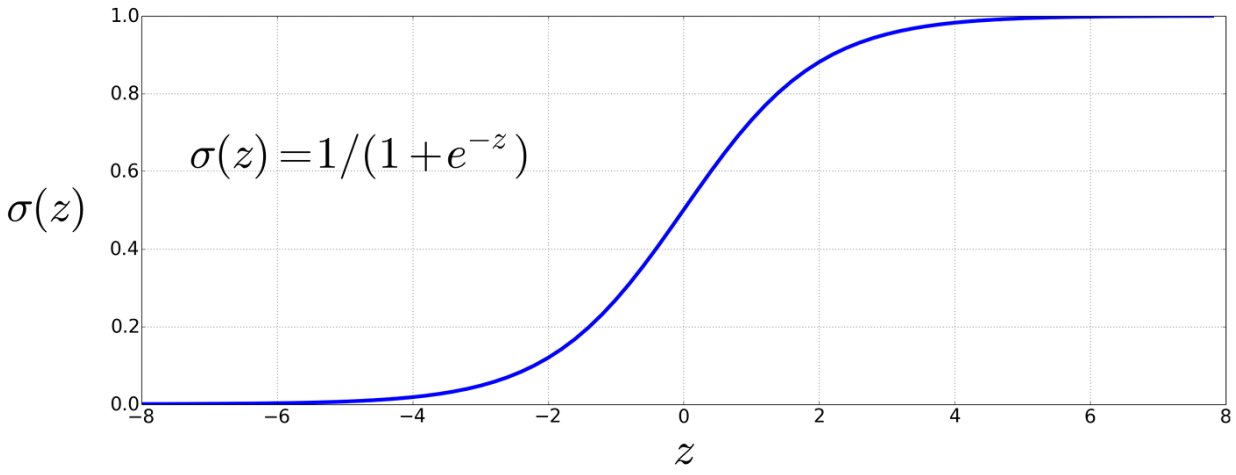


Fig. 4 – A graph of the sigmoid function [24]

The sigmoid function can then be used to calculate the probability that a word is a certain class – by saying that if the probability of a word being a certain class is more than the decision boundary (usually 0.5) then the word belongs to that class. For multilabel classification, softmax is a generalisation of the sigmoid function, and takes a vector of values and maps them to a probability distribution, where the highest probability dimension in the output is the class.

The naive Bayes classifier is also a probabilistic classifier but differs from logistic regression as it is a generative algorithm rather than a discriminative algorithm. Discriminative algorithms such as LR only try to differentiate between different classes whereas generative algorithms build a model that could generate input data. The naive Bayes classifier uses Bayesian theory to classify texts based on training data. Bayes' theorem predicts the probability of an event occurring given the probability of another event having already occurred so:

$$P(x|y) = \frac{P(y|x)\,P(x)}{P(y)}$$

where x and y are features and $P(y) \neq 0$. This can be used for classification by letting x be the class label and y to be a set of features (or a document) such that $y = f_1, f_2, \ldots, f_n$, which means the probability of a document to be a certain class c is:

$$\frac{P(f_1, f_2, \ldots, f_n \mid c)\,P(C)}{P(f_1, f_2, \ldots, f_n)}$$

This is calculated for all the possible classes and the maximum probability class is presented as the prediction. A support vector machine (SVM) is a discriminative algorithm like logistic regression but look for optimal solutions in n-dimensional space, where n is the number of features, and the value of each feature is the co-ordinate. An SVM algorithm aims to find the optimal hyperplane in the n-dimensional space. In their simplest form, SVMs differentiate between two separate classes, but can be used for multiclass labelling problems by breaking down the problem into various binary classifications as shown in the figure below.

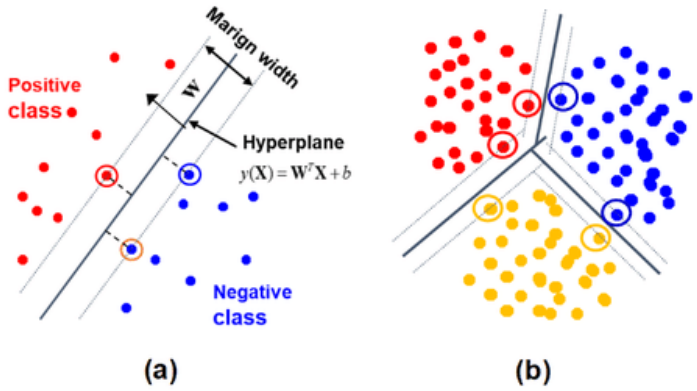


Fig. 5 - Visualisation of a SVM classifier

SVMs can also learn non-linear separations by using kernel functions where data points are transformed to higher-dimensional mappings for better division. This is also referred to as a kernel trick which is visualised in Fig. 5.

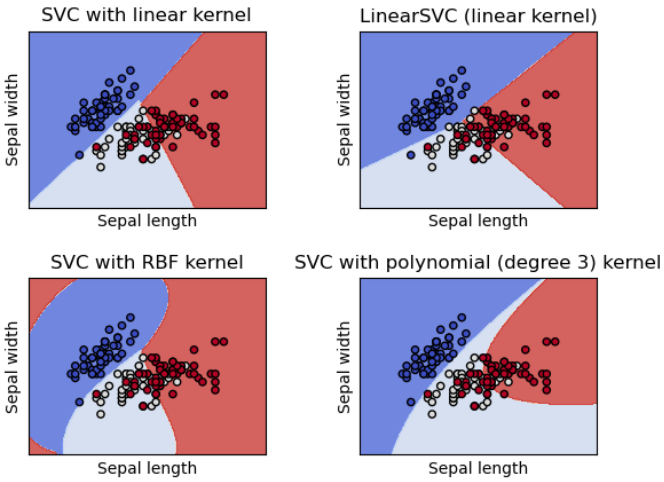


Fig. 6 - Example of SVM kernel trick [25]

2.3.3 Convolutional Neural Networks (CNNs)

Convolutional neural networks differ from regular neural networks as the neurons between layers are not fully connected. Convolutional neural networks were designed to model grid-structured inputs, where the spatial features of inputs are important; most notably, this applies to images. For example, if all the pixels of a picture were treated as inputs in a neural network, a fully connected neural network could quickly grow to an unmanageable number of weights. Additionally, the spatial relationship of each pixel is not considered.

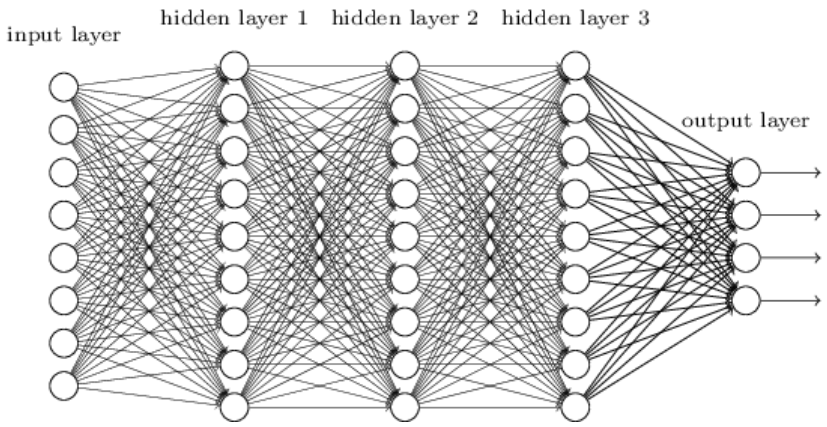


Fig. 7 - A fully connected neural network [21]

Where a classical neural network cannot map the spatial relationship between information well, CNNs perform exceptionally well on data such as images which exhibit spatial dependencies.

The main components of a convolutional network are the convolutional layers and the pooling layers. A traditional neural network layer is one dimensional, whereas a convolutional layer has neurons that are arranged in two or three dimensions. A 28 x 28 image in black and white can be represented as a convolutional layer of 28 x 28 input neurons in a grid like structure like in Fig. 8.

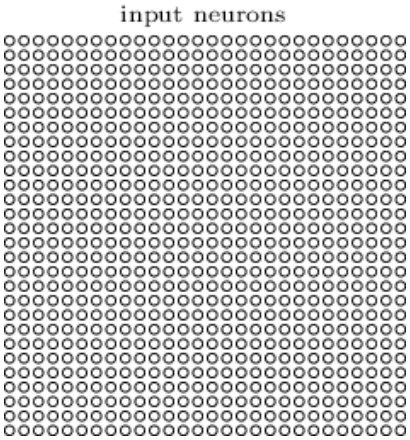


Fig. 8 - Neurons structured in a grid-like structure to represent an image [21]

These neurons can be mapped to another layer of neurons by convolving over the image with a filter or kernel. A convolution is when a matrix is applied to a section of the image to generate a convolved feature, which is then passed to the next layer.

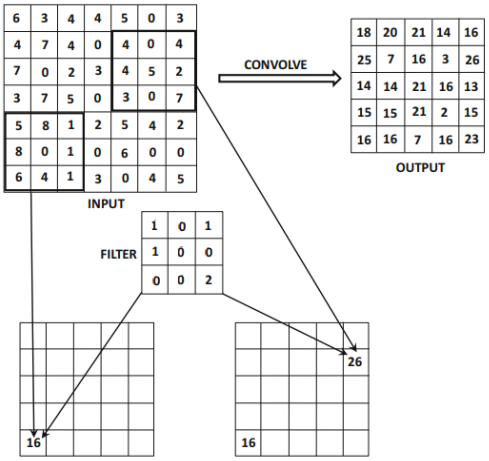


Fig. 9 – An example of a convolution step

Specific kernels and filters perform different functions and extract different features from an image, for example there are kernels for horizontal edge detection, and corner detection. Beside convolutional layers, CNNs also feature pooling layers. Pooling layers condense the outputs from a previous layer and essentially acts to summarise the features found in a region. For example. A max-pooling layer will return the maximum activation in the specified region, whereas average pooling will return the average activation.

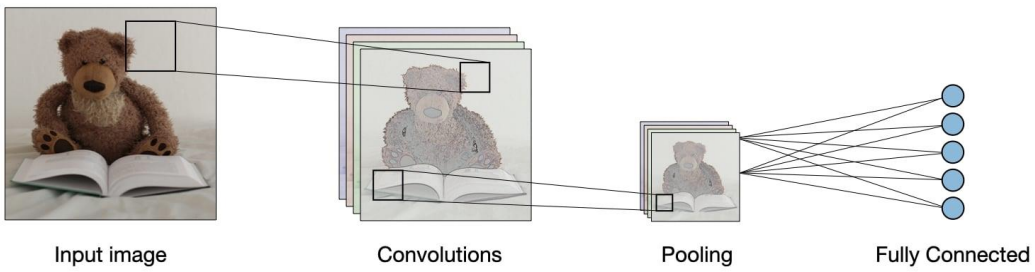


Fig. 10 – An example of simple convolutional layers in a CNN

Further into a CNN, more complicated features such as faces, or objects can be identified from features. Each neuron in the input, and each layer of convolutions have their own weights which are optimised for the given problem.

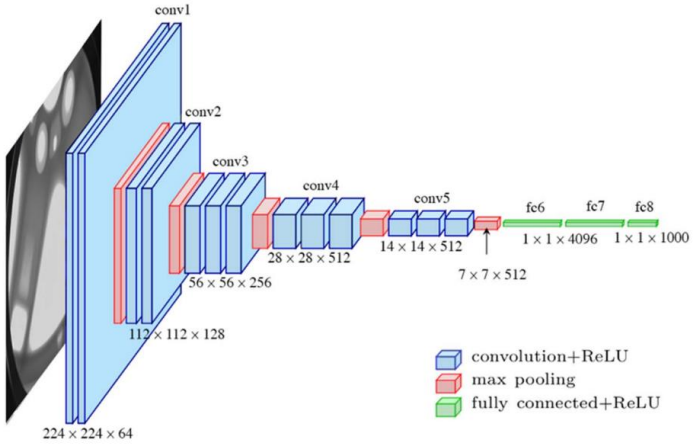


Fig. 11 - CNN layers used in VGG19

Examples of successful CNNs for image recognition include VGG16-19 [26] and ResNet [27] which both achieve over 90% accuracy on recognising images from ImageNet (a dataset of over 14 million images belonging to 1000 classes), which are significant improvements on previous results produced from shallow neural networks.

Although CNNs are generally used for vision tasks, the concepts of a CNN can be transferred to other applications such as for NLP.

2.3.4 Recurrent Neural Networks (RNNs)

Recurrent neural networks are another class of ANNs that process sequential data that have seen success when used for NLP and other tasks that feature time-dependent data. Applications of RNNs include music generation, text classification, and stock predictions amongst others.

RNNs differ to CNNs and other feed-forward neural networks as data progression through an RNN is non-linear. The order of inputs in retained in an RNN, and information from previous inputs can influence other output activations which allows for dynamic analysis of data that changes over time.

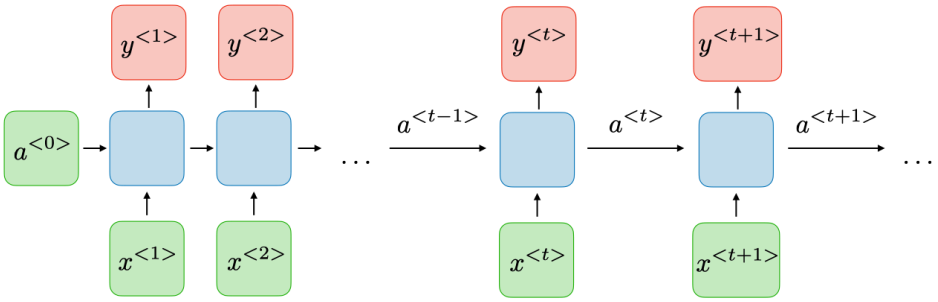


Fig. 12 - Architecture of a traditional RNN [28]

LSTMs (Long Short-Term Memory networks) are a variation of RNNs that include additional units known as *memory cells* that retain information in memory across long periods of time. This allows the models to learn long-term dependencies, which RNNs are not able to do due to short-term memory and vanishing gradient problems [29]. GRUs (Gated Recurrent Units) are a simplification of an LSTM unit, and work by incorporating "gates" into the model architecture. The gates in a GRU or LSTM unit can learn which data is relevant and whether to retain or discard this information for later use.

LSTMs are traditionally forward directional and only preserve information from the past. Bidirectional LSTMs use data from both the past and the future in determining outputs.

2.3.5 Transformers

Whilst LSTMs and GRUs retain longer memory than traditional RNNs, they still suffer from short-term memory over a longer period. Additionally, the sequential nature of RNNs prevents easy parallelisation. In 2017, Vaswani et al. introduced the Transformer architecture that uses attention mechanisms to solve sequence problems [30]. The framework features an encoder-decoder architecture and a self-attention mechanism that allows the model to make predictions and capture context in both short and long distances.

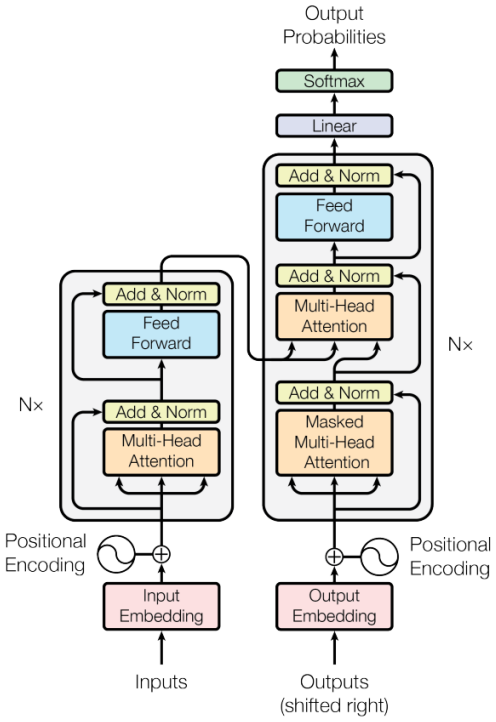


Fig. 13 - The Transformer model architecture [30]

The introduction of transformers was a pivotal moment in AI development, as the Transformer model achieved accuracy that was previously unseen in NLP. Since its introduction in 2017, many models based on transformer architecture have been released including vision transformers [31], that require less resources than DCNNs to train and perform to a similar or better standard.

Currently, transformer technology is seen as the state-of-the-art and commands the most attention within the ML community. Transformers have already surpassed performance of RNNs and CNNs, and its full potential is still being researched.

2.4 Optical Character Recognition Technologies

This section will discuss current and up-to-date open-source OCR tools and the technologies behind them. Although, OCR for documents in a closed environment (i.e., scanned documents) is generally seen as a "solved" problem – scene text recognition (STR, also known as "text in the wild") still presents many challenges. An example of scene text OCR includes recognising and understanding road signs or capturing car registration numbers. Research in improving OCR for scene text recognition focus on recognising text even in unfavourable conditions and is relevant for the processing of documents photographed on a mobile phone.

2.4.1 Pre-processing

Many difficulties in producing accurate OCR output directly result from the quality of source pictures. This is especially important to note as this project aims to extract text from pictures taken on mobile phones. While mobile phone camera quality is frequently improving (which minimises the problem of having to process low-quality pixelated photos where characters on receipts may have previously been too small to detect), there is still the significant issue of human error when taking photos. For example, some users may have shaky hands which could result in blurry images; other users may take images of receipts in bad lighting; or receipts may just be in bad shape and contain distortions. Whilst detrimental to the overall accuracy of character recognition, most of these types of errors can be mitigated with correctional pre-processing, such as identifying corners of documents for cropping, adjusting contrast and sharpness, and other image enhancements.

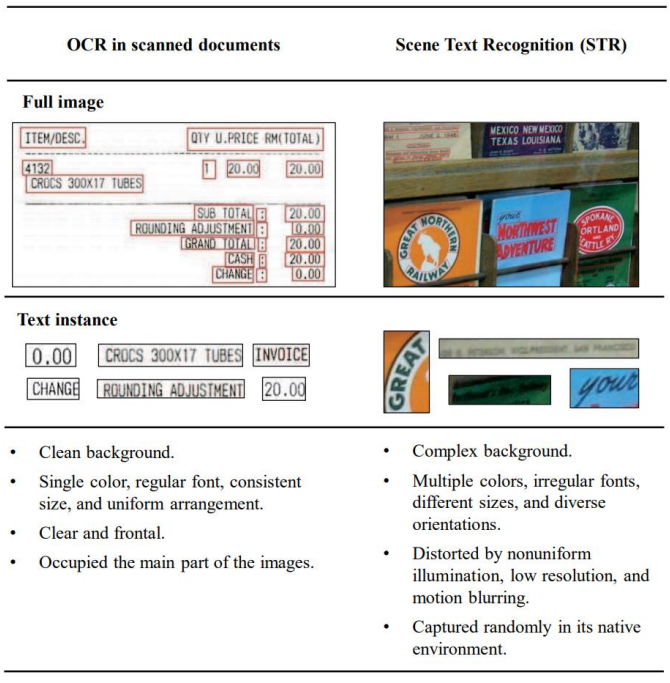


Fig. 14 - A comparison of OCR in scanned documents compared to STR [32]

El Harraj et al. demonstrate in [33] that significant improvement in the number of errors and the accuracy of OCR (around 2 to 6.8 percent improvement) to extract text can be achieved after performing different image enhancement steps.

2.4.2 TesseractOCR

TesseractOCR is an OCR tool that was developed by Hewlett-Packard between 1984 and 1994 [34]. It was later released as an open-source package by Google and is one of the most popular open-source OCR engines today.

The legacy versions of Tesseract up to Tesseract 4 used segmentation-based character recognition and an adaptive classifier. Though the legacy OCR system is still available, newer architectures introduced later perform better.

According to their documentation, the most up-to-date version, Tesseract 5, builds upon the previous versions by incorporating LSTM neural networks into its engine and is available for over 100 languages. The LSTM model is adapted to C++ from OCRopus [35] which is a Python-based open-source OCR system introduced by Google. The below diagram shows the network architecture of the LSTM-based OCR method as described by Ul-Hasan in [36] which outperformed legacy Tesseract systems across different scripts and languages.

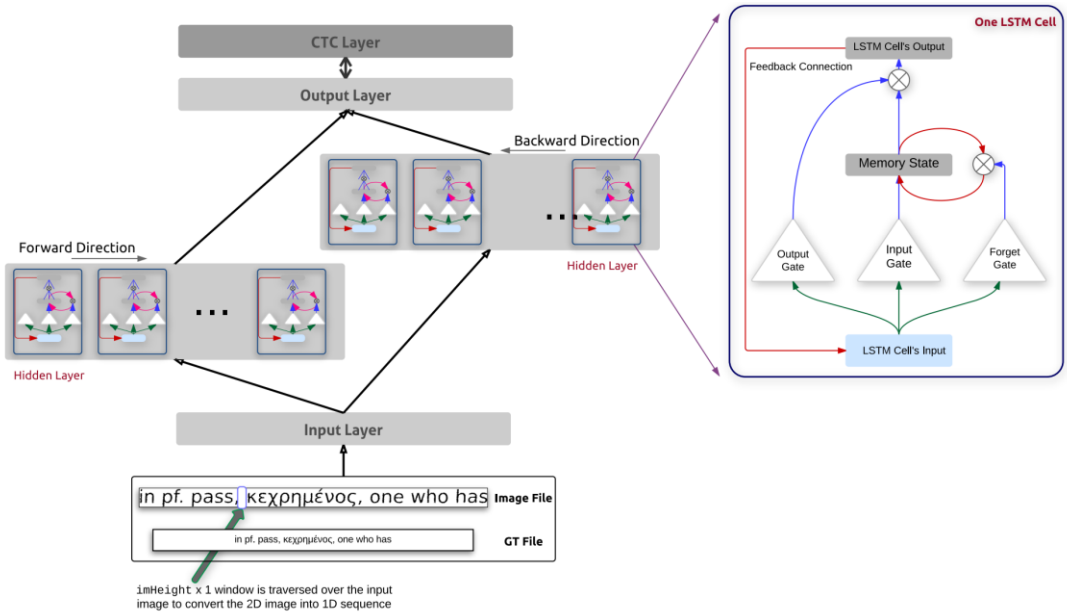


Fig. 15 - The generalised OCR methodology using LSTM proposed by Ul-Hasan

Additionally, previous Tesseract models could not be used to recognise multiple languages together and relied heavily on language modelling and post-processing. Ul-Hasan demonstrates that LSTM OCR can be used for language independent OCR and still yield effective results.

LSTM architecture was adapted for Tesseract in C++ and released in Tesseract 4.0 as presented by Smith in [37].

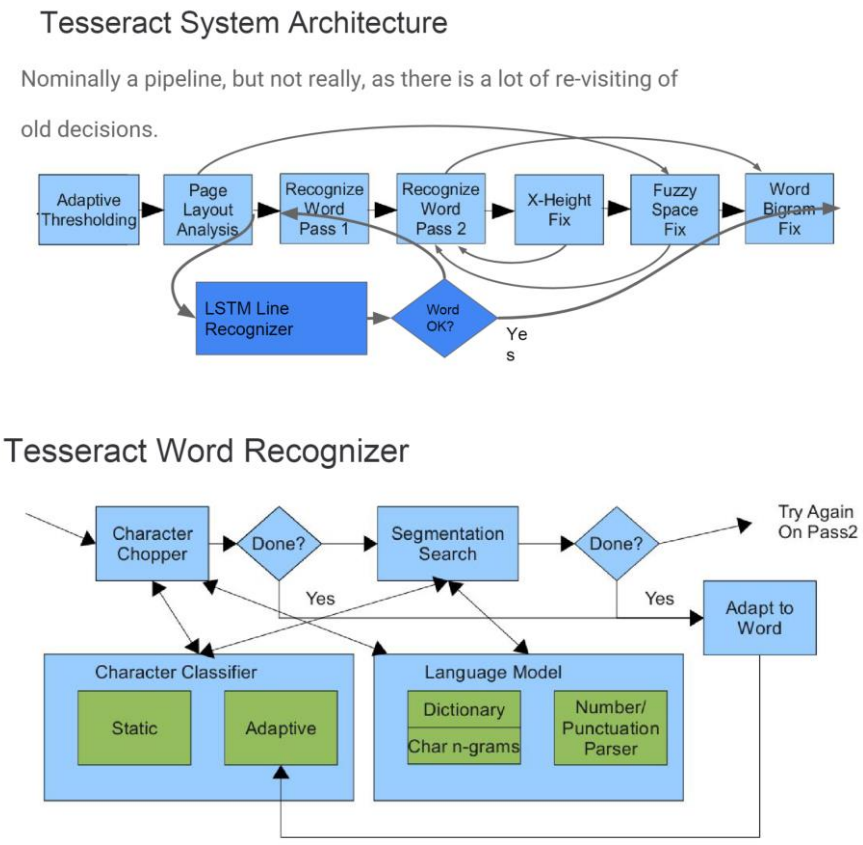


Fig. 16 - Tesseract OCR flow

2.4.3 Deep learning OCR packages

PaddleOCR and EasyOCR are two relatively new open-source OCR packages that utilise state-of-the-art deep learning techniques and research to present end-to-end deep learning pipelines for text detection and recognition. Whilst Tesseract performs well on scanned documents, complexities in the image such as distortion, mirrored text, or noise affect the effectiveness of character recognition. Recent deep learning techniques have aimed to improve the performance of OCR components, and EasyOCR and PaddleOCR present implementations employing models as presented by new research.

PaddleOCR is an OCR system developed by the Chinese company Baidu based on their PaddlePaddle framework (PArallel Distributed Deep LEarning). PaddleOCR is presented in [38] as a super lightweight OCR system that is able to effectively recognise Chinese and English characters, with the overall model size being 2.8MB for alphanumeric symbols. PaddleOCR follows the processing pipeline below.

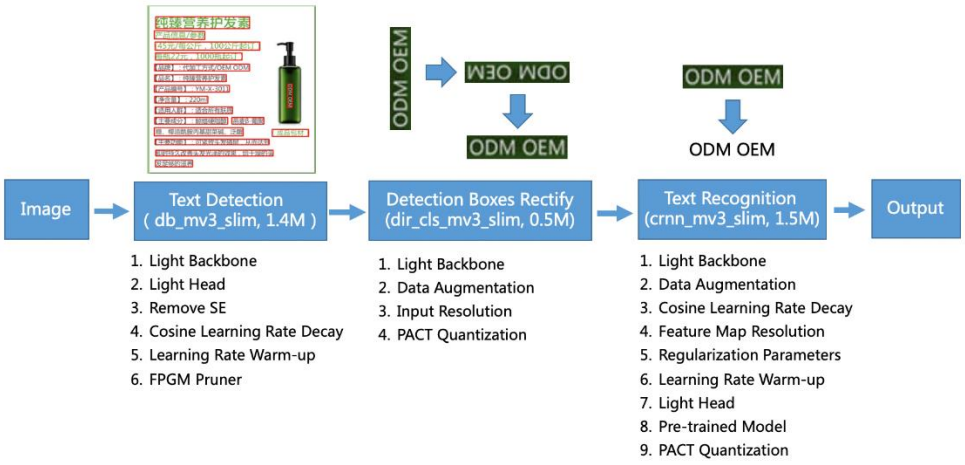


Fig. 17 – The PaddleOCR pipeline

PaddleOCR recognises text in 3 stages:

1. Text detection (position); first text is detected using Differentiable Binarization (DB) [39] with MobileNetV3 (a convolutional neural network introduced in [40]) as the backbone of the model with additional convolutional layers.
2. Text orientation and rectification; once the text bounding box has been detected, the direction of the text is determined by the direction classifier and geometrically transformed into the correct direction (including reversal or mirrored text)
3. Text recognition (content); finally CRNN (convolutional recurrent neural network) as introduced by Shi et al. in [41]) is used for text recognition to output the predicted text

Although the image-to-text pipeline that EasyOCR [42] implements is similar to PaddleOCR (without text orientation detection), the models and backbones used in each differ:

1. Text detection (position); first text is detected using the pretrained CRAFT algorithm introduced by Baek et al. in [43] which is a fully convolutional neural network based on VGG-16 [44].
2. Text recognition (content); EasyOCR also implements CRNN for text recognition.

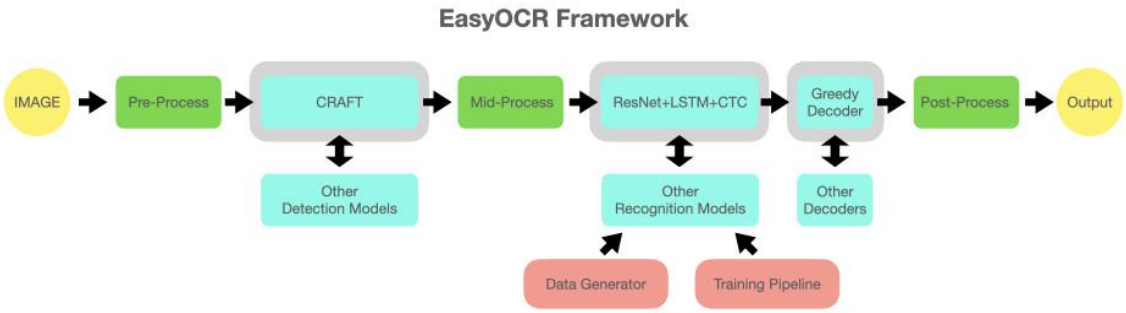


Fig. 18 - The EasyOCR framework

Both OCR tools also support multiple languages and scripts and are licensed under Apache-2.0.

2.4.4 CRNN

As CRNN is used in both PaddleOCR and EasyOCR, this section will discuss the text recognition method in further detail. CRNN was introduced in [41] to combine the benefits of DCNNs and RNNs in an end-to-end scene text recognition model. This allows the model to be trained with a singular loss function even though CRNN is composed of different architectures. The following image shows the proposed network architecture.

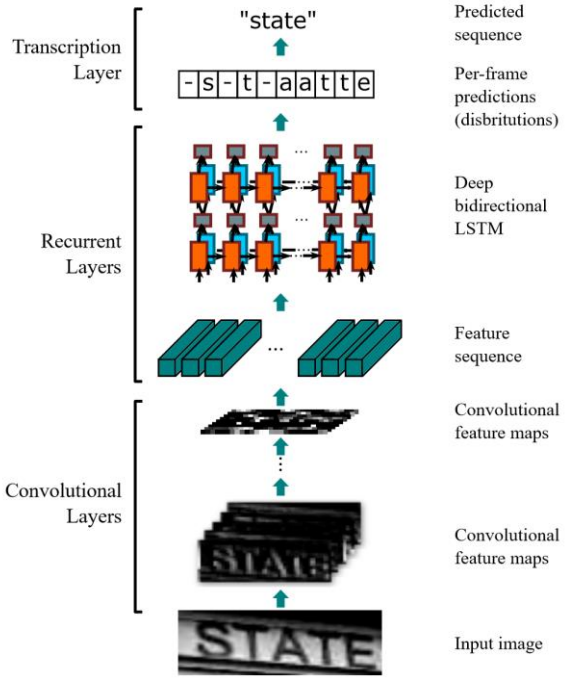


Fig. 19 - CRNN network architecture

In CRNN, convolutional feature maps taken from a convolutional layer and max-pooling layers are passed as feature sequences to a deep bidirectional RNN that can then produce a sequence of labels. Each column of the feature map corresponds to a region on the original image and the feature sequence is each region in the same order from left to right, as depicted below.

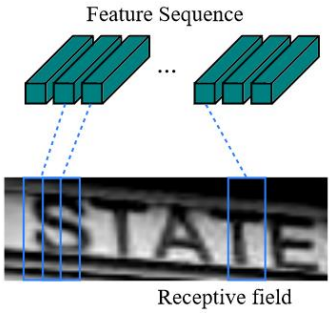


Fig. 20 - An example of how CRNN feature sequences are mapped to receptive fields

The CRNN architecture contains no fully connected layers which results in a less bulky and more efficient model than most deep NNs and performed better than other DCNNs at the time of its publication, which lead to it being the model of choice for EasyOCR and PaddleOCR.

2.5 Key Information Extraction Technologies

This section will review literature and techniques found in the AI space for key information extraction from visually rich documents (VRDs), specifically in relation to receipts. VRDs are any documents that contain large amounts text where visual structuring of the text may also provide indication to the contents of the document. Receipts are one example, but other instances of VRDs include terms and conditions, contracts, forms, and invoices. In each of the mentioned documents, the spatial positioning, structuring, and grouping of text on a page is crucial to the understanding of the document. Key Information Extraction in this context refers to the task of returning key fields from a document in a specified format for further use or processing. Examples include purchase date from invoices, name, and date of birth from forms, etc. Research in this space has seen recent innovations with some of the discussed articles and models released in the current year (2022).

2.5.1 ICDAR 2019

In the OCR and KIE space, The ICDAR (International Conference on Document Analysis and Recognition) Robust Reading Competitions (RRC) [45] have been significant catalysts for the publication of methods relating to document understanding and OCR. On their website, they refer to robust reading as "the research area dealing with the interpretation of written communication in unconstrained settings". Competitions held from 2011 encompass many real-world situations and the community's solutions for these challenges typically showcase state-of-the-art technologies and their evolutions. Their website lists over 33,000 registered users with 1,366 published public methods.

In 2019, ICDAR hosted a challenge of particular interest to this paper – the Robust Reading Challenge on Scanned Receipts OCR and Information Extraction, also known as SROIE [46]. In this challenge, users were provided with a novel dataset of receipts as introduced in [47] and asked to perform 2 tasks: text detection and recognition and key information extraction. Many of the methods developed during this challenge are still relevant today or have influenced further works that are now seen as state-of-the-art methods (such as LayoutLM [48] and StrucTexT [49] which will be discussed in detail). The SROIE dataset is also often used as a benchmark for new KIE methods published after 2019.

2.5.2 Traditional Key Information Extraction

Key Information Extraction has many ties with NLP, specifically with Named Entity Recognition (NER) tasks, which aims to extract relevant information from raw text.

Typically, with simpler datasets and use cases, information extraction can be performed through using regular expressions (regex) of varying degrees of complexity to identify key fields. For example, an email address will always have an '@' symbol, and can be extracted using a simple regular expression such as:

$$[A-Za-z0-9]+@[A-Za-z0-9-]+(\.[edu|ac.uk]\{2,\})+$$

which would recognise an educational email ending in .edu or .ac.uk. However, creating robust regular expressions can quickly become overcomplicated; the standard email address format also known as the RFC

5322 [50] describes the syntax of an email address, and can be implemented with the following regex as described in [51]:

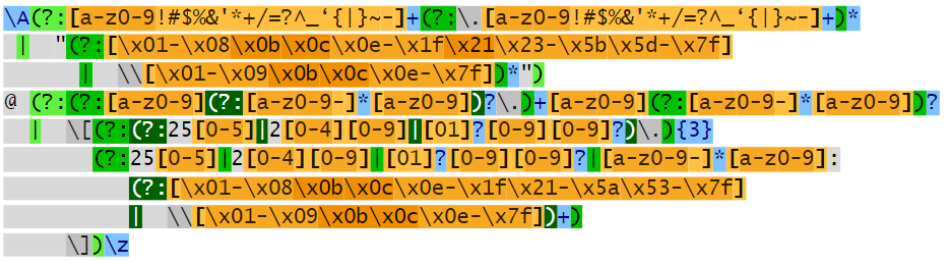


Fig. 21 - Standard email regular expression

Regular expressions can quickly become convoluted whilst still not being able to capture all fringe cases. Additionally, regular expressions have traditionally required manual creation and maintenance, though methods for learned regular expressions have also been explored [52]. Other limitations of regex include struggling to distinguish context (such as "William" being a name whilst "William Street" is a street); struggling to abstract on new unseen data or data that changes a lot which would require constant rule updates; and any fields that do not have rules (and therefore cannot be easily reduced to a regular expression).

Other traditional methods for KIE essentially simplify KIE to an NLP problem and do not utilise other information that may be available from a document (such as layout and text positioning). For example, in Li et al. present a solution for NER utilising LSTMs that reached a respectable 90.94 F1 score [53]. Whilst the problem of KIE does share similarities with NER, they are not the same task. More modern key information extraction methods rely not only on the textual data fetched from documents, but also utilise the spatial and visual information of documents to extract relevant information.

2.5.3 Grid-based methods

In order to preserve and communicate the spatial positioning of a document to machine learning models, methods such as CUTIE [54] and Chargrid [55] map documents to vector based representations and input these vectors into deep neural networks to extract key information. Both CUTIE and Chargrid propose methods that preserve layouts of documents by creating grid positional mappings where the positions of text on the document is reflected and passed to the model.

Chargrid directly maps each character to a pixel grid representation of a document and applies one-hot encoding to the chargrid, where blank spaces or padding is represented as a 0 and other characters are mapped to an encoding. In Chargrid, even visual information such as font size is retained, which would other not be available. An example Chargrid representation of a document is shown below.

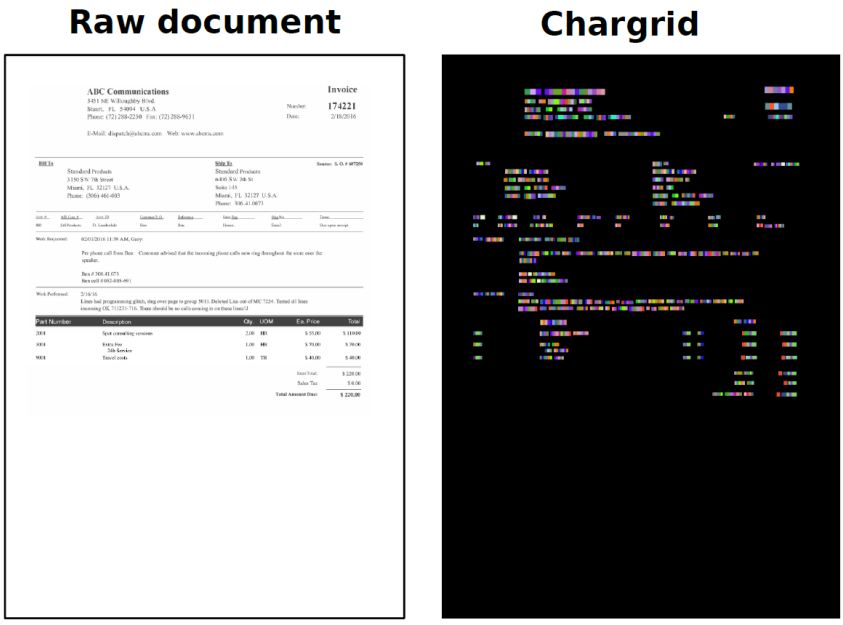


Fig. 22 - An example of a Chargrid representation of a document [55]

This representation of the text is then inputted into a convolutional neural network to predict class labels for each pixel in the grid. Chargrid performed better on KIE than traditional NLP algorithms or image only models but requires a large computational cost due to the one-hot encoding and chargrid representation.

CUTIE also represents documents as a grid but uses a different method for grid positional mapping. The model architecture is represented below.

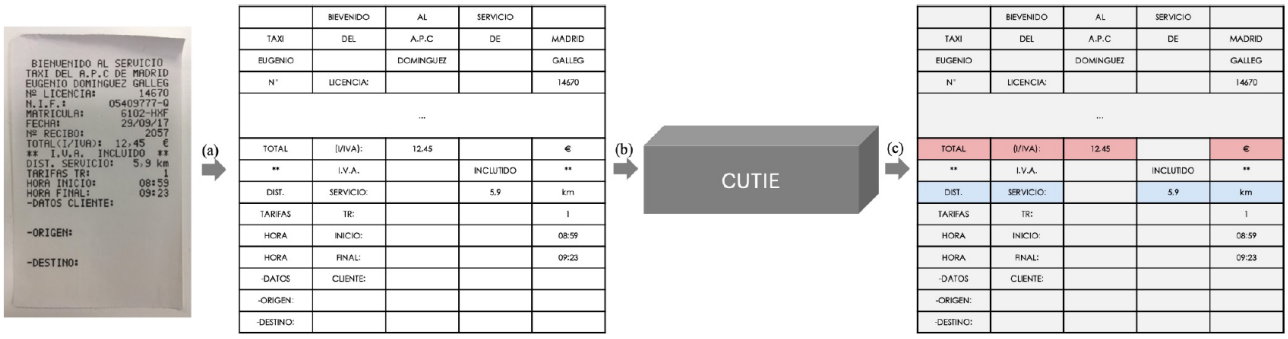


Fig. 23 - Example of CUTIE's grid method on a receipt [54]

CUTIE was presented with two different models, CUTIE-A and CUTIE-B, where both use word embedding layers for processing, and are neural network structures. CUTIE-A maintains high-resolution representations throughout convolutions, whilst CUTIE-B employs atrous (or dilated) convolutions. Both models perform better on SROIE than traditional NER and requires less computational power than Chargrid.

2.5.4 Graph-based methods

Graph-based models or embedding based models form a large area of research in the KIE space. Graph-based models represent text as and their spatial locations as a vector representation (or layout and visual embedding). The embeddings of text positions are learned by an algorithm and then combined with text embeddings or another representation of text. Recently published graph-based models include StrucTexT [49], PICK [56], BROS [57], GC-BiLSTM-CRF [58], GraphDoc [59], and LayoutLM [48] amongst others. The graph convolution of a document method used in [58] is represented in Fig..

Fig. 24 - Graph convolution of a document [58]

The other methods all propose different methods to map the spatial embeddings of an image, and then combined with different learning algorithms and architectures to perform KIE using multi-modal information embeddings. The multi-modal embeddings contain the visual embeddings, and the token or word embeddings of the document as generated by an OCR engine. Compared to grid-based methods, the graph-based methods overall perform better on KIE tasks and require slightly less computational power (depending on the embedding dimensions). These and similar graph-based methods show that incorporating spatial information on a document is beneficial in the extraction of key information from VRDs.

2.5.5 LayoutLM

LayoutLM as mentioned in the previous section is a graph-based model. However, it differs slightly from other graph-based models mentioned in that it proposes an architecture that utilises word embeddings, image embeddings, and layout embeddings together with transformer architecture to form an effective framework for KIE. Presented by Xu et al. in [48], the bounding boxes of text as generated by OCR is used to form 2D position embeddings as well as image embeddings for each word before being inputted into a BERT-inspired transformer model. The model architecture is presented below.

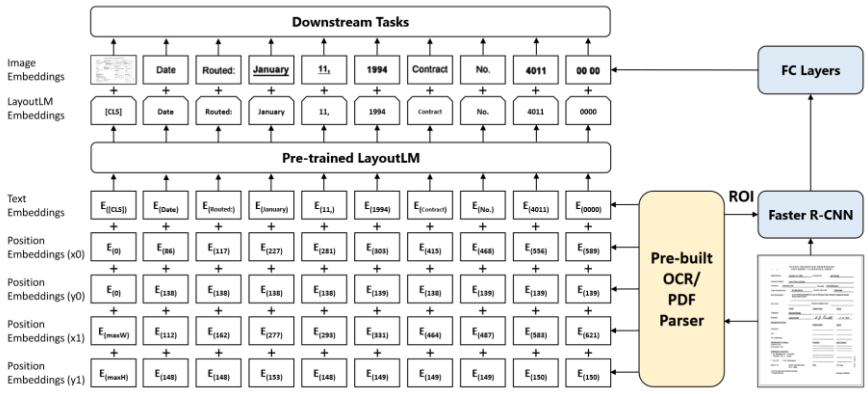


Fig. 25 – Architecture of original LayoutLM model

The original LayoutLM reaches an F1 score of 0.9524 and comparable accuracy for other benchmark datasets, and subsequent adjustments on the LayoutLM structure significantly improve on this. LayoutLMv2 [60] reaches an F1 score of 0.9781 on SROIE, and similar improvements on the other datasets. Where LayoutLM only uses visual layout embeddings during fine-tuning, LayoutLMv2 includes visual embeddings during pre-training. Additionally, the self-attention layers in the original LayoutLM are improved upon by adding relative 1D attention biases as well as 2D spatial attention biases to the attention scores. The LayoutLMv2 architecture is shown below.

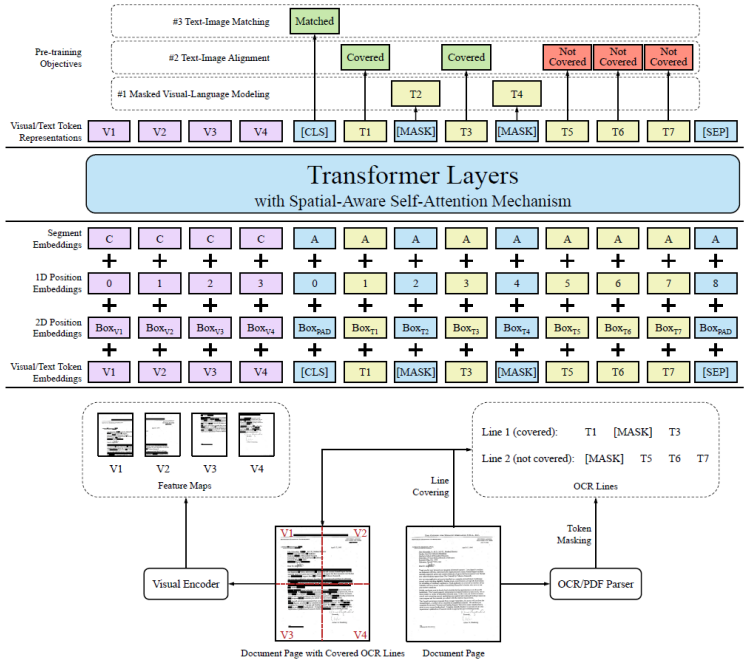


Fig. 26 - Architecture of LayoutLMv2

2.6 Text Analysis and Classification

This section of the literature review will cover recent and up-to-date developments and models in the field of text classification. The theory behind current solutions will also be discussed including word embeddings, traditional algorithms, and modern models for classification.

2.6.1 Vectorisation techniques

In order to train a classification model on text, the text data must first be converted into a format that a computer can parse. For text data, this is usually in the form of a vector. Over time, various methods have been developed to transform words in documents to a numerical representation.

This includes bag of words (BoW), bag of n-grams, using skip-grams, TF-IDF (term frequency-inverse document frequency), and word embeddings (which will be discussed in the next section).

A BoW is the simplest form of numerical word representation. A BoW is essentially a word count that tracks the occurrence of words (or tokens) in text and is produced by counting how many times each token appears in a sequence. First, a vocabulary is created based on the document domain (the set of unique words across the documents) which forms a vector with dimensions that are equal to the size of the vocabulary. Then, for each document, each dimension is counted (0 if the word is not present, 1 if it appears once, 2 if it appears twice, and so on), to create the feature vector for that specific document.

As mentioned in [61] the limitation of this approach is that the semantic meaning of words is lost; the position of the words in a sentence or document and its relationship to other words is not represented.

A bag of n-grams is similar to a bag of words in that it is also a flat vector that represents a count of how many times a token appears in a text. However, in a bag of n-grams, the tokens are n-grams rather than singular words. An n-gram is a sequence of n words, so a uni-gram is an n-gram where n equals one, and a bi-gram is a n-gram where n equals to. Technically, a BoW is a bag of uni-grams. For a bag of n-grams, again a vocabulary is created of the n-grams. For example, for the sentence “Natural language processing is fun”, there are 4 bi-gram tokens: [“natural language”, “language processing”, “processing is”, “is fun”]. N-grams capture the context of words slightly better than unigrams as some words are frequently used together in different contexts (such as the words “New York” meaning something different to when the words are used separately as “new” and “York”).

TF-IDF is another way to represent text as a vector, that performs better than the BoW approach as mentioned in [62], and was shown by Das et al. to produce the highest accuracy, precision, recall, and f-1 score when tested with 6 classification methods against two datasets in [63].

A TF-IDF score shows how important a word is in a document compared to the other words in the document. This is calculated as below:

$$TF - IDF = TF \times IDF$$

$$TF_{(term\ frequency)}(word) = \frac{(number\ of\ times\ the\ word\ appears\ in\ the\ document)}{(total\ number\ of\ words\ in\ the\ document)}$$

$$IDF_{(inverse\ document\ frequency)}(word) = \frac{(total\ number\ of\ documents)}{(total\ number\ of\ documents\ contraining\ the\ word)}$$

Since TF-IDF scores are calculated for words individually, semantic meaning can once again be lost, as in the case for the BoW approach. However, TF-IDF can be performed on n-grams to combine the benefits of both approaches to keep.

The vector representations discussed above largely cannot capture the context nor semantics of text. The vectors are also sparse vectors, which may contain a lot of 0s, with dimensions that can quickly reach computational limitations (as each unique word is one extra dimension in a unigram model and up to n extra dimensions in a n-gram model).

2.6.2 Static Word Embeddings

Static word embeddings are presented as an alternative to the above vectorisation methods, that perform better than sparse vectors in NLP tasks as stated in [24]. These word embeddings are a dense vector representation for words, where words are mapped in vector space in relation to other words, and the meaning of the word based on its' context is encoded into the vector [64].

word2vec [65] is an example of a word embedding algorithm. It features two model architectures that can be used to produce word embeddings: the CBOW (continuous bag of words) model and the skip-gram model. Skip-gram modelling considers words and their surrounding words (n-grams) and can then predict surrounding words when given a word. CBOW is the opposite, where a target word can be predicted given some surrounding context words.

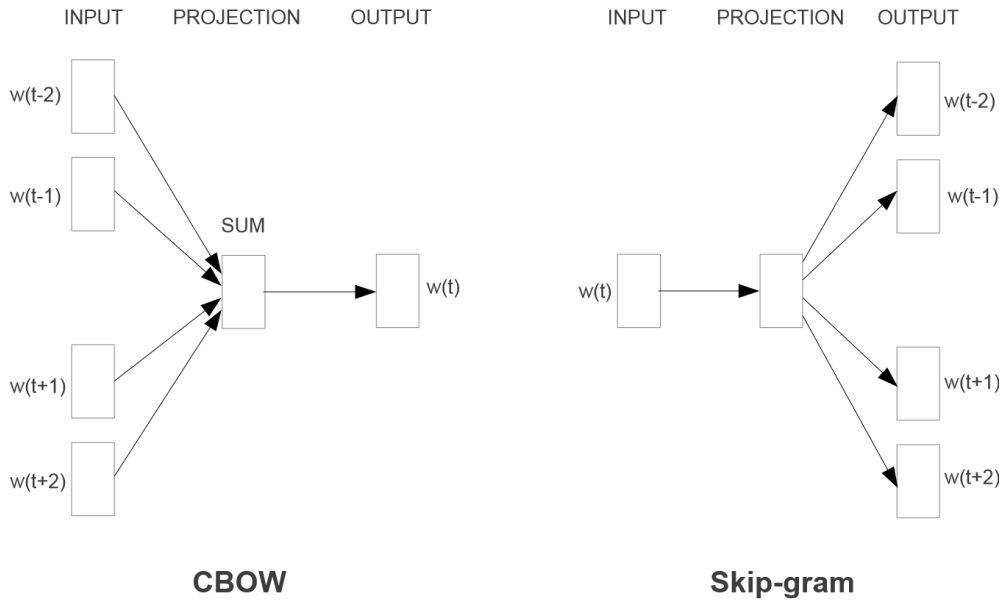


Fig. 27 - A visual CBOW and skip-gram comparison

fastText [66] is an extension of the word2vec model by using a subword model presented by Bojanowski et al., where each word is represented by character n-grams before being passed to the model. In their example the word *where* with n = 3 would generate the following bag of *character* n-grams <*wh, whe, her, ere, re*>. Therefore, fastText learns vectors for n-grams within words, whereas word2vec learns vectors only for complete words. fastText vectors outperformed word2vec vectors in tests against different measures.

GloVe [67] is a static word embedding model that differs from word2vec and fastText presented by Pennington et al. GloVe incorporates the frequency of global co-occurrences (the probability of context words appearing together) into its word embeddings, instead of relying only on local context.

In a comparison conducted by Dharma et al. [68], the performance of all three word embedding techniques are comparable, though fastText achieved the highest accuracy (97.2%) when classifying new stories into 20 newsgroup categories (topics).

2.6.3 Text classification models

Traditional classification algorithms such as SVM and LR have historically performed to a decent standard on text classification problems, for example, SVMs can achieve over 86.0 microaverage as demonstrated by Joachims [69]. CNNs and RNNs also achieve relative success on text classification tasks; Kim's CNN approach on top of pretrained word2vec embeddings outperform other methods [70].

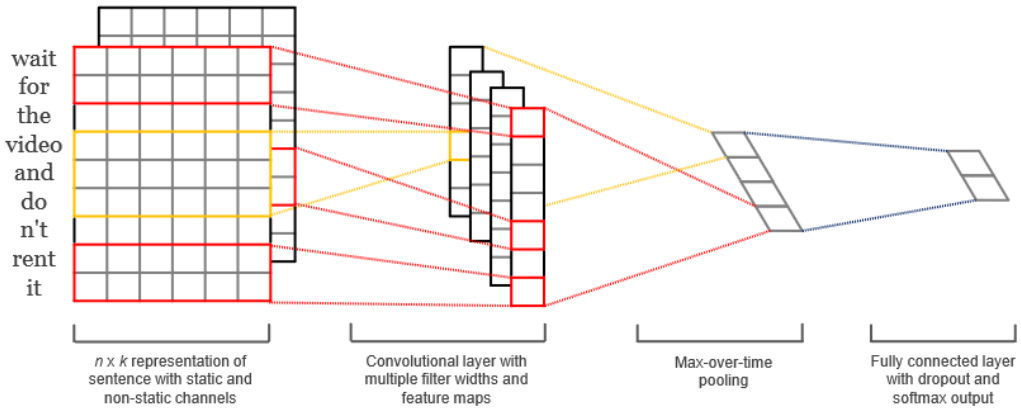


Fig. 28 - TextCNN model layers [70]

There is no singular algorithm that outperforms every other algorithm for any problem, as some algorithms work better on certain tasks whilst performing worse on others. However, most recently, transformer technology is an exception and seemingly outclasses other methods on a range of tasks; BERT [71], RoBERTa [72], and other BERT derivative models achieve state-of-the-art results on text classification and other NLP related tasks.

2.7 Summary

To summarise this chapter; research has shown that current apps on the market do not provide a sufficient solution to the problem of digitising receipts, especially for individuals rather than business users. Additionally, an end-to-end pipeline for the analysis of receipt data does not currently exist.

The individual components required to form such a pipeline were explored, and it can be concluded that recent advancements in the AI and ML communities related to OCR, KIE, and NLP can be used to implement a pipeline that can achieve high accuracy in identifying and classifying products from receipts.

In the next chapter, the insights gained from researching the project domain will be used to decide what experimentation should be conducted to implement the best performing pipeline.

3 Problem Analysis and Design

This chapter of the report will break down the project into its key components and will detail the methods that will be employed to investigate the best solution to the presented problem. This includes high-level overviews as well as more in-depth plans for implementation.

The aim of the project is to investigate the potential of an end-to-end pipeline that can recognise text from a mobile phone picture of a receipt, extract an itemised list of purchases, and classify the purchases into categories. Therefore, each component of an end-to-end pipeline will be analysed and designed.

3.1 General Design Overview

The project will be conducted in the following stages:

Research	During the literature review, relevant research was conducted into current solutions such as existing mobile phone applications. Significant findings that could affect design, development, or later stages were noted.
Problem Analysis & Design	Using the research from the previous stage, we then analyse which methods should be explored and compare options for implementation. This includes finding and analysing suitable datasets, choosing what technologies to work with, finding the best starting points to develop models from, and which models to explore.
Implementation	Once analysis and design has been completed, we can begin implementation of the chosen methods and solutions.
Testing	Then, testing will be conducted on the experiments, which involves measuring the success of the systems and methods created.
Evaluation	An evaluation of the experiments will be performed. The functionality of the proposed solution, and how well it meets the success criteria and objectives as described in the problem analysis stage will be discussed.

Table II - Project stages overview

3.1.1 System Architecture

As the aim of this project is to provide proof-of-concept of a working OCR and text classification pipeline for receipts, the project will be delivered as the individual models with working functionality demonstrated on an example receipt. This section will discuss and set out the general project implementation.

This project will be coded in Python 3.8 (specifically Python 3.8.13) [73]. As machine learning is a fundamental aspect of this project, Python is the ideal language to build the project in. A huge library of open-source frameworks and AI tools are available in Python for a range of ML tasks including OCR tasks and NLP tasks.

Training AI models can frequently be hardware intensive, and as such, a GPU (NVIDIA GeForceRTX 3060Ti) will be used to train and test models as part of this project.

Noteworthy installs for this project include: Anaconda [74] – used for Python package management; Tensorflow [75] and Keras [76] which are Python libraries that facilitate the development and training of AI models.

The project will be written in Python in Visual Studio Code [77]. The code will be saved in Jupyter Notebook's notebook format [78] as .ipynb files, allowing the code to easily be ran online using Google Colab [79], ideally with a GPU allocation.

In order to utilise a local GPU to optimise machine learning for neural networks, the NVIDIA CUDA Deep Neural Network library (cuDNN) [80] is a required installation. Once set-up, the library works natively with a range of Python AI frameworks, including those mentioned in the previously (Tensorflow, for example), to enhance neural network training.

3.1.2 Pipeline

The end-to-end pipeline of this project, from receiving a photo of a receipt from a mobile phone to the final output of a visual representation of classified products list, requires various components. Separate experimentation and testing will be performed on each component to find the best combination of processes before being brought together.

To extract data from physical receipts, photos will need to be parsed through a data extraction component, by using a suitable OCR method to extract all the text on the receipt. Then, a list of products and other relevant information must be extracted from the raw OCR output data, which be handled by the KIE component. Once the KIE component has processed the OCR text, it will be passed through the text classification model to assign predicted categories to each product. Finally, the list of categories is used to generate spending statistics in a readable format for the user.

The proposed pipeline to be implemented is shown in Fig. 29. below.

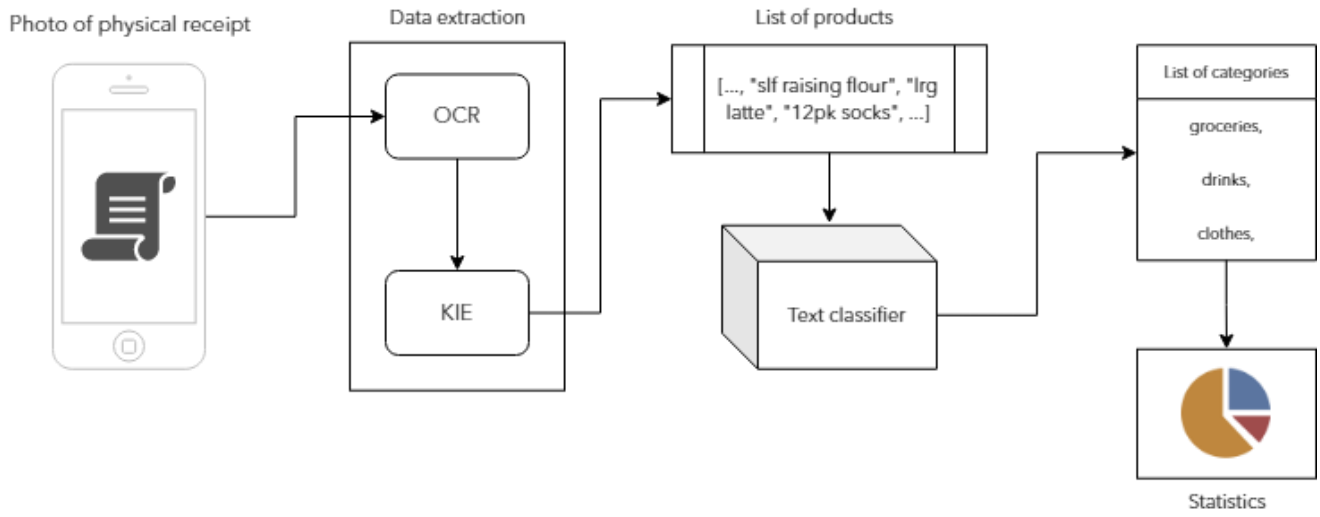


Fig. 29 - Proposed end-to-end pipeline

Working our way through the designed pipeline, each component will operate independently and perform its own transformation on the data. The individual designs of each component will be discussed further in its respective section.

3.2 Dataset

For any ML and AI development, relevant data is essential to the success of the project. For the development of the components specified in the previous section, suitable datasets are required for data extraction and text classification. This section discusses the process of finding appropriate datasets to use for the training and testing of OCR and KIE tools, as well as the text classification component for categorisation of the extracted items list. Pre-existing datasets will be discussed as well as the formation of a custom dataset tailored for this project.

3.2.1 Pre-existing datasets (SROIE and CORD)

For the task of data extraction, there exists two similar, relevant, and specific datasets; the SROIE dataset as discussed in Section 2.5.1 and CORD (Consolidated Receipt Dataset) presented by Park et al. [81]. Both datasets are well-known in the OCR community and information extraction space, often used as benchmarks for comparisons, and contain a collection of around 1,000 receipts that include labelled text.

The SROIE dataset provides two different ground-truth annotations (as the ICDAR Challenge was split into 3 tasks [46]). For the first and second task, each image is labelled with locations of text and the text itself, as the task was focused on localizing and recognizing text on receipts. For the third task, the images are labelled with the company name, the date of the receipt, the address, and the total amount of the receipt. However, noticeably, the labels provided do not include the ground-truth of items in the receipts. Whilst we can use the SROIE dataset to test methods of extracting text and KIE for the provided fields, we cannot use it for the specific

purpose of training or fine-tuning an AI to retrieve an itemised list of purchases (beside manually annotating all 1000 images).

The CORD data-set is provided with accompanying JSON (JavaScript Object Notation) files that categorise data into hierarchical groups, such as *'menu'* and '*menu.price'* , and the locations of the text. This is a more suitable dataset for the project domain and can be used with some cleaning to train a model to return an itemised purchase list.

Whilst the two datasets mentioned above can be used for training an OCR model for the first part of this project (specifically CORD to retrieve an item list), neither provide categorisation of items listed in the receipts. Therefore, there exists a need for a different type of dataset for the text classification section of the project.

At first, it seemed plausible to clean and amend the existing CORD dataset to create a list of items and manually categorise them. However, CORD consists of Indonesian receipts only, and comprehension of the bought items became problematic due to a linguistic barrier. For example, items such as *"Nasi Campur Bali"* are hard to categorise without being an Indonesian speaker or understanding the Indonesian language. This effect is compounded when factoring in shortened codes from the receipts (such as *"Bbk"*), which may be hard to decipher even by native speakers. Therefore, manually categorising the items revealed itself to be an impractical and time-consuming endeavour. Additionally, this project will be focusing on UK receipts and products, so classifying non-English items is out-of-scope but may be investigated in the future.

3.2.2 Kaggle receipt dataset

Upon a search of Kaggle (a well-known online platform in the data-science and ML communities [82]), only one relevant receipt dataset was available [83]. Unfortunately, this dataset is unsuitable as the images are not provided with ground-truth data to train an AI model with our specific purpose in mind. The receipts are also not from the UK.

3.2.3 Product Categorisation

Since a suitable dataset of receipt items and their labels does not exist, product categorisation methods will be tested using different datasets to the ones mentioned above. Suitable datasets for this task include Amazon's Amazon-PQA dataset introduced in [84] and available at [85] which includes 8M+ questions from 1M+ products, and the relevant product data. Repurposing this dataset to extract only the product names and the category that the product is listed under would not be difficult nor time-consuming. Another noteworthy dataset is Kaggle user Kashnitsky's "Hierarchical text classification" dataset [86] which is a similar, albeit smaller, Amazon product reviews dataset.

The "Product Classfication and Clustering" dataset [87] available from Kaggle is comprised of 3 sets of e-commerce data as introduced by Akritidis et al. [88]. The most relevant dataset for this project would appear to be the first dataset, the ShopMania dataset, which contains product data across 190 categories that has

been crawled from an online comparison website. The second dataset contains over 35,000 products across 10 categories, and the third dataset takes data from electronic stores, where the categories and products are limited to electronics.

Training a product classification model with the above datasets will not perfectly replicate results for training with receipt data. Although the datasets mentioned provide product names and categories of the products, it does not imitate receipt data and its complexities as mentioned before such as abbreviations for product names and store-specific acronyms. However, the datasets can be used to demonstrate a working text classification model.

3.3.4 Custom dataset

Due to the limitations of the above datasets, it would be beneficial to create a custom dataset of receipts with suitable ground truth data. The benefit of creating a custom dataset is that the dataset will be fit-for-purpose and will not require additional clean-up or amendments unlike using pre-existing datasets. On the contrary, creating a custom dataset can be manually intensive, and creating a good quality custom dataset will require careful consideration. It would be unrealistic to try and create a dataset of a size comparable to Amazon-PQA for example, but too small of a dataset would impact the quality of approximations made by a model. Training on a small dataset can lead to easy overfitting or underfitting of a model and affect the model's ability to abstract on unseen data. However, whilst the size of a training dataset is a considerable factor in a classification model's success, more importantly, the dataset needs to be balanced and contain good quality data that represents the problem well. Althnian et al. [89] show that a limited (or smaller dataset) does not imply that it cannot be used to train robust models.

With this in mind, an appropriate custom dataset for this project would contain a sufficient number of pictures (from a mobile phone) of receipts from different vendors and settings, labelled with the list of items that were purchased. The custom dataset will used to demonstrate the effectiveness of the designed OCR and text classification models on a smaller scale. Whilst it will not be an entirely true indicator of performance on a comprehensive dataset and unseen data, it is a starting point that can be built upon (for example, retraining the model with new data from users in a HITL architecture as discussed in the Literature Review).

To create this custom dataset, the following steps were followed:

- Over the course of the project initialisation, there will be an active effort to collect receipts and invoices from purchases in UK stores for the purpose of creating a training and test data set
- Photos of the collected receipts will be taken using a mobile phone
- Each receipt will be manually labelled with ground-truth data in a JSON file.
- The JSON file will be structured in the following format, with an example as so:

```
{
  "id": "",
  "date": "",
  "storeName": "",
  "itemList": [
    {
      "itemName": "",
      "quantity": 1,
      "category": "",
      "price": 0
    }
  ],
  "discountList": [
    {
      "itemName": "",
      "quantity": 1,
      "category": "",
      "price": -0
    }
  ]
}
```

```
{
  "id": "0012",
  "date": "01/01/1999",
  "storeName": "asda",
  "itemList": [
    {
      "itemName": "white bread slced",
      "quantity": 1,
      "category": "groceries",
      "price": 3.20
    },
    {
      "itemName": "coffe grnls",
      "quantity": 1,
      "category": "drinks",
      "price": 2.87
    }
  ],
  "discountList": [
    {
      "itemName": "whte brd slced",
      "quantity": 1,
      "category": "groceries",
      "price": -0.50
    }
  ]
}
```

Fig. 30 - Custom dataset JSON structure and example

- Similarly, whilst listed separately, discounts will also be treated as a negative “purchase” and assigned a category
- The bounding boxes of the text will be extracted using one of the OCR tools researched
- Each item in the list of purchases will be assigned a category out of 16:
    - groceries
    - drinks
    - snacks
    - eating out
    - alcohol
    - personal hygiene
    - cleaning & laundry
    - electrical
    - homeware
    - clothes & accessories
    - health & pharmacy
    - entertainment
    - nicotine
    - gardening
    - other

Over the course of the project 65 receipts were collected in total – with over 600 items across 30 stores. Examples of collected receipts and their ground truth data is shown below.

Fig. 31 - Example receipt 1 and corresponding JSON data


```
{
  "id": "0065",
  "date": "2022-04-24",
  "storeName": "B&M",
  "itemList": [
    {
      "itemName": "SOREEN L/BOX/5PK W/CHOC",
      "quantity": 2,
      "category": "snacks",
      "price": 2.00
    },
    {
      "itemName": "SIMPLE SOAP 2PK/125G",
      "quantity": 3,
      "category": "personal hygiene",
      "price": 2.67
    },
    {
      "itemName": "PEARS SOAP 100G/4PK",
      "quantity": 1,
      "category": "personal hygiene",
      "price": 2.29
    },
    {
      "itemName": "MIRACLE GRO/50LTR",
      "quantity": 1,
      "category": "gardening",
      "price": 7.49
    }
  ],
  "discountList": []
}
```


Fig. 32 - Example receipt 2 and corresponding JSON data

The receipts collected are of varying quality, with some receipts having almost no folds, text missing, or aberrations, and some photos, such as the one shown in Fig.32, are crinkled and damaged. The photo quality of the receipts is also of varying quality, some photos are slightly out of focus whereas others are clear and of high quality. This was intentional so that the performance of the OCR tool on bad quality photos could be analysed. Personal data such as card numbers and names have been manually blurred out of the photos (this is further discussed in Chapter 7).

3.3 OCR component

In this section, the steps required to extract all text from a photo of a receipt will be discussed, including pre-processing. A suitable and working OCR component is essential for a receipt extraction pipeline, as inaccurate OCR methods could lead to items or numbers on receipts being missed, requiring manual corrections. As there are no direct comparisons of the three open-source OCR tools discussed in Section 2.4, we will analyse and compare results from each tool to determine which is most suitable for the project's use case.

3.3.1 Pre-processing

To analyse the OCR packages, we will be comparing their ability in extract data from the SROIE dataset, which contains 1,000 receipts, as mentioned in Section 3.2.1. Though the receipts have already been cropped, the lighting and contrast (or visibility of characters) of the receipts is not standardised. Therefore, results acquired with no additional pre-processing will be compared with results acquired after minimal pre-processing.

For the custom dataset, the pictures taken from a mobile phone have purposefully been left un-processed so that pre-processing and its effect on OCR performance can be tested. After performing OCR on the raw photos, the photos will be adjusted (first, converted to black and white, and then, have adaptive thresholding performed on the image). A comparison of the OCR tools on the adjusted versions of the pictures, versus the raw photos will be made.

3.3.2 OCR model comparison method

During implementation, TesseractOCR, EasyOCR, and PaddleOCR will be used and tested for performance on the SROIE dataset. The SROIE dataset is accompanied with test scripts that generate the precision, accuracy, and harmonic mean of results against ground truths. The first test script analyses a method's effectiveness to locate text on a receipt and compares bounding box data, and the second test script analyses a method's ability to recognise text on a receipt by analysing the number of correct tokens found. No preparation of the dataset is needed for testing as the test scripts and ground-truth data is already provided.

OCR tool	Dataset	Metrics for comparison
TesseractOCR	SROIE (no pre-processing), SROIE (minimal pre-processing)	Precision, accuracy, harmonic mean of: Bounding boxes Text retrieved
PaddleOCR	SROIE (no pre-processing), SROIE (minimal pre-processing)	Precision, accuracy, harmonic mean of: Bounding boxes Text retrieved
EasyOCR	SROIE (no pre-processing), SROIE (minimal pre-processing)	Precision, accuracy, harmonic mean of: Bounding boxes Text retrieved

Table III - OCR engine experiments

The OCR tool that demonstrates the best performance on SROIE will be used to extract the bounding boxes of text in the custom dataset and used for experimentation with pre-processing techniques.

3.4 KIE component analysis & design

This section of the report will cover the design of the next component in the pipeline: the key information extractor. The raw data taken from the OCR component will need to be processed to determine which information is relevant for the task. Alongside the product list, key fields also include the date and perhaps the name of the store, which would aid in the generation of useful data analytics.

3.4.1 Dataset preparation

For this task, we will use the CORD dataset to demonstrate the effectiveness of LayoutLMv2. CORD consists of 1,000 receipts (much like SROIE) and includes ground truth data of all text found on the receipt image, tagged with annotations and labels (such as *'menu.price'*) and the coordinates of the bounding boxes of the data. The ground-truth data is provided as in JSON format.


```
{
    "words": [
        {
            "quad": {
                "x2": 252,
                "y3": 620,
                "x3": 252,
                "y4": 620,
                "x1": 204,
                "y1": 588,
                "x4": 204,
                "y2": 588
            },
            "is_key": 0,
            "row_id": 2068523,
            "text": "EGG"
        },
        {
            "quad": {
                "x2": 320,
                "y3": 618,
                "x3": 320,
                "y4": 618,
                "x1": 258,
                "y1": 588,
                "x4": 258,
                "y2": 588
            },
            "is_key": 0,
            "row_id": 2068523,
            "text": "TART"
        }
    ],
    "category": "menu.nm",
    "group_id": 4
},
```


Fig. 33 - An example picture of a receipt from the CORD dataset with its accompanying text data

3.4.2 Model architecture

As discussed in Section 2.5.5, LayoutLMv2 is currently the best performing pre-trained KIE model, that outperforms the original LayoutLM, other graph-based models, and grid-based models. Therefore, for this component of the pipeline, LayoutLMv2 will be finetuned on the CORD dataset as an example – to demonstrate the method's precision, recall, and f1 on a full dataset. Then, we will use the custom dataset for fine-tuning and visualise the results to show the performance of LayoutLMv2 for the project's proposed pipeline.

For training, LayoutLMv2 requires the bounding boxes of relevant text on the document, which is provided in CORD, but not the custom dataset. For the custom dataset, the bounding box data will be fetched from the best OCR tool as selected from the previous section.

3.5 Text classification component analysis & design

For the text classification component of the project pipeline, a suitable NLP model is required. In this section, we will discuss exploring different NLP techniques to gauge the effect on outcome metrics when training a text classification model.

On receiving raw text data of the product list obtained from the data extraction components, the text will then be processed in an NLP pipeline as depicted in the figure below.

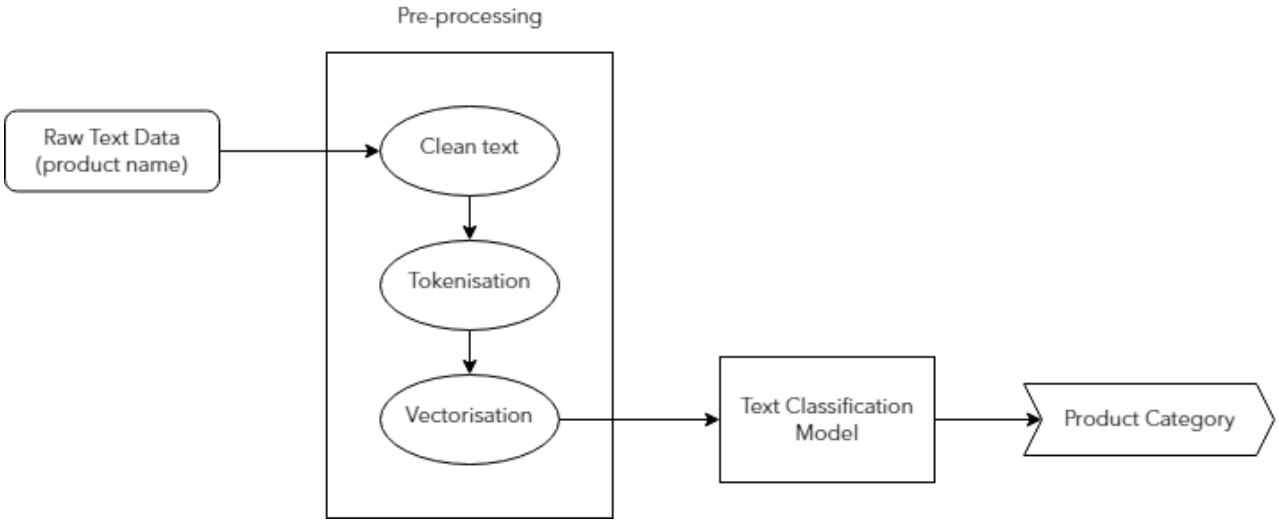


Fig. 34 - A typical text classification pipeline

Typically, an NLP model requires data to go through a few different stages before being processed. For the model to be able to evaluate textual features, the raw text data must be transformed into a machine-readable format. This is known as the pre-processing stage – where text will be cleaned and prepped, separated into tokens, vectorised into its numerical representation, before being processed by the text classifier. There is no industry standard for which models perform best, as each use case is unique. Different techniques for each transformation exist, each with their own benefits and drawbacks depending on the task and the input data.

This section details the experiments that will be conducted during the development phase. Each section will be explored separately, beginning with choosing the best way to clean the text, and then choosing which feature mapping technique best represents the data's features. The outcome from these experiments will then be carried forward and used to explore which AI model can best extrapolate from those feature maps. This incremental style of experimentation allows for full exploration of each component without unnecessary repetition.

3.5.1 Dataset preparation

To test the different text classification techniques, the Amazon-PQA dataset, ShopMania dataset, and our custom dataset will be used for training and validation.

Upon visualising the data within the 2 pre-existing datasets, as shown in Fig. 35 and 36, we can see that all three datasets are unbalanced. The Amazon-PQA dataset has over 10 million entries across 1 million products.

Some of the products are repeated numerous times as the question-answers are from the same product, and some categories have an excessive number of products compared to others.

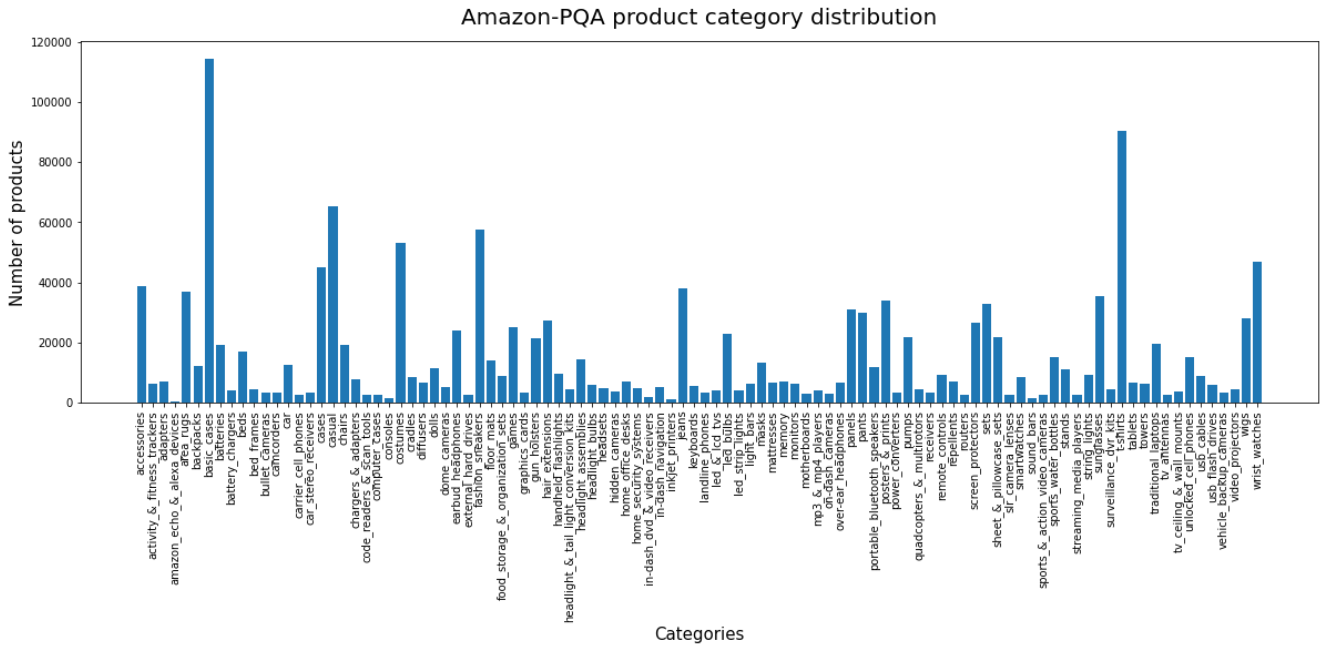


Fig. 35 - Visualisation of Amazon-PQA dataset

Similarly, the ShopMania dataset contains around 300,000 products which are also not distributed evenly across its 190 categories, with the smallest category only containing 1 product. However, the largest categories contain 10,000 entries, compared to Amazon-PQA's largest category of over 100,000 entries.

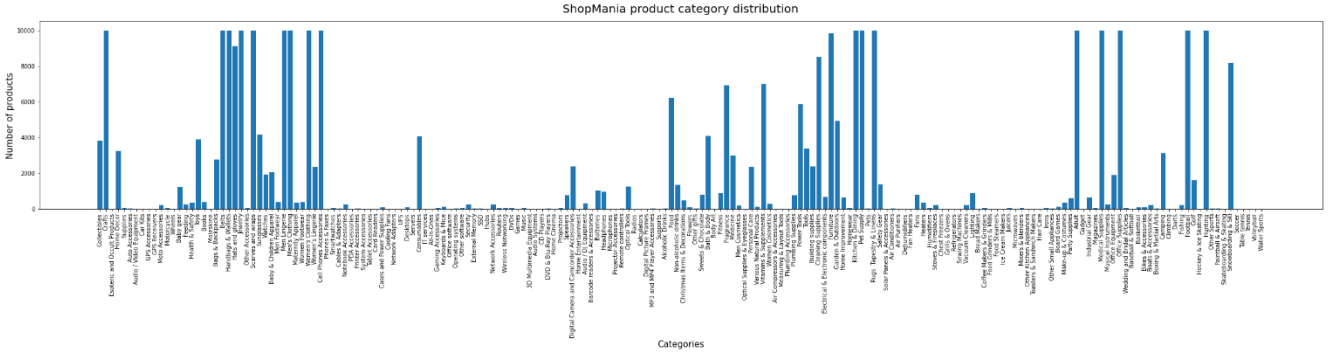


Fig. 36 - Visualisation of ShopMania dataset

The custom data set contains over 500 product items across different categories and will be useful to gauge how the models created during implementation perform against a smaller dataset when compared to the other two larger datasets. How a model treats unseen data is an important evaluation to make.

Across 600 of items found in the custom dataset, a large percentage of those items were classified as *'groceries'*. Of the proposed list of 16 categories, some were not represented more than once or twice such as *'alcohol'*.

As discussed, none of the datasets are balanced and therefore, exploring different ways of balancing the datasets and how this affects results will be an important aspect before implementation and processing of the data. Additionally, due to the size of the two pre-existing datasets, it may be necessary to reduce the number of entries for each category for ease of processing and experimentation. For example, the category *'basic case'* in the Amazon-PQA dataset has over 100,000 examples, which is excessive compared to the smallest category of just over 300.

Three different methods to resolve imbalance affecting datasets are:

- Oversampling – increasing the number of samples of underrepresented classes by creating artificial samples.
- Undersampling – decreasing the number of samples of overrepresented classes by removing some samples from such classes.

For the ShopMania and Amazon-PQA datasets, undersampling would be the most suitable approach as it will also reduce the size of the dataset and require less demanding computation power. However, for the custom dataset it could be beneficial to oversample or explore other options.

3.5.2 Textual pre-processing

Once the datasets have been prepared, the first stage of data preparation is cleaning the text – the logistics of this process will be discussed in this section and the experimentation design will be set out for implementation.

As discussed in Section 2.6.1, before text can be classified by a machine, pre-processing and cleaning must be performed to remove noise and clutter from text data. Whilst some text cleaning techniques are now standard procedure, not all procedures work for every dataset. As such, we will explore different variations of pre-processing techniques to determine the most efficient and effective techniques to represent receipt tokens.

Beside case normalisation and removing punctuation, other optional cleaning techniques as discussed include:

- Removing digits
- Stop word removal
- Stemming
- Lemmatizing

To compare these different methods, we will perform the transformations, and then create a BoW vector and feed the data through a simple logistical regression algorithm. The accuracy scores and time taken will be compared to a baseline of classifying text that has gone through the minimum necessary pre-processing steps (case normalisation and punctuation removal). This will show which text cleaning methods most successfully remove noise and form the best resulting vocabulary for our use case.

Exp. no.	Experiment	Vectorisation method	Model	Metrics
1	Case normalisation & removing punctuation	Bag-of-words	Logistic regression	Accuracy
2	Removing digits	Bag-of-words	Logistic regression	Accuracy
3	Removing stop words	Bag-of-words	Logistic regression	Accuracy
4	Stemming	Bag-of-words	Logistic regression	Accuracy
5	Lemmatizing	Bag-of-words	Logistic regression	Accuracy

Table IV - Text pre-processing experiments

3.5.3 Tokenisation & Vectorisation

After the text has been normalised, it will then be tokenised and transformed into numerical features before being passed to the machine learning model.

The impact of the following vectorisation and tokenisation methods on the 3 datasets will be examined:

- Bag of words (unigrams)
- Bag of bigrams
- TF-IDF (term frequency-inverse document frequency)
- Word embeddings – GloVe, word2vec, and fastText

We will compare the performance of a logistic regression algorithm when trained on each of the above vectorisation methods to decide on the best way to model receipt text data features.

Exp. no.	Experiment	Pre-processing	Model	Metrics
6	Bag of Words	Best from above	Logistic regression	Accuracy
7	Bag of bigrams	Best from above	Logistic regression	Accuracy
8	TF-IDF	Best from above	Logistic regression	Accuracy
9	Word embeddings	Best from above	Logistic regression	Accuracy

Table V - Text vectorisation experiments

3.5.4 Classification models

Once the text has been transformed into a vector representation, we will then feed the vectors into a model to train our classifier.

We will train the following models, as mentioned in Section 2.6.3 and compare them against each other:

- Logistical Regression Classifier (LR)
- Support Vector Machine (SVM)
- Naïve-Bayes Classifier (NB)
- Convoluted Neural Network (TextCNN proposed by Kim [70])
- Recurrent Neural Network
- Long Short-Term Memory network
- Bidirectional LSTM
- BERT

Each model will be finetuned based on its optimal hyperparameters and epochs and the maximum achievable test accuracy on each of the datasets will be used for comparison. Since one of the project's objectives is to reach an 80% accuracy in product classification, models that perform under this threshold will not be considered. Further determining factors will then include training time, prediction time, confusion matrices, and other metrics, to decide on the most appropriate solution for the task.

Exp. no.	Experiment	Pre-processing method	Vectorisation method	Metrics
10	SVM	Best from above	Best from above	Accuracy
11	LR	Best from above	Best from above	Accuracy
12	NB	Best from above	Best from above	Accuracy
13	CNN	Best from above	Best from above	Accuracy, Loss
14	RNN	Best from above	Best from above	Accuracy, Loss
15	LSTM	Best from above	Best from above	Accuracy, Loss
16	Bi- directional LSTM	Best from above	Best from above	Accuracy, Loss
17	BERT	Best from above	Best from above	Accuracy, Loss

Table VI - Classification model experiments

3.5.5 Extra data

An extra consideration to be made is whether the addition of supplementary data from a receipt can be leveraged by a model to achieve higher accuracy. For example, *"pizza"* from a supermarket such as "*Sainsbury's*" could be classed as "*groceries*" but "*pizza*" from a restaurant (i.e., "*Bella Italia*") could be classed as 'eating out'.

For this experiment, the baseline of training a model on product names independently will be compared to the performance of a model trained with an extra input dimension of vendor name. This experiment will only be performed using the custom dataset.

Exp. no.	Experiment	Model	Dataset	Metrics
18	Item name only	Best model from above	Custom dataset only	Accuracy, Loss
19	Item and store name	Best model from above	Custom dataset only	Accuracy, Loss

Table VII - Adding store name experiments

4 OCR Design, Implementation, & Testing

In this chapter, the OCR engines as discussed in Section 3.3 will be implemented and analysed. Each method's efficiency, portability, and ability to distinguish text in receipts will be discussed before deciding which tool will be used in the final pipeline.

4.1 Dataset preparation and pre-processing

The dataset used for this section is the SROIE dataset available from the ICDAR RRC website. The dataset is split into 626 receipts for training and 361 receipts for testing. However, since we will not be training any of the OCR tools, the train and test folders were combined and used to test the OCR capabilities of each tool. Additionally, only the ground truth data for task 1 and task 2 were used as task 3 focused on KIE.

Upon processing the dataset, only 986 receipts were in the correct format (.jpg) with matching ground truth data. Some examples of receipts and a receipt's accompanying ground truth data is shown below.

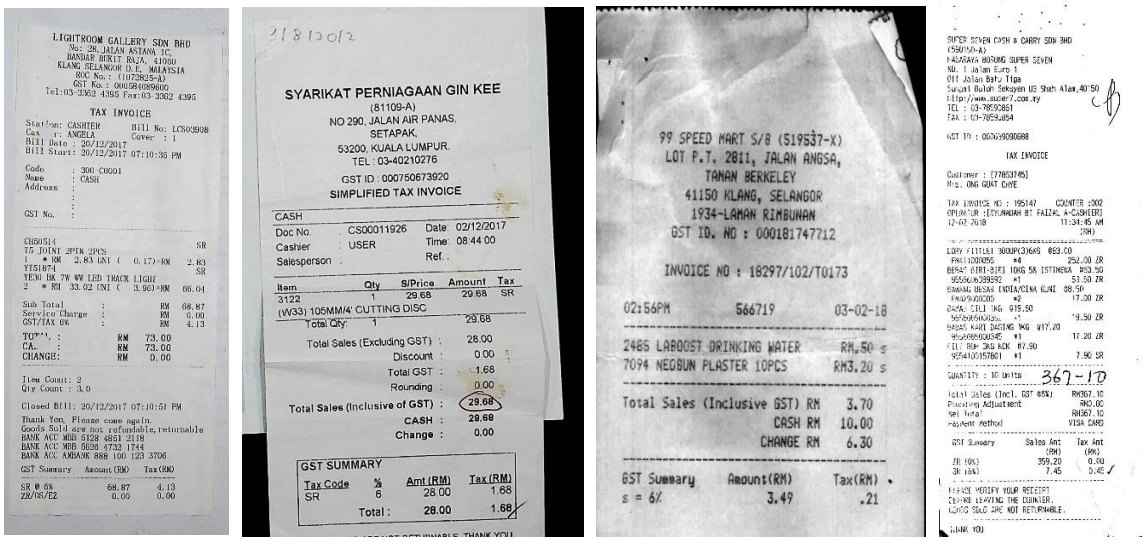


Fig. 37 – Examples of receipts from SROIE

```
173,86,680,86,680,144,173,144,LIGHTROOM GALLERY SDN BHD
236,130,610,130,610,179,236,179,NO: 28, JALAN ASTANA 1C,
226,161,626,161,626,206,226,206,BANDAR BUKIT RAJA, 41050
192,192,657,192,657,241,192,241,KLANG SELANGOR D.E, MALAYSIA
261,229,588,229,588,272,261,272,ROC NO. : (1072825-A)
249,262,600,262,600,302,249,302,GST NO. : 000584089600
148,287,700,287,700,338,148,338,TEL:03-3362 4395 FAX:03-3362 4395
310,363,536,363,536,401,310,401,TAX INVOICE
79,404,354,404,354,444,79,444,STATION: CASHIER
462,421,745,421,745,456,462,456,BILL NO: LCS03908
78,438,130,438,130,467,78,467,CAS
180,446,332,446,332,476,180,476,R: ANGELA
461,455,625,455,625,489,461,489,COVER : 1
78,470,448,470,448,513,78,513,BILL DATE : 20/12/2017
78,504,648,504,648,553,78,553,BILL START: 20/12/2017 07:10:35 PM
76,569,149,569,149,601,76,601,CODE
245,579,428,579,428,609,245,609,: 300-C0001
75,601,146,601,146,633,75,633,NAME
244,610,349,610,349,641,244,641,: CASH
73,636,197,636,197,668,73,668,ADDRESS
```

Fig. 38 - Example of a receipt's ground truth data from SROIE

The ground truth files contain any text from the receipt and the x and y coordinates of the bounding boxes for the text in the following format:

x1_1, y1_1, x2_1, y2_1, x3_1, y3_1, x4_1, y4_1, transcript_1

x1_2, y1_2, x2_2, y2_2, x3_2, y3_2, x4_2, y4_2, transcript_2

x1_3, y1_3, x2_3, y2_3, x3_3, y3_3, x4_3, y4_3, transcript_3

…

As mentioned, scripts for evaluation are also provided. For the localisation task, the produced bounding boxes of text blocks are compared to the ground truth bounding boxes. Text can be located at different levels (character vs text line vs words) and the evaluation script returns the mean average precision and recall. For the second task of recognition, the tokens retrieved from OCR is used to compare to the ground truth tokens, with no location data required. The precision and recall as with task 1 are returned. For each method, the test scripts require the outputs of each file's OCR as a text file that can then be evaluated.

As the photos are already cropped and not perspective-skewed so we do not have to perform these transformations on the dataset. For the experiments on how pre-processing affects the performance of the OCR, the following thresholding is performed to make the receipt black and white.

```
file_name = 'Datasets\SROIE2019\images\X00016469612.jpg'
img = cv2.imread(file_name)
✓ 0.7s

bw = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

thresh = cv2.adaptiveThreshold(bw,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,15)
✓ 0.2s
```

Fig. 39 - Methods to pre-process images

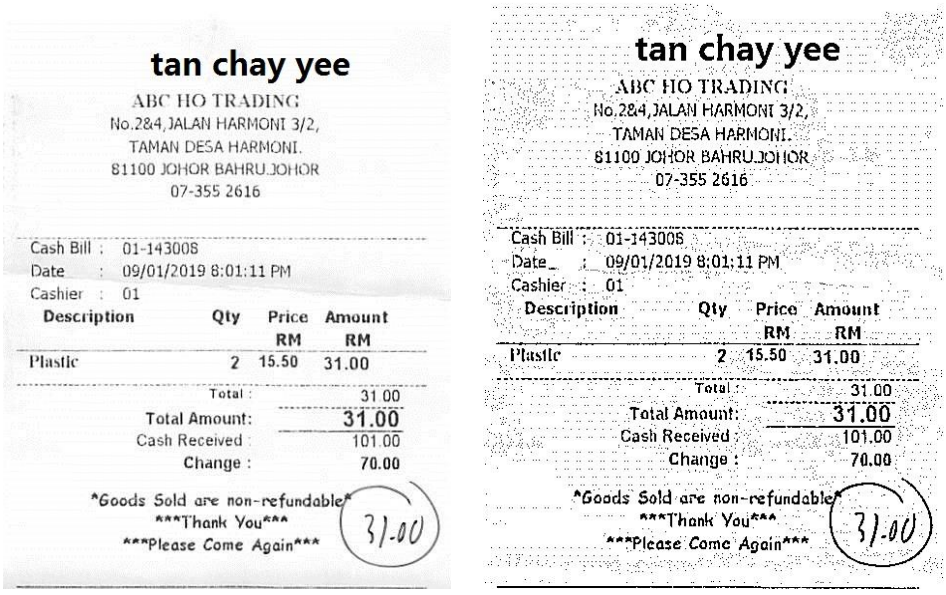


Fig. 40 - Example of a SROIE receipt before and after thresholding is applied

4.2 TesseractOCR

TesseractOCR is available as an API and is buildable using C++. For this project, TesseractOCR is implemented using pytesseract [90] which is a python wrapper for the TesseractOCR engine.

For the localisation test, pytesseract's image_to_data method returns words that are found with the corresponding width, height, x, y coordinates and other data (such as line number and page number).

	level	page_num	block_num	par_num	line_num	word_num	left	top	width	height	conf	text
4	5	1	1	1	1	1	78	52	52	23	93.060196	tan
5	5	1	1	1	1	2	141	57	91	18	91.850983	woon
6	5	1	1	1	1	3	244	57	78	26	92.627731	yann
10	5	1	2	1	1	1	113	168	43	16	90.490517	INDAH
11	5	1	2	1	1	2	168	168	35	15	92.356400	GIFT
...	...	...	...	...	...	...	...	...	...	...	...	...
131	5	1	11	1	3	2	142	943	16	14	95.330765	In
132	5	1	11	1	3	3	169	942	83	15	91.273285	Wholesale
133	5	1	11	1	3	4	263	943	26	14	89.947159	And
134	5	1	11	1	3	5	300	942	7	13	29.558960	f
138	5	1	12	1	1	1	291	917	67	43	95.000000	

Fig. 41 - Example of TesseractOCR output data

To format the OCR results from pytesseract into the format required by the SROIE evaluation scripts, height, width, coordinate data, and the found text is extracted from the result dataframe. The method created for this is shown below.

```python
img_dir = 'Datasets/SROIE2019/images'
result_dir = 'Datasets/SROIE2019/tesseract'
images = [os.path.splitext(filename)[0] for filename in os.listdir(img_dir)]


for idx, image in enumerate(images):
    if idx % 50 == 0:
        print(idx+1,'/',len(images))
    img_path = os.path.join(img_dir, image + '.jpg')
    result1_path = os.path.join(result_dir, 'task 1', image + '.txt')
    result2_path = os.path.join(result_dir, 'task 2', image + '.txt')

    img = cv2.imread(img_path)
    h, w, _ = img.shape

    d = pytesseract.image_to_data(img, output_type=Output.DATAFRAME)
    boxes = []

    d = d[d.conf != -1]

    for i, row in d.iterrows():
        x1 = row['left']
        y1 = row['top']
        x2 = row['left']+row['width']
        y2 = y1
        x3 = x2
        y3 = row['top']+row['height']
        x4 = x1
        y4 = y3

        boxes.append([x1,y1,x2,y2,x3,y3,x4,y4,row['text']])

    task1out(result1_path, boxes)
    task2out(result2_path, boxes)
```

Task 1 method

```python
def task1out(fname, result):
    formatted_output = []
    for line in result:
        formatted_output.append([int(n) for n in line[:8]])

    with open(fname, 'w') as f:
        for row in formatted_output:
            f.write(','.join(str(v).upper() for v in row) + '\n')
```

Task 2 method

```python
def task2out(fname, result):
    formatted_output = []

    for line in result:
        out = [str(line[8]).upper()]
        formatted_output.append(out)

    with open(fname, 'w') as f:
        for row in formatted_output:
            for token in row:
                f.write(token+'\n')
```

Fig. 42 - Methods to parse SROIE dataset through TesseractOCR

The bounding boxes from pytesseract can be used to show the bounding boxes on the original image through cv2's *rectangle* and *imshow* methods. The method and an example receipt with bounding boxes drawn on is shown below.

```
for i, row in d.iterrows():
    x1 = row['left']
    y1 = row['top']
    x2 = row['left']+row['width']
    y2 = y1
    x3 = x2
    y3 = row['top']+row['height']
    x4 = x1
    y4 = y3

    boxes.append([x1,y1,x2,y2,x3,y3,x4,y4,row['text']])
    (x, y, w, h) = (row['left'], row['top'], row['width'], row['height'])
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

print(boxes[0])
cv2.imshow('img', img)
```

Fig. 43 - Method to show bounding boxes on receipts

Fig. 44.a, 44.b, 44.c - Examples of visualising the bounding boxes of text that TesseractOCR was able to localise

As shown, some of Tesseract's OCR predictions do not locate text properly. It also detects things such as the barcode in Fig. 44.b. It also completely misses the text in the price column of the receipt in Fig. 44.c.

4.3 EasyOCR

EasyOCR is available as a python package that requires pytorch and CUDA (however it can also be used with a CPU, though it will be slower to process images).

The complete end-to-end OCR engine is callable through the Reader class which can be passed different languages (for example, English and Chinese). Unlike TesseractOCR, the detected text is returned from the *readtext()* method with full bounding box coordinates as default along with a confidence score as an array. One line of example output is provided below along with the methods for formatting the results.

```
([[18, 20], [136, 20], [136, 52], [18, 52]], '3180203', 0.8713365861964874)
```

Fig. 45 - Example EasyOCR output showing bounding box co-ordinates, text found, and confidence rating

Task 1 method

```
def task1out(fname, result):
    formatted_output = []
    for line in result:
        out = flatten(line[0])
        formatted_output.append([int(n) for n in out])

    with open(fname, 'w') as f:
        for row in formatted_output:
            f.write(','.join(str(v).upper() for v in row) + '\n')
```

Task 2 method

```
def task2out(fname, result):
    formatted_output = []

    for line in result:
        out = []
        for token in line[1].upper().split(' '):
            out.append(token)
        formatted_output.append(out)

    with open(fname, 'w') as f:
        for row in formatted_output:
            for token in row:
                f.write(token+'\n')
```

```
for idx, image in enumerate(images):
    if idx % 50 == 0:
        print(idx+1,'/',len(images))
    img_path = os.path.join(img_dir, image + '.jpg')
    result1_path = os.path.join(result_dir, 'task 1', image + '.txt')
    result2_path = os.path.join(result_dir, 'task 2', image + '.txt')
    result = reader.readtext(img_path)

    task1out(result1_path, result)
    task2out(result2_path, result)
```

Fig. 46 - Methods to parse SROIE through EasyOCR

As with Tesseract, we can use the bounding box data to plot the detected text using the cv2 and matplotlib libraries.

```
import cv2
import matplotlib.pyplot as plt

image = 'X510056849111'
img_path = os.path.join(img_dir, image + '.jpg')
result = reader.readtext(img_path)

print(result[0])

img = cv2.imread(img_path)
spacer = 100
font = cv2.FONT_HERSHEY_SIMPLEX

for detection in result:
    top_left = tuple(detection[0][0])
    bottom_right = tuple(detection[0][2])
    text = detection[1]
    img = cv2.rectangle(img,top_left,bottom_right,(0,255,0),3)
    spacer+=15


plt.figure(figsize=(10,10))
plt.imshow(img)
plt.show()
```

Fig. 47 - Method to plot text localised with EasyOCR

Three examples of receipts and text parsed from EasyOCR with the bounding boxes drawn on are shown below.

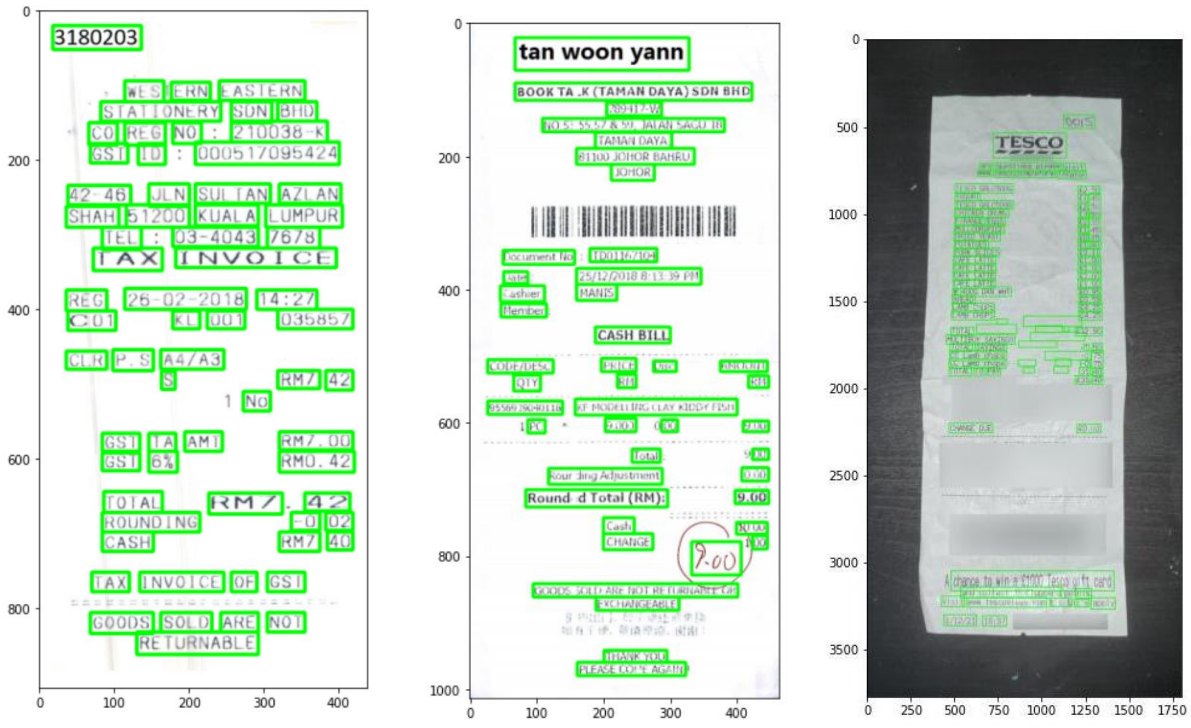


Fig. 48.a, 48.b, 48.c - Examples of visualising the bounding boxes of text that EasyOCR was able to localise

EasyOCR seems to localise text more accurately than TesseractOCR and is not confused by other lines on the page (such as the barcode. However, it does miss some smaller text such as the quantities in the third receipt.

4.4 PaddleOCR

Similar to EasyOCR, PaddleOCR is available as a python package and requires pytorch and CUDA. Another similarity of PaddleOCR to EasyOCR is the format of the returned text. The results from OCR are returned with full bounding box data and a confidence score as an array. The end-to-end OCR tool provided by PaddleOCR is in the form of its *PaddleOCR* class:

```
reader = PaddleOCR(lang='en')
```

Fig. 49 - Method to fetch PaddleOCR class

The method for formatting the results for the evaluation script is similar to the method used for the EasyOCR implementation, though the code used for visualising the bounding boxes on receipts is different as PaddleOCR provides its own *drawocr()* method.

```
from PIL import Image
from paddleocr import draw_ocr

image = 'X510056849111'
img_path = os.path.join(img_dir, image + '.jpg')

result = reader.ocr(img_path, cls=False)

image = Image.open(img_path).convert('RGB')

boxes = [line[0] for line in result]
txts = [line[1][0] for line in result]
scores = [line[1][1] for line in result]
im_show = draw_ocr(image, boxes, txts, scores)

im_show = Image.fromarray(im_show)
im_show.save('result.jpg')
```

Fig. 50 - Method to visualise bounding boxes on receipts

Examples of the text found by PaddleOCR is shown below.

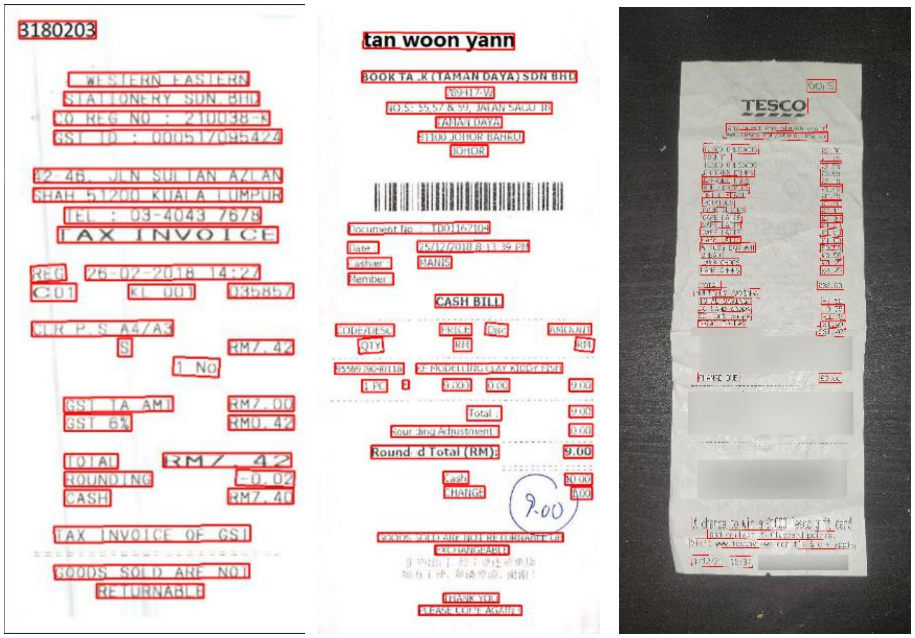


Fig. 51.a, 51.b, 51.c - Examples of visualising the bounding boxes of text that PaddleOCR was able to localise

Visually, the PaddleOCR engine appears to localise text the most successfully, ignoring irrelevant black lines such as the bare codes, and recognising the small quantity numbers that EasyOCR missed.

4.5 Results & Evaluation

The results of all 3 OCR tools across the two tasks of localisation and recognition are summarised in the table below.

OCR engine		Results					
		Task 1 - Localisation			Task 2 - Recognition		
		Precision	Recall	Hmean	Precision	Recall	Hmean
TesseractOCR	No processing	0.115	0.198	0.146	0.678	0.697	0.687
	Processed	0.182	0.215	0.197	0.503	0.500	0.540
EasyOCR	No processing	0.597	0.607	0.602	0.597	0.607	0.602
	Processed	0.701	0.698	0.700	0.652	0.534	0.540
PaddleOCR	No processing	0.938	0.920	0.929	0.804	0.689	0.742
	Processed	0.940	0.931	0.936	0.817	0.710	0.759

Table VIII - Results of OCR experiments

From the results, it can be derived that PaddleOCR performed the best overall at both localising and recognising text, although EasyOCR also performs relatively well. Other deciding factors include the time taken for the engine to process almost 1,000 receipts; in this aspect, PaddleOCR also performed better than Tesseract and EasyOCR. Additionally, processing the receipts by applying thresholding had an overall beneficial effect on OCR results and will be implemented in the final pipeline.

5 KIE Design, Implementation and Testing

This chapter of the report documents the implementation attempt of the chosen KIE tool, LayoutLMv2, as discussed in Section 3.4. The dataset used to fine-tune and test the implementation of LayoutLMv2 is the CORD dataset of 1,000 receipt images.

5.1 Dataset preparation

LayoutLMv2 requires 3 inputs for training. An array of the tokens must be passed as an input, along with the normalised bounding boxes of where the tokens are located on the image of the document, and the label of the token (for example, *"menu.price"* from the CORD categories). Following Github user NielsRogge's tutorial [91], the CORD dataset can be prepared for input into LayoutLMv2 by first normalising the bounding boxes, and retrieving the text and label information from the nested JSON.

```python
import os
import json

def normalize_bbox(bbox, width, height):
    bbox[0] = int(1000*(bbox[0] / width))
    bbox[1] = int(1000*(bbox[1] / height))
    bbox[2] = int(1000*(bbox[2] / width))
    bbox[3] = int(1000*(bbox[3] / height))

    return bbox
```

Fig. 52 - Method to normalise the bounding box co-ordinates

```python
def generate_annotations(path: str):
    files = []

    for js in os.listdir(path):
        with open(path+js) as f:
            files.append(json.load(f))

    words = []
    boxes = []
    labels = []

    for js in files:
        text = []
        box = []
        label = []

        width, height = js['meta']['image_size']['width'], js['meta']['image_size']['height']

        for element in js['valid_line']:
            for word in element['words']:
                txt = word['text']

                x1 = word['quad']['x1']
                y1 = word['quad']['y1']
                x3 = word['quad']['x3']
                y3 = word['quad']['y3']

                bbox = [x1, y1, x3, y3]
                bbox = normalize_bbox(bbox, width, height)

                if len(txt) < 1:
                    continue
                if min(bbox) < 0 or max(bbox) > 1000:
                    continue
                if ((bbox[3] - bbox[1]) < 0) or ((bbox[2] - bbox[0]) < 0):
                    continue

                text.append(txt)
                box.append(bbox)
                label.append(element['category'])

        words.append(text)
        boxes.append(box)
        labels.append(label)

    return words, boxes, labels
```

Fig. 53 - Method to restructure CORD json data

Additionally, some of the labels had very few entries. To solve this problem, the underrepresented labels were replaced with a label named *'other'*.

```
replacing_labels = {'menu.etc': 'other', 'mneu.itemsubtotal': 'other', 'menu.sub_etc': 'other', 'menu.sub_unitprice': 'other', 'menu.vatyn': 'other',
                    'void_menu.nm': 'other', 'void_menu.price': 'other', 'sub_total.othersvc_price': 'other'}
```

Fig. 54 - Dictionary of labels to replace

CORD is already split into training, validation, and testing datasets, so once the inputs as generated from the above methods, they can be saved as *.pickle* files to be loaded later.

```
train_path='Datasets/CORD/train/json/'
val_path = 'Datasets/CORD/dev/json/'
test_path = 'Datasets/CORD/test/json/'

words_train, boxes_train, labels_train = generate_annotations(train_path)
words_val, boxes_val, labels_val = generate_annotations(val_path)
words_test, boxes_test, labels_test = generate_annotations(test_path)
```

```
import pickle
with open('CORD_train.pkl', 'wb') as t:
    pickle.dump([words_train, labels_train, boxes_train], t)
with open('CORD_dev.pkl', 'wb') as t:
    pickle.dump([words_val, labels_val, boxes_val], t)
with open('CORD_test.pkl', 'wb') as t:
    pickle.dump([words_test, labels_test, boxes_test], t)
```

Fig. 55 - Saving prepared CORD inputs for LayoutLMv2

5.2 Finetuning LayoutLMv2

We will be finetuning LayoutLMv2 using the model from HuggingFace's transformers library which provides the pretrained LayoutLMv2 model as an API [60]. Using LayoutLMv2 requires the Facebook detectron2 package [92]. As detectron2 relies on a specific version of CUDA and torch, it was easiest to implement the finetuning of LayoutLM2 using Google Colab, as installing detectron2 on a local machine caused dependecy issues and could not be achieved.

The pretrained LayoutLMv2 processor can be loaded by calling the LayoutLMv2Processor class from the transformer library.

```
from transformers import LayoutLMv2Processor

processor = LayoutLMv2Processor.from_pretrained("microsoft/layoutlmv2-base-uncased", revision="no_ocr")
```

Fig. 56 - Fetching pre-trained LayoutLMv2 model

The processor prepares inputs for training LayoutLMv2, including the text, the labels, and the images.

```
# first, take an image
item = self.image_file_names[idx]
image = Image.open(self.image_dir + item).convert("RGB")

# get word-level annotations
words = self.words[idx]
boxes = self.boxes[idx]
word_labels = self.labels[idx]

assert len(words) == len(boxes) == len(word_labels)

word_labels = [label2id[label] for label in word_labels]
# use processor to prepare everything
encoded_inputs = self.processor(image, words, boxes=boxes, word_labels=word_labels,
                                padding="max_length", truncation=True,
                                return_tensors="pt")
```

Fig. 57 - Preparing CORD images and data for LayoutLMv2

Once the data has been prepared, the pretrained LayoutLMv2 model can be finetuned as follows.

```
from transformers import LayoutLMv2ForTokenClassification, AdamW
import torch

model = LayoutLMv2ForTokenClassification.from_pretrained('microsoft/layoutlmv2-base-uncased',
                                                         num_labels=len(labels))

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
optimizer = AdamW(model.parameters(), lr=5e-5)

global_step = 0
num_train_epochs = 4

#put the model in training mode
model.train()
for epoch in range(num_train_epochs):
   print("Epoch:", epoch)
   for batch in tqdm(train_dataloader):
        # get the inputs;
        input_ids = batch['input_ids'].to(device)
        bbox = batch['bbox'].to(device)
        image = batch['image'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        token_type_ids = batch['token_type_ids'].to(device)
        labels = batch['labels'].to(device)

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = model(input_ids=input_ids,
                        bbox=bbox,
                        image=image,
                        attention_mask=attention_mask,
                        token_type_ids=token_type_ids,
                        labels=labels)
        loss = outputs.loss

        # print loss every 100 steps
        if global_step % 100 == 0:
          print(f"Loss after {global_step} steps: {loss.item()}")

        loss.backward()
        optimizer.step()
        global_step += 1
```

Fig. 58 - Method to finetune LayoutLMv2 on CORD

5.3 Implementation results

The LayoutLMv2 model that has been finetuned on CORD training data can then be used to predict labels on test data.

The following results were achieved using LayoutLMv2.

```
Overall results: {'precision': 0.9336349924585219, 'recall': 0.9385898407884761, 'f1': 0.9361058601134216}
                        precision    recall  f1-score   support

               enu.cnt       0.96      1.00      0.98       224
      enu.discountprice       0.78      0.70      0.74        10
       enu.itemsubtotal       0.00      0.00      0.00         6
                enu.nm        0.94      0.98      0.96       251
               enu.num        0.83      0.91      0.87        11
             enu.price        0.96      0.98      0.97       247
           enu.sub_cnt        1.00      0.59      0.74        17
            enu.sub_nm        0.77      0.75      0.76        32
         enu.sub_price        0.83      0.95      0.88        20
         enu.unitprice        0.97      0.97      0.97        68
        otal.cashprice        0.93      0.92      0.92        71
      otal.changeprice        0.97      0.97      0.97        59
   otal.creditcardprice       0.75      0.71      0.73        17
       otal.emoneyprice       0.50      1.00      0.67         2
       otal.menuqty_cnt       0.77      0.79      0.78        29
      otal.menutype_cnt       0.00      0.00      0.00         7
        otal.total_etc        0.00      0.00      0.00         4
      otal.total_price        0.95      0.94      0.95       101
ub_total.discount_price       0.88      1.00      0.93         7
           ub_total.etc       0.88      0.88      0.88         8
 ub_total.service_price       1.00      1.00      1.00        12
ub_total.subtotal_price       0.92      0.99      0.95        69
     ub_total.tax_price       0.96      0.91      0.93        47

             micro avg       0.93      0.94      0.94      1319
             macro avg       0.76      0.78      0.76      1319
          weighted avg       0.92      0.94      0.93      1319
```

Fig. 59 - Finetuned LayoutLMv2 training validation metrics

The model appears to work relatively well, achieving high scores across precision, recall, and f1 scores. It seemed to have trouble predicting the *"menu.itemsubtotal"* , the *"total.menutype_cnt"*, and the *"total.total_etc"* categories. This may be due to lack of training data. For the more populous categories such as name, discount price, and price, the model performed well.

For our proposed pipeline and use case, the only categories for LayoutLMv2 to predict will be item name, prices, item price, store name, date, and discount price. As these are present on almost all receipts, LayoutLMv2 should perform well on the custom dataset.

6 Text Classification Design, Implementation, and Testing

This chapter of reports on the experiments of the designed models and the test results from experimentation for the text classification component. It will cover how implementation was achieved, whether any obstacles were happened upon, if and how they were overcome, and justification behind any adjustments that were made. An evaluation of the resulting implementation will also be conducted.

6.1 Development & Testing

As outlined in Section 3.5, the text classification pipeline will be implemented in the following order: dataset preparation, text pre-processing (cleaning), vectorisation, and choosing the text classification model. The code for implementation will be explained as well the results from the experiments.

6.2 Dataset preparation

The 3 datasets for this section are made available in different formats. The Amazon-PQA dataset is available via Amazon's AWS (Amazon's Web Services) client in a .tar file. Once the dataset is downloaded it can be extracted into the file structure as below containing a .json file for each category.

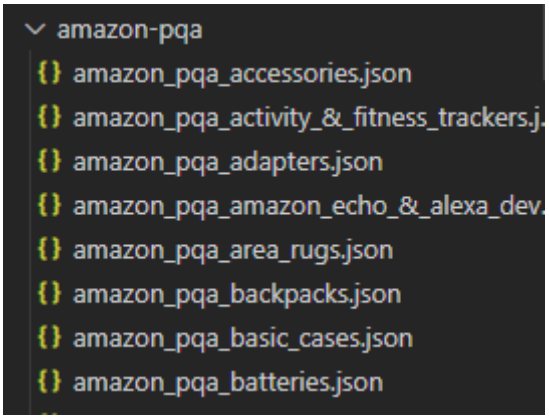


Fig. 60 - Structure of Amazon-PQA dataset

Each JSON file contains all the question-answer entries as a JSON object containing 10 fields including fields such as **question_text** and **brand_name** which are irrelevant to this project. The fields that are relevant for our use case are the **asin_id** (the product identifier) and the **item_name**. To extract this data, each category's JSON code is parsed and the **asin** and **item_name** fields are selected and placed into an array attached with the category name (taken from the JSON file name), resulting in a pandas DataFrame object as below.

	asin	item_name	category
9376925	B071RNCFQF	BlackBerry KEYone 32GB BBB100-2 - 4.5 Inch Fa...	unlocked_cell_phones
2576710	B07Y1HQ6N5	ABALAGU Men's Deluxe Santa Suit 10pc. Christma...	costumes

Fig. 61 - Example of a product from the Amazon-PQA dataset

As mentioned in Section 3.2.3, there are duplicates of products due to the nature of the database. Before removing copies, there are almost 10 million question-answer entries, which equates to around 1.5 million products once we remove duplicates based on the **asin** of entries. Once the set of products is retrieved, the **asin** column of the DataFrame can be dropped as it is no longer needed for the task of text classification, leaving the final prepared Amazon-PQA dataset to look like the below.

	item_name	category
890647	iPhone Xs MAX Magnetic Case,Full-Edge Protecti...	basic_cases
7030343	for MATR X Powerwatch X - Powerwatch 46-50 mm ...	screen_protectors
3331491	Christian Paris Men's Louis Spikes Flat Leathe...	fashion_sneakers

Fig. 62 - Amazon-PQA dataset structure without ASIN

Visualizing the distribution of data across categories as in Fig. 35., there were over 100,000 products in the **basic cases** category whilst the smaller categories had less than 1,000 entries (with the smallest category having 319 products). Additionally, on most machines, processing over 1 million items would be intensive and time-consuming (particularly for a bag of words implementation where each representation of a product could be a matrix of over 50,000 dimensions). Therefore, it was necessary to reduce the size of the dataset, which had the additional advantage of improving the distribution of the dataset.

To evenly reduce the size of the dataset, a random selection of 100 products were chosen from each category to create a dataset of 100,000 products across 100 categories. The distribution is pictured below.

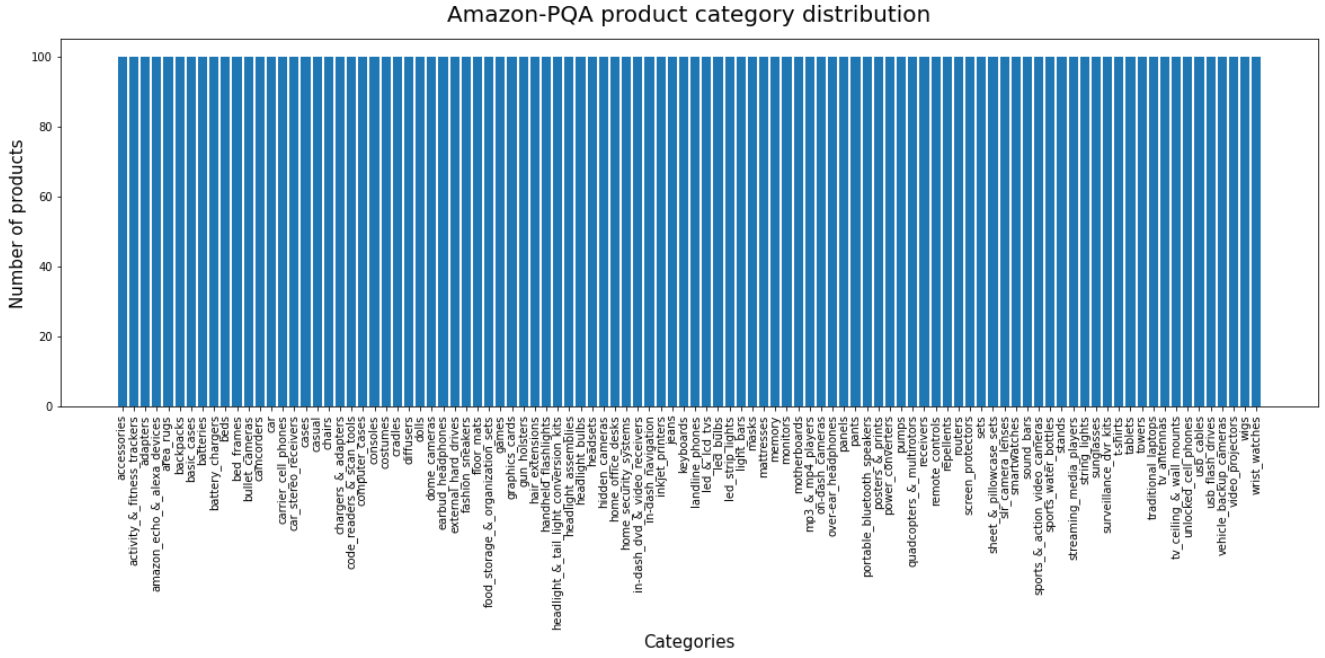


Fig. 63 - Visualising Amazon-PQA dataset after performing random undersampling

The ShopMania dataset is available in .csv format and .xml format. Using pandas' *read_csv()* method, the csv version of the data can be easily read into a DataFrame for processing. Loading the ShopMania dataset required almost no preparation as only 4 fields were present (**item_id**, **item_name**, **category_id**, and **category_name**).

The dataset contains 190 categories and whilst around 50 categories had a significant number of entries (up to 10,000), most categories had less than 100 entries. In total, there were over 300,000 entries, which is still a large amount for processing and experimentation. Similarly to the Amazon-PQA dataset, the larger categories were undersampled by randomly selecting 100 entries from each, and any categories with less than 100 entries were left untouched. The resulting distribution, as shown in Fig. 64., is less evenly distributed compared to the Amazon-PQA dataset but represented a more realistic or closer distribution to our use case (for example, *'groceries'* in our custom dataset has more data than *'clothes'*). The final data set has 11,208 entries for processing, across 190 categories.

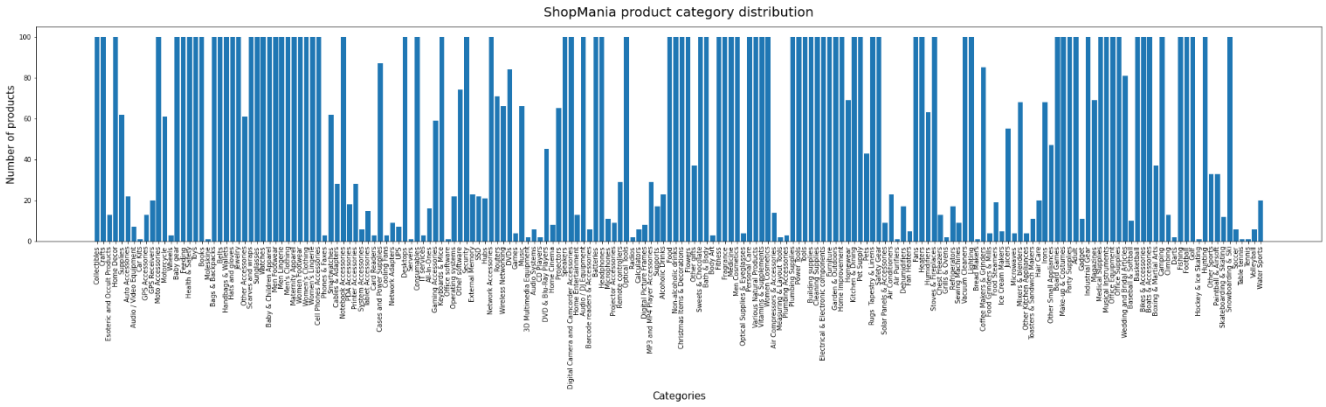


Fig. 64 - Visualisation of ShopMania dataset after random undersampling

Finally, the custom dataset has around 600 entries in total across 10 categories.

6.3 Text pre-processing

This section documents the methods created for the pre-processing experiments performed on the text as listed in Section 3.5.2 (repeated here for the reader's convenience).

Exp. no.	Experiment	Vectorisation method	Model	Metrics
1	Case normalisation & removing punctuation	Bag-of-words	Logistic regression	Accuracy
2	Removing digits	Bag-of-words	Logistic regression	Accuracy
3	Removing stop words	Bag-of-words	Logistic regression	Accuracy
4	Stemming	Bag-of-words	Logistic regression	Accuracy
5	Lemmatizing	Bag-of-words	Logistic regression	Accuracy

6.3.1 Implementation of experiments with pre-processing techniques

After the dataset preparation, the data is processed as a dataframe and can be processed for a text classification model. To experiment with pre-processing techniques, a method was created for each pre-processing step.

For the base implementation of textual data cleaning (Experiment 1), the product names were cleaned by replacing capital letters with lowercase letters and removing punctuation (using the method shown in Fig. 65).

```
def lower(text):
    result = " ".join([word.lower() for word in text.split()])
    return result


def remove_punc(text):
    remove = str.maketrans('', '', (string.punctuation + '£'))
    return text.translate(remove)
```

Fig. 65 - Methods for case normalisation and removing punctuation from a string input

The cleaned text was then used to create a bag of words (with sklearn's *CountVectorizer* class) before being used to train on a logistic regression algorithm (implemented using sklearn's *linear_model* module).

```
def get_vectors(train_data, val_data):
    vectorizer = CountVectorizer()

    train_data = vectorizer.fit_transform(train_data)
    val_data = vectorizer.transform(val_data)

    return train_data, val_data


def get_lr_model(x, y, iter=100):
    lr = LogisticRegression(class_weight='balanced', max_iter=iter, n_jobs=8).fit(x, y)

    return lr
```

Fig. 66 - Methods to create bag of words vectors and train logistic regression algorithm

The other transformations were performed on top of the baseline, before being turned into a bag of words and used for logistic regression. The processes of removing digits and punctuation and case normalisation, were performed using Python's native string module. For example, the custom method to remove numbers:

```
def remove_nums(text):
    remove = str.maketrans('', '', string.digits)
    return text.translate(remove)
```

Fig. 67 - Method to remove numbers from an input string

Removing stop words, stemming words, and lemmatizing words was performed using the nltk package's *PorterStemmer(), WordNetLemmatizer(),* and stopwords list.

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords
```

```
def remove_stopwords(text):
    text = " ".join([word for word in text.split() if word not in stoplist])
    return text


def stemmer_nltk(text):
    stemmed = " ".join(stemmer.stem(word) for word in text.split())
    return stemmed


def lemmatizer_nltk(text):
    lemmatized = " ".join(lemmatizer.lemmatize(word) for word in text.split())
    return lemmatized
```

Fig. 68 - Methods to remove stop words, stem words, and lemmatize words

6.3.2 Text pre-processing experiment results

The results of each transformation across each dataset are summarised below. Improvements on the baseline implementation are highlighted in green whereas any deterioration in accuracy is highlighted in red.

Experiment	Accuracy (%) after 100 iterations of Logistic Regression (3 s.f.)		
	Amazon-PQA	ShopMania	Custom dataset
1) Case normalisation + removing punctuation (baseline)	84.2	78.2	79.5
2) Baseline + Removing digits	84.4	78.7	81.1
3) Baseline + Removing stop words	84.0	78.2	79.5
4) Baseline + Stemming	84.7	79.1	78.0
5) Baseline + Lemmatizing	83.3	69.9	77.3

Table IX - Results of text pre-processing experiments

As shown by the results, each of the techniques effected the model's prediction accuracy by less than ±2%, besides lemmatizing ShopMania's input text, which caused a significant drop of ~10% in accuracy. In fact, lemmatizing across all three datasets seemed to have a negative effect. Conversely, removing numbers had a positive effect on accuracy across all three datasets, suggesting that removing numbers is a necessary step in text cleaning. The other transformations performed differently across the datasets.

Since each of the transformations were performed separately, secondary tests experiments were performed where different combinations of the cleaning techniques that performed relatively well were also tested. This deviates from the initial design of experimentation but was necessary to aid the decision-making process when choosing the final text-cleaning techniques to be performed, as the different combinations had varying effects on the model accuracy, as shown in Table X.

Experiment	Accuracy (%) after 100 iterations of Logistic Regression (3 s.f.)		
	Amazon-PQA	ShopMania	Custom dataset
Case normalisation + removing punctuation + removing numbers (baseline)	84.4	78.7	81.1
Baseline + Removing stop words	84.9	78.9	80.3
Baseline + Stemming	85.1	78.8	79.5
Baseline + Stemming + Removing stop words	84.8	78.9	79.5

Table X - Results of further text pre-processing experiments

From the results above, combining the cleaning techniques was effective in some cases and detrimental in others (particularly for the custom dataset which did not see any improvements). Although further experiments could be conducted, it was decided that this was unnecessary, since the effects of each combination of techniques was within ±2% which is a negligible amount. These tests show that text cleaning techniques and their benefits or negatives are dependent on the dataset. Therefore, for the purpose of the next experiments, the final text cleaning techniques chosen were whichever performed best on each dataset (so Amazon-PQA text was case normalised, stripped of punctuation and numbers, and stemmed, whereas the custom dataset was only case normalised and stripped of punctuation and numbers). The final text pre-processing method for ShopMania is as below.

```
df = datasets['shopmania'][0]

df['item_name'] = df['item_name'].astype(str)
df['item_name'] = df['item_name'].apply(lower)
df['item_name'] = df['item_name'].apply(remove_punc)
df['item_name'] = df['item_name'].apply(remove_nums)
df['item_name'] = df['item_name'].apply(remove_stopwords)
df['item_name'] = df['item_name'].apply(stemmer_nltk)
```

Table XI – Final method to clean ShopMania dataset

6.4 Tokenisation & Vectorisation

After comparing the pre-processing techniques in the section above using bag of words vectorisation, this section experiments with vectorisation and tokenisation techniques. The following experiments (table repeated from Section 3.5.3 for reader convenience) were conducted on the datasets.

Exp. no.	Experiment	Pre-processing	Model	Metrics
6	Bag of Words	Best from above	Logistic regression	Accuracy
7	Bag of bigrams	Best from above	Logistic regression	Accuracy
8	TF-IDF	Best from above	Logistic regression	Accuracy
9	Word embeddings	Best from above	Logistic regression	Accuracy

6.3.1 Implementation of experiments on vectorisation techniques

For experiment 7, the method for creating a bag of bigrams was very similar to creating a bag of words (unigrams). The parameter for *ngram_range* in sklearn's *CountVectorizer* was adjusted to 2 for bigrams.

```
def get_vectors(train_data, val_data):
    vectorizer = CountVectorizer(ngram_range=(2,2))

    train_data = vectorizer.fit_transform(train_data)
    val_data = vectorizer.transform(val_data)

    print(train_data.shape)

    return train_data, val_data
```

Fig. 69 - Method to create bag of n-grams

Representing the texts in TF-IDF form required using sklearn's *TfidfTransformer* on a bag of words to create the feature vectors.

```
from sklearn.feature_extraction.text import TfidfTransformer

def transform_data(train, val):
    vectorizer = CountVectorizer()
    transformer = TfidfTransformer()

    X_train = vectorizer.fit_transform(train) # BoW
    X_train = transformer.fit_transform(X_train) # TF-IDF

    X_val = vectorizer.transform(val)
    X_val = transformer.transform(X_val)

    print(X_train.shape)

    return X_train, X_val
```

Fig. 70 - Method to create TF-IDF vectors

During the design phase of the project, initially the vectorisation comparisons were to be carried out on a logistic regression algorithm, the word embedding implementation with logistic regression was not yielding results consistent with expectations from research. As discussed in Section 2.6.2, word embeddings theoretically perform better than sparse representations, so the word embeddings experiment was reconducted using keras. The neural network only contained one Dense layer of nodes equal to the number of categories of each dataset with a softmax activator which, as discussed in Section 2.3.2, is almost equivalent to a logistic regression model. Allowing the embedding layer to be trained resulted in better results that could be compared with the other vectorisation methods.

The word embeddings were created by fine tuning a pre-trained embedding model with the tokens from the dataset and then creating an embedding matrix from which embeddings could be fetched from.

```python
def train_ft(df):
    cor_path = datapath("lee_background.cor")
    ft = FastText(vector_size=100)

    train_ft = df['item_name'].tolist()

    train_ft = [x.split() for x in train_ft]

    ft.build_vocab(corpus_file=cor_path)

    ft.train(
        corpus_file=cor_path, epochs=ft.epochs,
        total_examples=ft.corpus_count, total_words=ft.corpus_total_words,
    )

    ft.build_vocab(train_ft, update=True)
    ft.train(train_ft, total_examples=len(train_ft), epochs=ft.epochs)

    return ft


def get_ft_matrix(vocab, word_index, ft):
    n_tokens = len(vocab) + 2
    embedding_dim = 100
    hits = 0
    misses = 0

    matrix = np.zeros((n_tokens, embedding_dim))

    for word, i in word_index.items():
        try:
            embedding_vector = ft.wv[word]

            if embedding_vector is not None:
                matrix[i] = embedding_vector
                hits += 1
            else:
                matrix[i] = np.random.randn(embedding_dim)
                misses += 1
        except Exception as e:
            matrix[i] = np.random.randn(embedding_dim)
            print(e)
            misses += 1

    print("Converted %d words (%d misses)" % (hits, misses))

    return matrix, n_tokens
```

Fig. 71 - Loading a fastText embedding initialiser

The embedding matrix is then used in a keras *Embedding* layer to process inputs into 100-dimension word embeddings, before being processed through the rest of the network.

```
inputs = tf.keras.Input(shape=(max_len,))

embedding_layer = l.Embedding(
    n_tokens,
    dim,
    input_length=max_len,
    embeddings_initializer=tf.keras.initializers.Constant(matrix),
    trainable = True
)(inputs)
```

Fig. 72 - Creating an embedding layer that processes vectors into word embeddings

Extra experiments were also implemented to evaluate the difference between GloVe pretrained embeddings, word2vec pretrained embeddings, and fastText pretrained embeddings (as listed in Table []) – techniques which were discussed in Section 2.6.2.

Experiment	Model	Metrics
GloVe	Neural network (1 layer)	Accuracy, Loss
word2vec	Neural network (1 layer)	Accuracy, Loss
fastText	Neural network (1 layer)	Accuracy, Loss

Table XII – Extra word embedding experiments

6.4.2 Tokenisation & vectorisation experiment results

The results from the above experiments are shown below.

Experiment	Accuracy (%)		
	Amazon-PQA	ShopMania	Custom dataset
6) Bag of words (baseline)	84.9	78.9	81.1
7) Bag of bigrams	73.5	70.1	41.6
8) TF-IDF	85.9	75.0	82.3
9) word2vec embeddings	31.4	13.1	21.2
9.5) word2vec embeddings (neural network)	78.3	70.6	84.2

Table XIII - Results from vectorisation experiments

The results show that integrating the order of words into the vectors in a bag of bigrams results in worse accuracy. This suggests that the order or semantic meaning of words in a product name are less important, and that the bag of words or TF-IDF approach may be the best vectorisation method for our purpose. The

exception to this is the implementation of word embeddings, as the plots of accuracy and loss across iterations indicate that the algorithm is underfitting, which could mean that improved results may be seen with a deeper network (or a different model - which is explored in the next section).

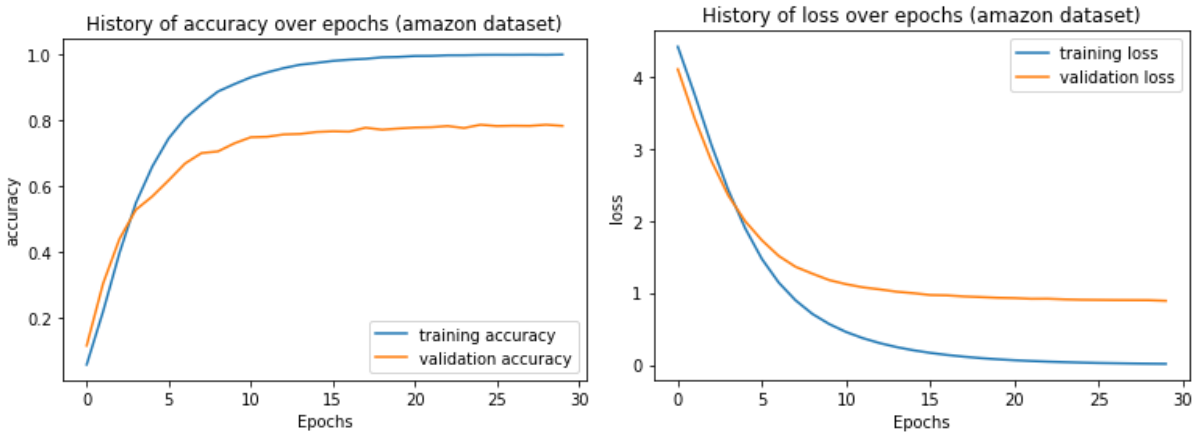


Fig. 73 - History of training and validation accuracy and loss across epochs for model trained on Amazon-PQA dataset

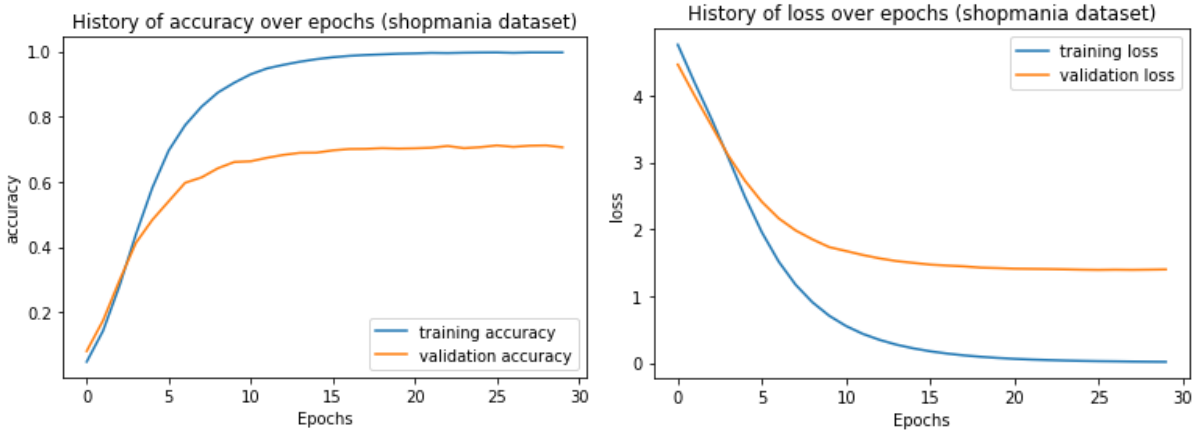


Fig. 74 - History of training and validation accuracy and loss across epochs for model trained on ShopMania dataset

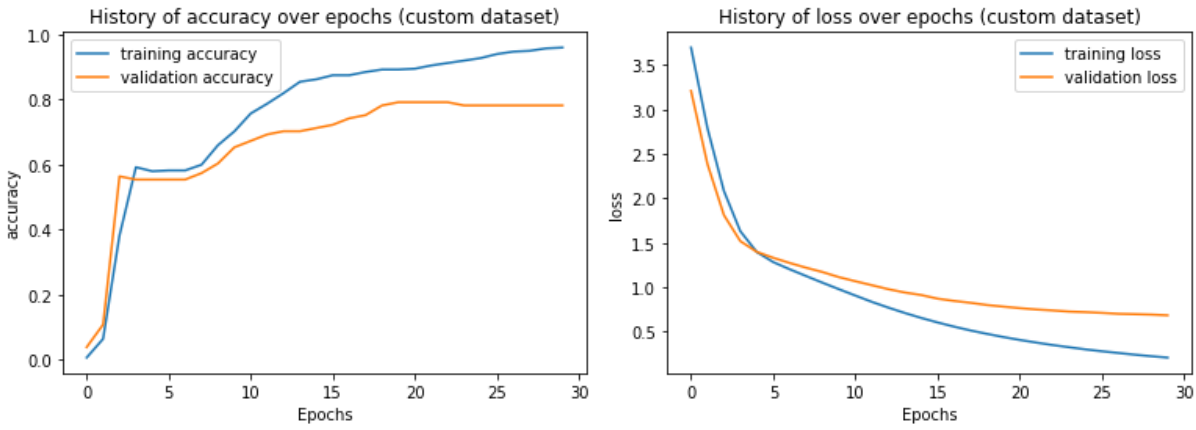


Fig. 75 - History of training and validation accuracy and loss across epochs for model trained on custom dataset

Additionally, the word embedding implementation was set to 30 iterations which was less than the maximum iterations set for the logistic regression method (although the graphs show a plateau at around 10 epochs regardless). Word embeddings also show improved accuracy on the custom dataset.

The results and accuracy-loss graphs of the different word embedding experiments are presented in Table XIV.

Experiment	Metric	Amazon-PQA	ShopMania	Custom dataset
word2vec	Accuracy (%)	75.6	69.4	84.1
	Loss	1.01	1.47	0.70
GloVe	Accuracy (%)	78.7	75.3	81.8
	Loss	0.84	1.15	0.77
fastText	Accuracy (%)	79.6	75.8	79.5
	Loss	0.83	1.16	0.75

Table XIV – Results of word embedding experiments

Across the datasets, all three word embedding methods performed similarly, although fastText embeddings performed the best on the Amazon-PQA and ShopMania datasets.

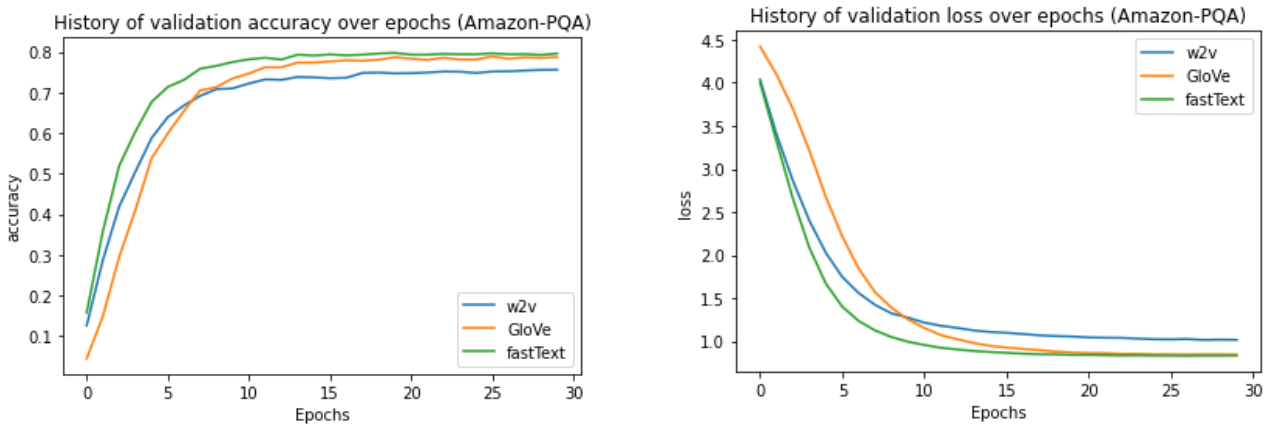


Fig. 76 - History of validation accuracy and loss across epochs for different embedding models trained on Amazon-PQA dataset

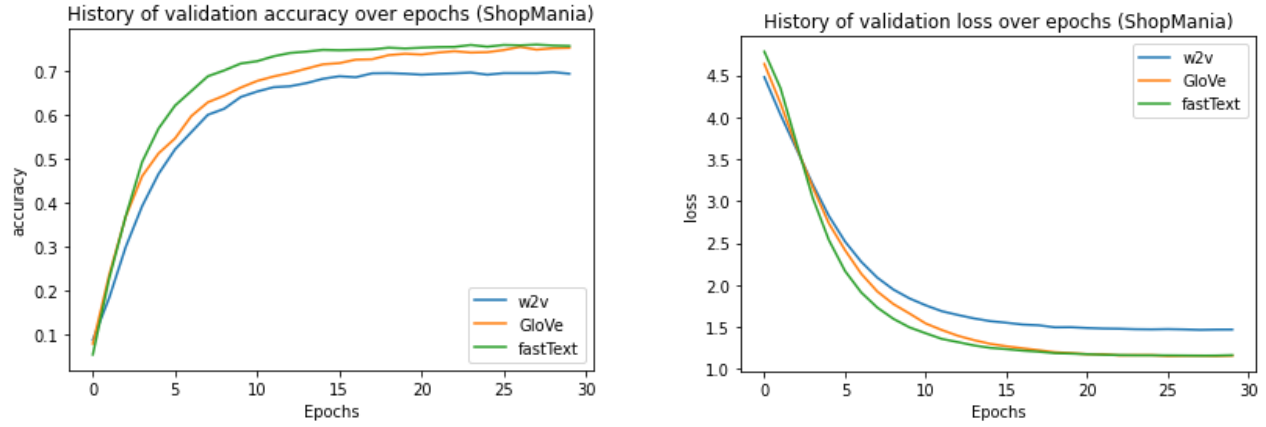


Fig. 77 -History of validation accuracy and loss across epochs for different embedding models trained on ShopMania dataset

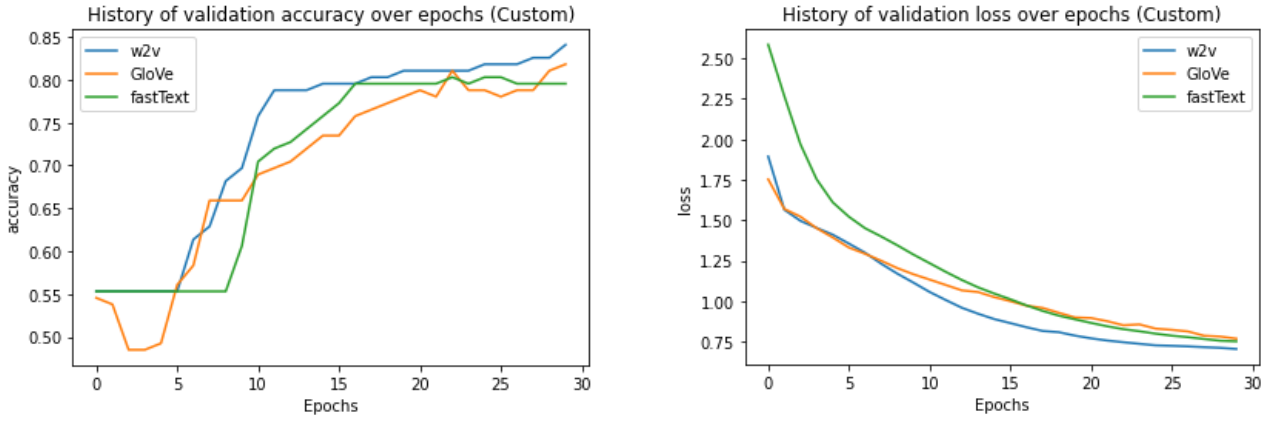


Fig. 78 - History of validation accuracy and loss across epochs for different embedding models trained on custom dataset

The graphs of validation accuracy and loss show that the models trained on the Amazon-PQA dataset plateau the quickest, at around 10 epochs, followed by ShopMania at 15 epochs, whereas the loss graph for the custom dataset indicates more training could result in better accuracies.

Overall, the word2vec model for the custom dataset outperforms the other vectorisation techniques, whilst bag of words performed best on the other two datasets.

6.5 Classification Models

The following model architectures for text classification were implemented and trained using either TF-IDF vectors or fastText embeddings. Due to compatibility of the classic models with embedding vectors, it was not possible to implement the best vectorisation method (which would be fastText or word2vec embeddings) as described in Section 3.5.4. Therefore, the next best vectorisation method was used for the traditional models.

Exp. no.	Experiment	Vectorisation method	Metrics
10	LR	TF-IDF	Accuracy
11	NB	TF-IDF	Accuracy
12	SVM	TF-IDF	Accuracy
13	Neural network	TF-IDF	Accuracy
14	TextCNN	fT embeddings	Accuracy, Loss
15	RNN	fT embeddings	Accuracy, Loss
16	Bi-directional LSTM	fT embeddings	Accuracy, Loss
17	BERT	fT embeddings	Accuracy, Loss

6.5.1 Implementation of experiments with model architectures

The classic models (LR, SVM, and NB) were constructed using sklearn modules (such as *LinearSVC* class for the SVM implementation and *MultinomialNB* for naive Bayes implementation. The simple neural network was implemented using keras and tensorflow packages.

```python
def get_nb_model(x,y):
    nb = MultinomialNB().fit(x, y)
    return nb


def get_svm_model(x,y):
    svm = LinearSVC().fit(x,y)
    return svm
```

```python
from keras.models import Sequential
import keras.layers as l

import tensorflow as tf

def get_nn_model(n_classes):
    m = Sequential()

    m.add(l.Dense(128, activation='relu'))
    m.add(l.Dense(n_classes, activation='softmax'))

    optim = tf.keras.optimizers.Adam()

    m.compile(
    loss="categorical_crossentropy", optimizer=optim, metrics=['accuracy']
    )

    return m
```

Fig. 79 - Training a naive-bayes classifier, a SVM, and creating a shallow neural network

The neural network constructed has 128 hidden neurons and the model using the Amazon-PQA dataset can be visualised as in Fig. 80.

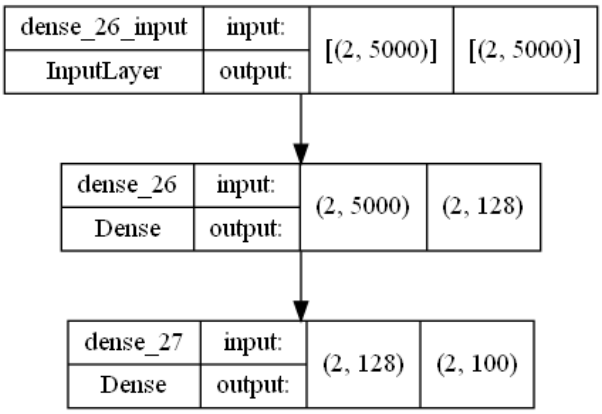


Fig. 80 - Shallow neural network built to classify Amazon-PQA products

The input of 5,000 was set as the maximum number of features for the TF-IDF vectorizer to limit the computation time needed between epochs. Then there is a layer of 128 hidden neurons which is fully connected to the final output layer of 100 neurons (representing each category in the Amazon-PQA dataset).

To model the more complex algorithms, keras and tensorflow libraries were used to create convolutional layers (using *Conv2D* and *MaxPooling2D* layers to replicate the convolutional and pooling layers in a *Sequential* model) for the implementation of Kim's TextCNN [70] implementation.

```
def get_textcnn_model(matrix, n_tokens, max_len, n_classes, dim=100):
    inputs = tf.keras.Input(shape=(max_len,))

    embedding_layer = l.Embedding(
        n_tokens,
        dim,
        input_length=max_len,
        embeddings_initializer=tf.keras.initializers.Constant(matrix),
        trainable = True
    )(inputs)

    reshape = l.Reshape((max_len, 100, 1))(embedding_layer)

    conv_0 = l.Conv2D(100, kernel_size=(3,100), activation='relu')(reshape)
    conv_1 = l.Conv2D(100, kernel_size=(4,100), activation='relu')(reshape)
    conv_2 = l.Conv2D(100, kernel_size=(5,100), activation='relu')(reshape)

    maxpool_0 = l.MaxPooling2D(pool_size=(max_len - 3 + 1, 1))(conv_0)
    maxpool_1 = l.MaxPooling2D(pool_size=(max_len - 4 + 1, 1))(conv_1)
    maxpool_2 = l.MaxPooling2D(pool_size=(max_len - 5 + 1, 1))(conv_2)

    concatenate = l.Concatenate(axis=1)([maxpool_0, maxpool_1, maxpool_2])

    flatten = l.Flatten()(concatenate)

    dropout = l.Dropout(0.5)(flatten)

    output = l.Dense(n_classes, activation='softmax')(dropout)

    embedding_model = tf.keras.Model(inputs=inputs, outputs=output)

    optim = tf.keras.optimizers.Adam()

    embedding_model.compile(
        loss="categorical_crossentropy", optimizer=optim, metrics=['accuracy']
    )

    return embedding_model
```

Fig. 81 - Creating TextCNN implementation

SimpleRNN, *LSTM,* and *Bidirectional* keras layers are used for the implementations of recurrent neural network models.

```python
def get_rnn_model(enc, n_classes):
    model = Sequential([
        enc,
        l.Embedding(
            input_dim=len(enc.get_vocabulary()),
            output_dim=100,
            mask_zero=True
            ),
        l.SimpleRNN(64),
        l.Dense(1, activation='relu'),
        l.Dense(n_classes, activation='softmax')
    ])

    optim = tf.keras.optimizers.Adam()

    model.compile(
        loss="categorical_crossentropy", optimizer=optim, metrics=['accuracy']
    )

    return model
```

```python
def get_bilstm_model(n_tokens, max_len, n_classes, dim=100):
    model = Sequential()

    model.add(l.Embedding(
            n_tokens,
            dim,
            input_length=max_len,
            mask_zero=True
            ))
    model.add(l.Bidirectional(l.LSTM(128, return_sequences=True)))
    model.add(l.Bidirectional(l.LSTM(64)))
    model.add(l.Dense(64, activation='relu'))
    model.add(l.Dense(n_classes, activation='softmax'))

    optim = tf.keras.optimizers.Adam()

    model.compile(
        loss="categorical_crossentropy", optimizer=optim, metrics=['accuracy']
    )

    return model
```

Fig. 82 - RNN and Bi-LSTM models

During testing, single *SimpleRNN* and *LSTM* layers were not effective for text classification; the models were either not learning at all, or training very slowly. Even upon adjustment of hyperparameters, no improvement was witnessed. However, the implementation of bidirectional LSTM layers yielded sufficient results with non-pre-trained embeddings layer, as well as fastText pre-trained embeddings.

For the BERT model, the implementation was created using HuggingFace's pretrained *TFBertModel* and *BertTokenizer*.

```python
def bert(df, d_labels):
    tokenizer = BertTokenizerFast.from_pretrained(pretrained_model_name_or_path=model_name, config=config)

    transformer_model = TFBertModel.from_pretrained(model_name, config = config)

    bert = transformer_model.layers[0]

    input_ids = l.Input(shape=(24,), name="inputs", dtype='int32')
    inputs = {'input_ids': input_ids}

    bert_model = bert(inputs)[1]

    dropout = l.Dropout(config.hidden_dropout_prob, name='pooled_output')
    pooled_output = dropout(bert_model, training=False)

    category = l.Dense(
        units=len(df.category.value_counts()),
        kernel_initializer=TruncatedNormal(stddev=config.initializer_range),
        name = 'category'
        )(pooled_output)

    outputs= {'category': category}
    model = Model(inputs=inputs, outputs=outputs, name="BERT_multilabel")

    X_train, X_test = split_dataset(df)
    Y_train, Y_test, n_classes = create_one_hot_labels(X_train['category'], X_test['category'], d_labels)

    x = tokenizer(
        text=X_train['item_name'].astype(str).to_list(),
        add_special_tokens = True,
        max_length = 24,
        truncation = True,
        padding = True,
        return_tensors='tf',
        return_token_type_ids = False,
        return_attention_mask = False,
        verbose=True
    )

    optimizer = tf.keras.optimizers.Adam(
        learning_rate=5e-05,
        epsilon=1e-08,
        decay=0.01,
        clipnorm=1.0
    )

    loss = {'category': tf.keras.losses.CategoricalCrossentropy(from_logits = True)}
    metric = {'category': tf.keras.metrics.CategoricalAccuracy('accuracy')}

    model.compile(
        optimizer = optimizer,
        loss = loss,
        metrics = metric
    )

    history = model.fit(
        x={'input_ids': x['input_ids']},
        y={'category': Y_train},
        validation_split=0.2,
        batch_size=20,
        epochs=30
    )

    return history
```

Fig. 83 - BERT model implemented with keras

6.5.2 Model architecture experiment results

The following table shows the accuracy scores achieved from the experimentation with traditional algorithms.

Experiment	Accuracy (%)		
	Amazon-PQA	ShopMania	Custom dataset
10) Logistic regression (baseline)	85.9	74.9	81.1
11) Naïve-Bayes	85.0	70.5	69.3
12) SVM	86.3	80.2	79.2
13) Shallow neural network	86.6	81.0	81.2

Table XV - Results from traditional model experiments

Out of the classic models SVM performs the best on average but for all datasets, the basic neural network performed better across the datasets. Additionally, the neural network was implemented with default hyperparameters (static learning rate), and 30 epochs, so the accuracy could be improved with further exploration of hyperparameters.

The results of each experiment are show in Table X. For each dataset, the best result is highlighted in green, and the second-best result is highlighted in blue.

Experiment	Metric	Amazon-PQA	ShopMania	Custom dataset
13.5) Neural network (1 hidden layer)	Accuracy (%)	79.6	75.8	79.5
	Loss	0.83	1.16	0.75
14) TextCNN	Accuracy (%)	80.1	77.4	81.2
	Loss	0.89	1.71	0.84
15) RNN	Accuracy (%)	N/A	N/A	N/A
	Loss	N/A	N/A	N/A
16) Bidirectional LSTM	Accuracy (%)	72.60	67.6	72.0
	Loss	1.62	2.39	1.44
16.5) Bi-LSTM (fT embeddings)	Accuracy (%)	67.3	53.0	74.2
	Loss	2.56	3.12	1.24
17) BERT	Accuracy (%)	84.1	65.6	71.7
	Loss	0.91	2.06	1.30

Table XVI – Further results from model experiments

In reviewing the results, BERT did not perform as well as expected. Where in theory, transformer models typically perform better on text classification tasks than other models, in our experiments this was not reflected. This may be due to the size and simplicity of our dataset. However, word embeddings processed through a shallow CNN performed well in the experiments.

6.6 Including extra data

This section explores how to implement the following experiments as listed in Section 3.5.5 and repeated here for reader convenience.

Exp. no.	Experiment	Model	Dataset	Metrics
18	Item name only	TextCNN	Custom dataset only	Accuracy, Loss
19	Item and store name	TextCNN	Custom dataset only	Accuracy, Loss

6.6.1 Implementation of including extra data experiments

To include the store data as an input into the model, the store names are vectorised similarly to the product text and inputted into a keras embedding layer to create a store name embedding of 50 dimensions.

```python
def get_store_vectors(df, X2_train, X2_test, max_len):
    vectorizer = TextVectorization(output_sequence_length=max_len)

    vectorizer.adapt(df['store_name'])

    vocab = vectorizer.get_vocabulary()

    word_index = dict(zip(vocab, range(len(vocab))))

    X2_train = vectorizer(X2_train)
    X2_test = vectorizer(X2_test)

    print(len(X2_train[0]))

    return X2_train, X2_test, vocab, word_index, vectorizer
```

```python
store_inputs = tf.keras.Input(shape=(50,))

store_embedding = l.Embedding(
    store_tokens,
    dim,
    input_length=50,
    trainable = True
)(store_inputs)

reshape = l.Reshape((max_len, 100, 1))(product_embedding)
pooled = l.GlobalMaxPooling1D()(store_embedding)
```

Fig. 84 - Methods to create store embeddings

This is then concatenated with the product names after they have been convolved over.

```
concatenate = l.Concatenate(axis=1)([flatten,pooled])
```

Fig. 85 - Layer to concatenate product name embeddings and store embeddings

A visualisation of this model with two inputs is depicted in Fig. 86.

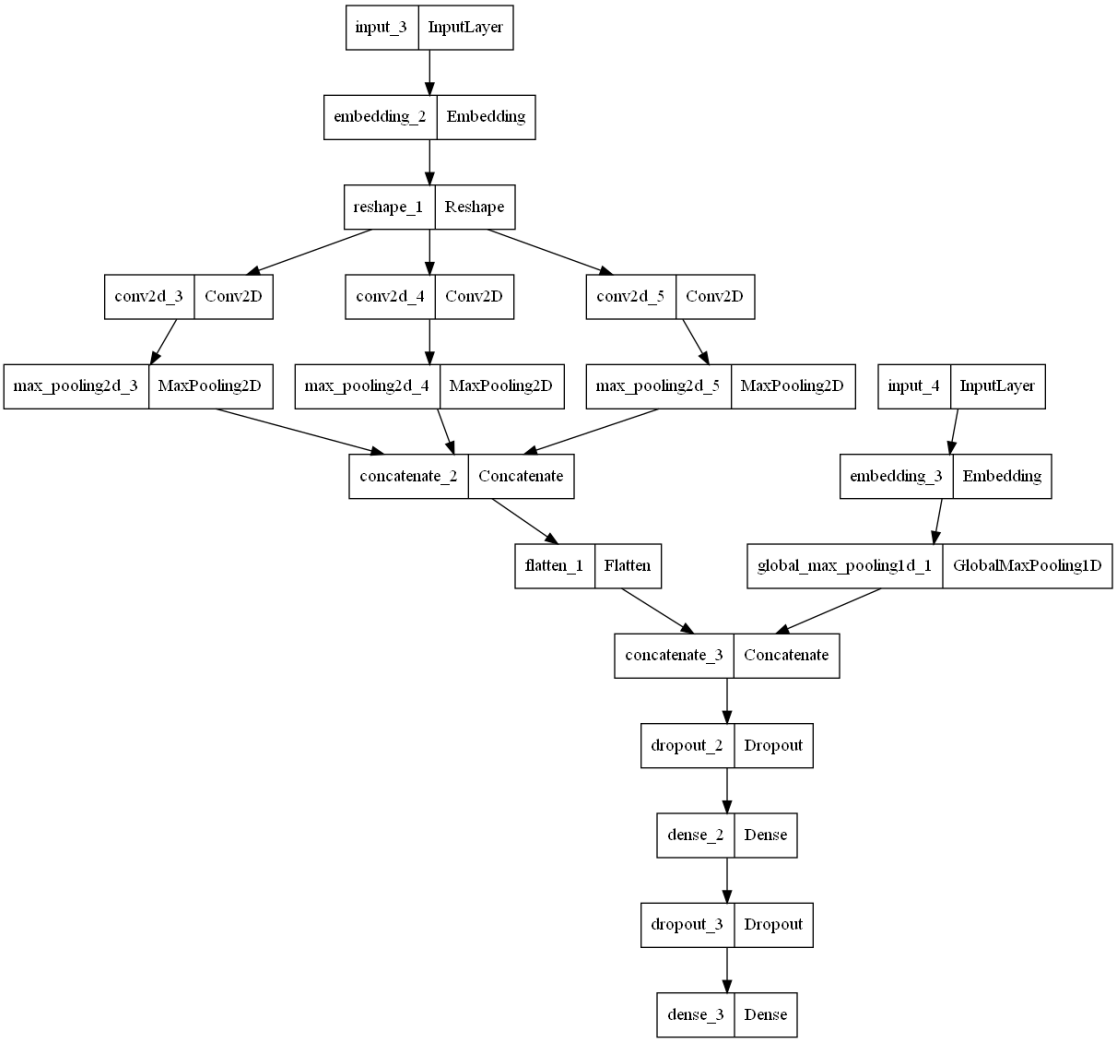


Fig. 86 - Visualisation of model for training with store name embeddings and product name embeddings

6.6.2 Including store name experiment results

The results of including the store name where the product was purchased as an input into the model is shown in the table below.

Experiment	Metric	Custom dataset
Product name only	Accuracy (%)	81.2
	Loss	0.84
Product name and store name	Accuracy (%)	92.9
	Loss	0.52

Table XVII - Result of adding the store name as an input experiment

Including the store name had a significant effect on the performance of the model, increasing accuracy by 10% and reducing loss to almost half the previous value.

The accuracy and loss graph of the performance of the model when only including the product name as an input is shown below.

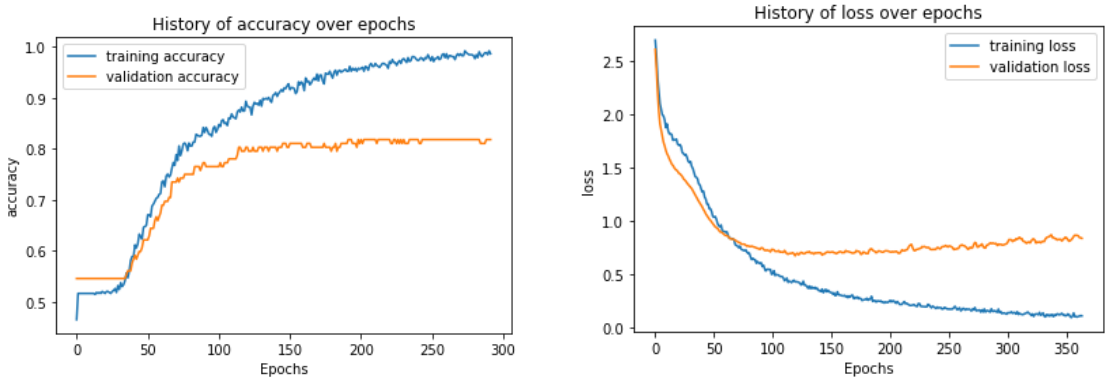


Fig. 87 - History of training and validation accuracy and loss across epochs when training with custom dataset

From these graphs, the model without using store names is underfitted and perhaps only including product name data does not provide the model with enough information to improve accuracy over 80%. By including the store name as an input, the model can better predict the category of the item.

6.7 Evaluation

In this section, the results from the implementation of the text classification model will be discussed in detail. From the results of testing the implementations above, the most effective and well performing elements are chosen for the final text classification component pipeline.

The vocabulary gathered from receipts is distinct and at times limited, and as such, there were a few things to consider when evaluating the results from implementation. The models that were implemented performed differently when trained on data from the Amazon-PQA and ShopMania datasets compared to when they were trained with data from the custom dataset. However, the results were similar enough when comparing aspects of pre-processing, vectorisation, and model architecture, that abstracting the best architecture for text (and specifically product) classification from these results is plausible.

The pre-processing steps in the final model, and the reasoning behind these choices are as follows:

- Ensure all product names are in lowercase, and punctuation is removed as this is the minimum cleaning for classification.
- Remove all numbers as this was shown to improve accuracy during experimentation in Section 6.1.2:

Experiment	Accuracy (%) after 100 iterations of Logistic Regression (3 s.f.)		
	Amazon-PQA	ShopMania	Custom dataset
2) Baseline + Removing digits	84.4	78.7	81.1

Table XVIII - Best results from pre-processing experiments

- Do not stem the words – although stemming improved accuracy on the Amazon-PQA and ShopMania datasets as shown in table, it negatively affected performance for the custom receipt data. This may be due to receipt data already being limited with many words shortened to code versions of the original word. Since product names are short to begin with, contextual indicators in longer texts may help resolve stemmed word meanings, however, when there is only one word in the product name, it will be harder to distinguish between two stemmed words. An example of this are the words "*scratchings"* and *"scratcher"* which would both be stemmed to *"scratch"*. If both products only contained one word, stemming would not be helpful, as there would be no way to indicate the difference between the snack (pork scratchings) and the personal tool (back scratcher).

Experiment	Accuracy (%) after 100 iterations of Logistic Regression (3 s.f.)		
	Amazon-PQA	ShopMania	Custom dataset
Baseline + Stemming	85.1	78.8	79.5

Table XIX - Results of stemming experiment

- Remove stop words (custom stop word list) - as receipt text is already limited and every word on a receipt is valuable, regular stop words will not be removed as this either had a negative effect or no effect on accuracy from testing.

Experiment	Accuracy (%) after 100 iterations of Logistic Regression (3 s.f.)		
	Amazon-PQA	ShopMania	Custom dataset
3) Baseline + Removing stop words	84.0	78.2	79.5

Table XX - Result of removing stop words

However, after more consideration and analysis of the custom dataset, it was necessary to remove custom stop words to filter out retailer specific codes and other redundant words. Examples of stop words include *'cc'*, *'tesco'*, for Tesco Clubcard and *'ws'* for Waitrose which appeared frequently within the custom dataset, but do not affect the semantic meaning of products. For example, *'cc beef'* and *'cc chocolate'* belong to different categories and is not affected by the *'cc'* code. Inevitably, this list would grow with more data from other retailers and would need monitoring and amending. As such, this is an optional suggestion, as it was found to improve performance but not by much.

- Replace all quantities such as *'g'*, *'kg'*, *'litres'*, *'pk'* with standard quantity indicators (such as <UNIT_WEIGHT>, <UNIT_LIQUID>, <UNIT_SINGULAR>) – similar to the point above, quantity indicators hold similar semantic meaning with *'g'* and *'kg'* offering identical context indicators but differing in presentation. For this reason, it was decided to replace quantity indicators with standardised tokens, so that liquids and solids can still be differentiated.

The final pre-processing methods for the custom dataset are implemented as below:

```
stop = ['cc', 'wr', 'tesco', 'aa', 'ess']

units_solid = ['kg', 'g', 'gram', 'grams']
units_liquid = ['ml', 'l', 'cl', 'ltr']
units_length = ['m', 'mm', 'cm', 'km', 'metre']
units = ['pk', 'sm', 'lg', 'large', 'extra', 'sng']

def remove_stopwords(text):
    text = [word for word in text.split() if (word.lower() not in stop and len(word) > 1)]
    return " ".join(text)

def replace_units(text):
    words = []
    for word in text.split():
        if word in units_solid:
            words.append('<UNIT_SOLID>')
        elif word in units_liquid:
            words.append('<UNIT_LIQUID>')
        elif word in units:
            words.append('<UNIT>')
        else:
            words.append(word)
    return " ".join(words)

dataset['item_name'] = dataset['item_name'].astype(str)
dataset['item_name'] = dataset['item_name'].apply(lower)
dataset['item_name'] = dataset['item_name'].apply(remove_punctuation)
dataset['item_name'] = dataset['item_name'].apply(remove_nums)
dataset['item_name'] = dataset['item_name'].apply(replace_units)
dataset['item_name'] = dataset['item_name'].apply(remove_stopwords)
```

Table XXI - Final text pre-processing pipeline

Then once the text has been pre-processed, it will be transformed into word embedding format using the pretrained fastText model as word embeddings produced the best results as reiterated in Table XXII. Furthermore, fastText embeddings produced the best average performance across the 3 datasets.

Experiment	Metric	Amazon-PQA	ShopMania	Custom dataset
fastText	Accuracy (%)	79.6	75.8	79.5
	Loss	0.83	1.16	0.75

Table XXII - Results of fastText embedding experiment

For the actual classification, TextCNN will be used as the model, as it performed the best in a reasonable amount of time.

Experiment	Metric	Amazon-PQA	ShopMania	Custom dataset
14) TextCNN	Accuracy (%)	80.1	77.4	81.2
	Loss	0.89	1.71	0.84

Table XXIII - Results of TextCNN implementation

Although BERT should have been able to produce better results, in practice this was not true. Additionally, the training time for the model was too high. This could cause issues if the pipeline was to be deployed, since retraining the model on new data would take an excessive amount of time if using BERT, however it could be considered if the model will not be retrained frequently.

Finally, based on the last set of experiments, it was found that including the name of the store where a product was purchased at increases the model's ability to predict an item's category.

The final pipeline trained on our custom dataset reached a validation accuracy of 92.9% which is more than the target accuracy as outlined in Section 1.4. The text classification model is visualised in Fig. 86 in Section 6.6.2.

7 Statement of Ethics

Ethics are defined as "moral principles that govern a person's behaviour or the conducting of an activity" in [93]. This section of the report will address the legal, social, ethical, and professional factors that were considered throughout the course of this project, and how these factors may have affected the execution of the project. This will involve reviewing the IEEE Computer Society and ACM's Code of Ethics, the BCS Code of Conduct, the Computer Misuse Act (1990), and the UK GDPR amongst other relevant current legislations. Additionally, this section will also consider the project's necessity, contribution to, and impact on current society.

7.1 Code of Conduct

It is a computer scientist's duty to act ethically and to serve public interest. This includes ensuring to "do no harm" and abide by any applicable laws in force.

Legally, The Computer Misuse Act (1990) was enforced to prevent unauthorised access to computer systems, making hacking an imprisonable offence. This includes any attempts to steal data or destroy devices, but also any unsanctioned modification or deletion of data. No such activities were involved in this project and thus this particular law was abided by completely.

Ethically, the aim of the project to develop a machine learning pipeline to analyse receipt data is not intended to cause harm but must be analysed to consider whether the results of the project could be used adversely. In this subsection, we discuss the necessity, goodness, and potential consequences of the project within its field.

Whilst there exist receipt extraction techniques for businesses, there are none for personal users, as researched in Section 2.1. A consideration to make is the effect that the project outcomes could have on the public. For example, the pipeline proposed in this paper could be used to create a mobile app to release to the public. In this instance, we must analyse whether this could negatively or positively impact society. The aim of providing such a pipeline is to provide an easier alternative to budgeting, and to improve the financial well-being and capability of the public. Though the ideas proposed could help more people budget, it may also have an adverse effect of causing stress to those who already worry about their finances as it can place their purchases under a microscope for them.

In [4], it was found that those who said they budget were more likely to be worried about their financial status and have negative feelings regarding finances. However, though there is a correlation between budgeting and negative feelings, this does not necessarily indicate that budgeting is the causation of negative feelings. Instead, it seems to imply that previously only those in extreme circumstances would consider budgeting due to the tediousness of manual bookkeeping, which can be mentally taxing. The aim of the project is to ease the process of managing finances and spending, which would be beneficial for both burdened and unburdened individuals, as it could help reduce the time spent calculating and thinking about where money goes.

7.2 Professional practices

During the project, and throughout this report, care has been taken to ensure any ideas that originated from others are credited appropriately. This is an important consideration as it affects the integrity of the work carried out and the results reported. Any theory, work, academic paper, or solutions that were reviewed or gave inspiration to the ideas in this report have been properly cited and referenced.

Additionally, as the pipeline suggested will not be released commercially and was created for research purposes, licensing of software or code was not a consideration that needed to be made during the project. However, should the pipeline be used or released in the future, the components discussed and implemented for this project are all open-source and most are licensed under the Apache License 2.0 [94]. The Apache License 2.0 is a free software license that permits anyone to use, modify, and distribute the software; this includes for commercial uses – as long as the license, copyright, and notice of usage of the software is made clear. Whilst open-source software is largely considered to be ethical due to the freedom that the licensing permits, there are still factors to be considered. For example, when using open-source software, datasets, and packages to complete project, it is necessary to recognise and appreciate the software and research community contributions behind these repositories. To respect the work of previous contributors, open-source software should be used as intended ('do no harm') and not abused for illegal or unethical purposes.

Other considerations that must be taken into account when using open-source software is the user's responsibility to report any issues found for the improvement of the software. This not only applies to bugs but also to other flaws in the design such as unperceived biases of software that need to be addressed.

7.3 Data Protection

As the project involves financial data, the implications of data protection and safety must be thoroughly considered. Both legally and ethically, the privacy and confidentiality of others must be respected, under UK law and ACM's Code of Ethics. The UK GDPR (which is the UK's implementation of the European Union's General Data Protection Regulation) states that personal data is protected and must be processed lawfully and fairly [95]. Personal data is any data that is related to an identifiable person according to [96]. This refers to direct identifiers (such as name, email, ID numbers, and card numbers) or data that when collected in combination can be assigned to a person (this could include date of birth, address/location data, and IP addresses). Any identifiable data is subject to GDPR laws, which states that any such information collected must be used fairly and transparently, and only for explicit and specified purposes among other requirements. It must also be accessible to identifiable individuals at any time.

Aspects of the project that have involved potentially sensitive data include the survey that was conducted on budgeting habits and financial situation, and the custom dataset curation which involved collecting personal sales receipts.

In the financial survey that was sent out (please see Appendix A) all responses received were anonymous so it does not fall under GDPR. Ethically, participation in the survey was strictly voluntary. Care was taken to ensure no personal or identifiable data was asked for in the survey, and only age and student status were asked to be provided (which are not identifiers, even when used in conjunction). To ensure survey participants did not feel pressured to answer on more sensitive topics, "Prefer not to say" was provided as an option on difficult questions such as "Have you ever been stressed, anxious, or worried about your personal finances?". Additionally, care was taken to ensure the results would not be skewed by discrimination, or leading questions, and participants would have the option to expand on answers (through providing boxes for long answer inputs).

Then, throughout the course of the project, there was an active effort to collect receipts for the custom dataset. This means that the dataset contains personal receipts collected from friends and family. As such, it was important to ensure that as a data controller and processor, the data subjects were treated with respect. It was decided that any data that would be collected would be anonymised so that it would not be able to be linked to any one person, and that any contributions to the dataset remain anonymous.

All contributors were made aware of the project aims and informed of how their data would be used, and that it would be stored anonymously. It was ensured that the receipts used for this project would not contain any identifiable information so that the data from the receipts cannot be paired to a specific person. All contributions were voluntary, and no receipts were taken without consent.

During the collection period, any physical receipts that were provided were placed into a folder for safe keeping. After this period, physical receipts were shuffled before being processed. Photos of the receipts were then taken, and the physical receipts were securely disposed of through a shredder.

Whilst many of the collected receipts did not contain any identifiable information, some receipts that involved card payments list the purchaser's card type and last four digits of their card. Although the last four digits of a card and the card type are not unique and therefore cannot be linked to any individual, other receipts also included first name (restaurant and delivery orders, for example, or other potential identifiers such as purchase IDs/customer ID), and to preserve anonymity it was essential for this data to be hidden. As the potential identifiers were not necessary to the project (since the main project aim is to extract the purchased items list), it was decided that card information and other personal indicators would be removed by blurring out this type of information on the pictures of the collected receipts. This was done manually across all receipts to ensure that any personal details were hidden.

Although the most obvious data subjects in relation to the receipt dataset are the receipt contributors, other data parties also exist. Receipts usually contain store information such as location and store ID, but during processing of the dataset it was found that occasionally, customer service representative data is also shown. This is an indirect data subject which requires consideration. There would be no reasonable way to ensure that the specified party agrees to their name, clerk ID, location ID or other identifiers to be shared. Therefore, it

was also decided that this information would be removed from the receipts. Again, the removal of store locations, store IDs, clerk IDs, clerk names, and other such data had no effect on the outcomes or progression of the project and serves the purpose of protecting the identities of store employees.

The original digitised receipts were kept on a password encrypted PC and then deleted after being used to generate a version without card details, as it is necessary for the dataset to be made available to others along with the code for this project.

The other datasets used in this project are the Amazon-PQA dataset and the ShopMania dataset. Amazon-PQA contains questions and answers from users, but no personal data (including user IDs) are included in the dataset. The ShopMania dataset only contains product information.

Neither the SROIE dataset nor the CORD dataset contain personal data either, with Park et al. having specifically removed any identifiable information (for example, name, and card number) from the CORD receipts.

During this project, where pictures of receipts were manually edited to ensure no personal data was present, this may not always be true especially if the project outcomes are used for processing further real-life unedited data. This would have to be considered by anyone who wishes to further the project or develop an app based off the pipeline presented in this project. Potential solutions include encrypting receipts throughout different parts of the pipeline, or having the model be integrated into the app on the local machine which would prevent data being passed to and from a hosted server, and only accessible by the intended user. Other solutions could include password locks and other privacy mechanisms for apps.

Overall, the confidentiality of those involved in this project and the project outcomes have been thoroughly considered and evaluated and care has been taken to ensure no abuses of privacy took place during the project. This is also reflected in the Self-Assessment for Governance and Ethics - Human and Data Research (SAGE-HDR) completed as part of this project. The SAGE-HDR form is available in Appendix B.

8 Final proposed pipeline

In this chapter, the final pipeline will be presented using example data to demonstrate the working components and how they work together. The diagram of the pipeline as proposed in Section 3.1.2 is shown again for reader convenience.

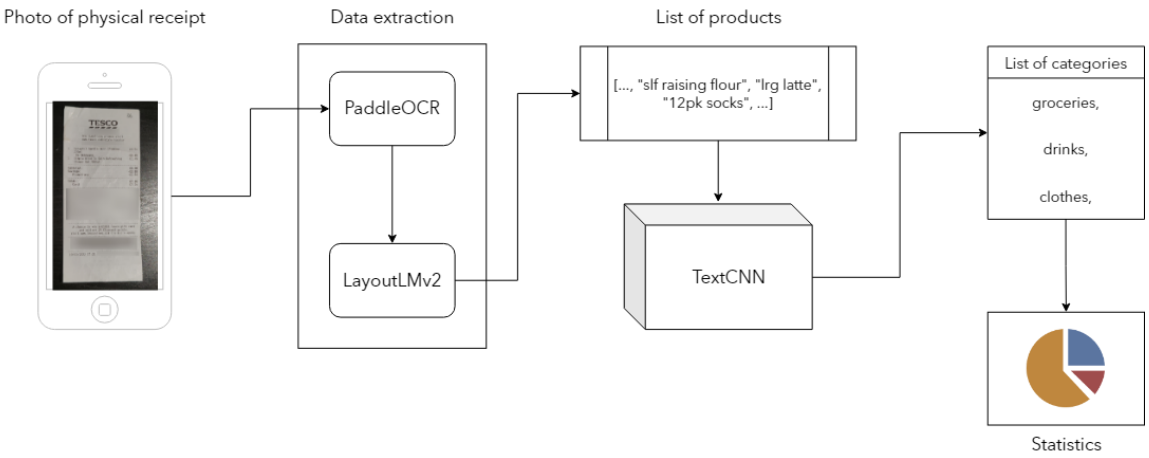


Fig. 88 - Final proposed pipeline

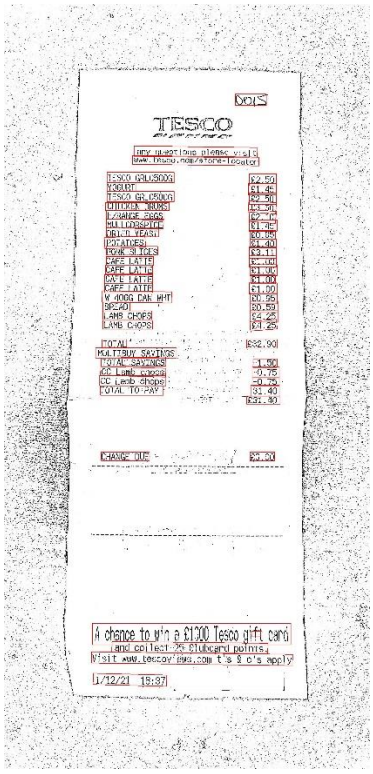
8.1 OCR component

First, a user takes a photo of a receipt, preferably on a clear and contrasting background to produce the best OCR results. However, the images are pre-processed prior to going through OCR which improves the quality of text recognised from the receipts as discussed in Section 4.5.

OpenCV is used to transform the picture to black and white, and then adaptive thresholding is applied to the image. An example of an image from the custom dataset prior to pre-processing, and after pre-processing is shown below.

Fig. 89 - Example of receipt from custom dataset before and after thresholding is applied

After the photo has been pre-processed, the text is retrieved using PaddlePaddle's OCR engine; the results from scanning the image are saved with bounding box information and the raw text recognised from the picture.




```
[[[1137.0, 444.0], [1290.0, 444.0], [1290.0, 499.0], [1137.0, 499.0]], ('OOS', 0.5678446888923645)]
[[[639.0, 699.0], [1246.0, 707.0], [1245.0, 758.0], [639.0, 750.0]], ('any questions please visit', 0.9605622887611389)]
[[[631.0, 746.0], [1254.0, 754.0], [1253.0, 797.0], [631.0, 789.0]], ('www.tesco.com/store-looator', 0.9812812209129333)]
[[[506.0, 829.0], [836.0, 829.0], [836.0, 880.0], [506.0, 880.0]], ('TESCO GRLC500G', 0.9838820099830627)]
[[[1205.0, 836.0], [1338.0, 836.0], [1338.0, 895.0], [1205.0, 895.0]], ('02.50', 0.8789843320846558)]
[[[506.0, 880.0], [647.0, 880.0], [647.0, 923.0], [506.0, 923.0]], ('YOGURT', 0.9982645511627197)]
[[[1201.0, 884.0], [1338.0, 884.0], [1338.0, 939.0], [1201.0, 939.0]], ('1.45', 0.9984419941902161)]
[[[506.0, 923.0], [836.0, 923.0], [836.0, 974.0], [506.0, 974.0]], ('TESCO GRLC500G', 0.9818261861801147)]
[[[1205.0, 931.0], [1338.0, 931.0], [1338.0, 986.0], [1205.0, 986.0]], ('2.50', 0.9998048543930054)]
[[[498.0, 978.0], [812.0, 978.0], [812.0, 1017.0], [498.0, 1017.0]], ('CHICKEN DRUMS', 0.9963753819465637)]
[[[1204.0, 969.0], [1339.0, 979.0], [1335.0, 1034.0], [1200.0, 1024.0]], ('23.50', 0.869147002696991)]
[[[502.0, 1017.0], [788.0, 1017.0], [788.0, 1060.0], [502.0, 1060.0]], ('F/RANGE EGGS', 0.9975857138633728)]
[[[1209.0, 1012.0], [1339.0, 1022.0], [1334.0, 1081.0], [1204.0, 1071.0]], ('$2.10', 0.8915732502937317)]
[[[503.0, 1060.0], [784.0, 1064.0], [783.0, 1108.0], [502.0, 1103.0]], ('MULLCDRSPICE', 0.9776263236999512)]
[[[1205.0, 1064.0], [1342.0, 1064.0], [1342.0, 1119.0], [1205.0, 1119.0]], ('1.45', 0.9891484975814819)]
[[[502.0, 1104.0], [759.0, 1104.0], [759.0, 1147.0], [502.0, 1147.0]], ('DRIED YEAST', 0.997506499290466 3)]
[[[1209.0, 1115.0], [1334.0, 1115.0], [1334.0, 1158.0], [1209.0, 1158.0]], ('0.85', 0.9996396899223328)]
[[[493.0, 1151.0], [693.0, 1142.0], [696.0, 1197.0], [496.0, 1206.0]], ('POTATOES', 0.9966012835502625)]
[[[1209.0, 1158.0], [1338.0, 1158.0], [1338.0, 1202.0], [1209.0, 1202.0]], ('E1.40', 0.8972908854484558)]
[[[502.0, 1194.0], [755.0, 1194.0], [755.0, 1237.0], [502.0, 1237.0]], ('PORK SLTCES', 0.950711190700531)]
[[[1205.0, 1198.0], [1338.0, 1198.0], [1338.0, 1253.0], [1205.0, 1253.0]], ('23.11', 0.9517288208807812)]
[[[502.0, 1245.0], [735.0, 1245.0], [735.0, 1288.0], [502.0, 1288.0]], ('CAFE LATTE', 0.966411292552948)]
[[[1205.0, 1241.0], [1338.0, 1241.0], [1338.0, 1296.0], [1205.0, 1296.0]], ('1.00', 0.9987834095954895)]
[[[498.0, 1288.0], [731.0, 1288.0], [731.0, 1331.0], [498.0, 1331.0]], ('CAFE LATTE', 0.8787266612052917)]
[[[1209.0, 1284.0], [1342.0, 1284.0], [1342.0, 1339.0], [1209.0, 1339.0]], ('1.00', 0.999845027923584)]
[[[494.0, 1335.0], [731.0, 1335.0], [731.0, 1378.0], [494.0, 1378.0]], ('CAFE LAtte', 0.6817888021469116)]
[[[1213.0, 1335.0], [1334.0, 1335.0], [1334.0, 1378.0], [1213.0, 1378.0]], ('1.00', 0.9995998740196228)]
[[[494.0, 1378.0], [731.0, 1378.0], [731.0, 1422.0], [494.0, 1422.0]], ('CAFE LATTE', 0.9321764707565308)]
[[[1205.0, 1374.0], [1342.0, 1374.0], [1342.0, 1429.0], [1205.0, 1429.0]], ('1.00', 0.9994626045227051)]
[[[483.0, 1413.0], [824.0, 1418.0], [823.0, 1473.0], [482.0, 1469.0]], ('W 40OG DAN WHT', 0.9029567837715149)]
[[[1209.0, 1422.0], [1342.0, 1422.0], [1342.0, 1477.0], [1209.0, 1477.0]], ('00.95', 0.9046783447265625)]
[[[486.0, 1469.0], [619.0, 1469.0], [619.0, 1512.0], [486.0, 1512.0]], ('BREAD', 0.9989264607429504)]
[[[1209.0, 1465.0], [1346.0, 1465.0], [1346.0, 1520.0], [1209.0, 1520.0]], ('20.59', 0.9993699789047241)]
[[[486.0, 1512.0], [727.0, 1512.0], [727.0, 1555.0], [486.0, 1555.0]], ('LAMB CHOPS', 0.9589815139770508)]
[[[1209.0, 1508.0], [1346.0, 1508.0], [1346.0, 1567.0], [1209.0, 1567.0]], ('4.25', 0.9981428384780884)]
[[[486.0, 1559.0], [731.0, 1559.0], [731.0, 1602.0], [486.0, 1602.0]], ('LAMB CHOPS', 0.9984245300292969)]
[[[1208.0, 1564.0], [1339.0, 1554.0], [1343.0, 1601.0], [1212.0, 1611.0]], ('4.25', 0.9995535612106323)]
[[[482.0, 1653.0], [603.0, 1653.0], [603.0, 1696.0], [482.0, 1696.0]], ('TOTAL', 0.9360522031784058)]
[[[1193.0, 1653.0], [1342.0, 1653.0], [1342.0, 1696.0], [1193.0, 1696.0]], ('32:90', 0.9084435701370239)]
[[[450.0, 1693.0], [848.0, 1693.0], [848.0, 1744.0], [450.0, 1744.0]], ('MULTIBUY SAVINGS:', 0.9480230808258057)]
[[[478.0, 1744.0], [795.0, 1739.0], [796.0, 1783.0], [478.0, 1787.0]], ('TOTAL ISAVINGS', 0.93006670475006 1)]
[[[1225.0, 1748.0], [1342.0, 1748.0], [1342.0, 1791.0], [1225.0, 1791.0]], ('-1.50', 0.9915622472763062)]
[[[475.0, 1786.0], [796.0, 1795.0], [795.0, 1838.0], [474.0, 1830.0]], ('cc Lamb chops', 0.971774392805481)]
[[[1221.0, 1787.0], [1354.0, 1787.0], [1354.0, 1846.0], [1221.0, 1846.0]], ('-0.75', 0.978802502155304)]
[[[470.0, 1830.0], [796.0, 1830.0], [796.0, 1881.0], [470.0, 1881.0]], ('CC Lamb chops', 0.9935648441314697)]
[[[1221.0, 1838.0], [1354.0, 1838.0], [1354.0, 1881.0], [1221.0, 1881.0]], ('-0.75', 0.9934003949165344)]
[[[474.0, 1881.0], [783.0, 1881.0], [783.0, 1928.0], [474.0, 1928.0]], ('totAl to PAY:', 0.7029151320457458)]
```


Fig. 90 - Example of bounding boxes of text localised with PaddleOCR and the OCR output

Visualising the bounding boxes shows that PaddleOCR recognises all of the text presented on the receipt (besides the Tesco logo). The text recognised also aligns closely with the real ground truth text although pound symbols seem to not be recognised by the OCR engine. The processed photo, bounding boxes, and text are then processed through the KIE component.

8.2 KIE component

The KIE component of the pipeline will use a pretrained LayoutLMv2 model that has been finetuned on receipt data with the following labels:

- store_name
- date
- item_name
- quantity
- category
- price

The data from the OCR component will be passed to LayoutLMv2 to predict the labels of recognised text clusters. The labels of text and the raw text will then be saved as a JSON representation of the receipt.


```
{
  "id": "0015",
  "date": "2021-12-01",
  "storeName": "TESCO",
  "itemList": [
    {
      "itemName": "TESCO GRLC500G",
      "quantity": 1,
      "category": "",
      "price": 2.50
    },
    {
      "itemName": "YOGURT",
      "quantity": 1,
      "category": "",
      "price": 1.45
    },
    {
      "itemName": "TESCO GRLC500G",
      "quantity": 1,
      "category": "",
      "price": 2.50
    },
    {
      "itemName": "CHICKEN DRUMS",
      "quantity": 1,
      "category": "",
      "price": 3.50
    },
```


Fig. 91 - Example of expected structured json file to be returned by KIE component

The empty category field will be completed by the text classification system.

8.3 Product classification component

As specified in Section 3.3.4, there are 16 categories that a product can potentially be classified under:

- groceries
- drinks
- snacks
- eating out
- alcohol
- personal hygiene
- cleaning & laundry
- electrical
- homeware
- clothes & accessories
- health & pharmacy
- entertainment
- nicotine
- gardening
- other

The product classification model will take the names of the products, and the store name to output the category of the products and complete the empty category field of the JSON file.

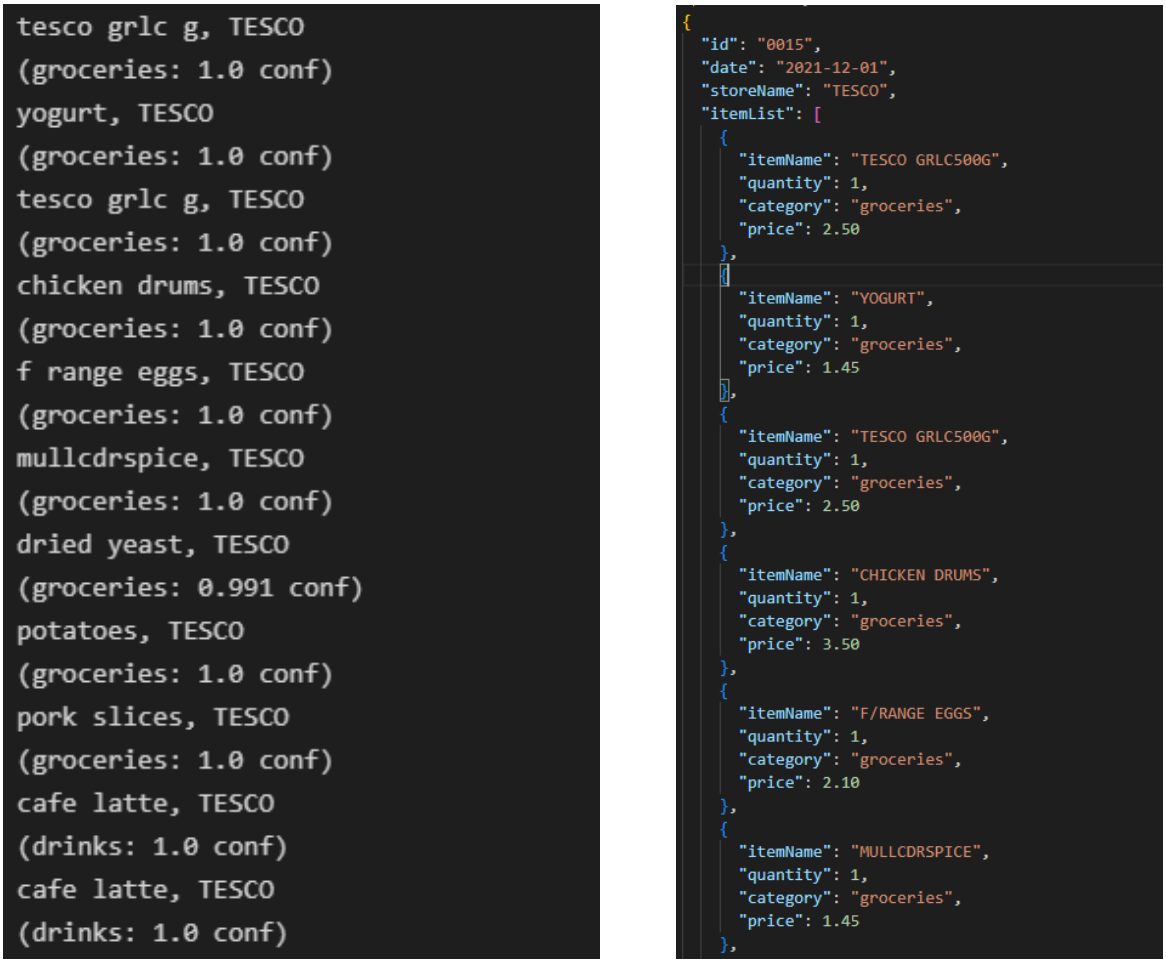


Fig. 92 - Example of implemented model predictions on the receipt from Fig.90.

8.4 Data analysis component

Once the categories have been predicted, a simple summary of the receipt can be created and displayed for the user. Once the user has built up a collection of scanned receipts that has been parsed through the pipeline, other analytics could also be created. For example, the dates of receipts and purchase information could be used to display how purchases are spread across categories in one month. More detailed analyses could also be generated depending on the application or a user's particular use case.

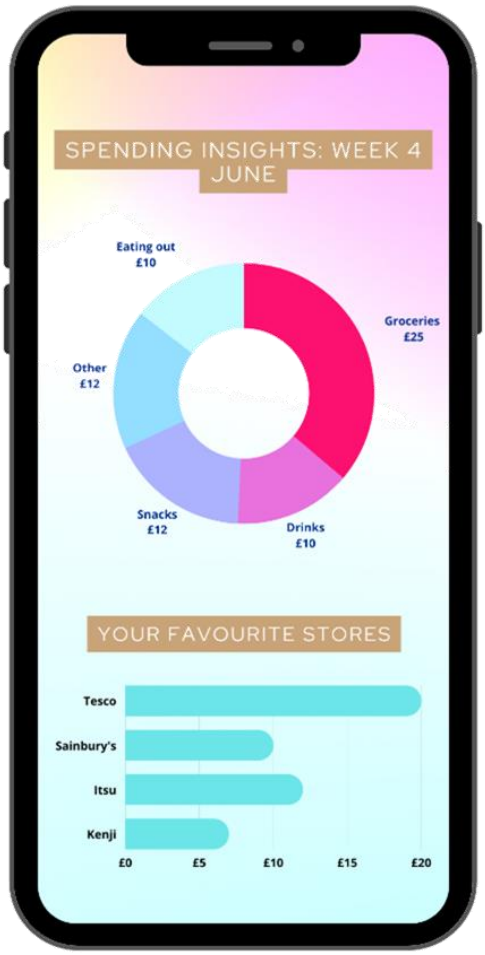


Fig. 93 - Mock-up of potential insights for a mobile budgeting app

9 Project Evaluation

This chapter concludes the project and reflects on the project's overall trajectory by revisiting the aims of the project, as listed in Section 1.3 and 1.4 and reviewing the overall project progression. It also includes a discussion into extending upon the work presented in this project.

The success criteria and objectives are reiterated below for reference.

The objectives of the project were to:

7. Identify appropriate receipt OCR solutions and methods
8. Identify appropriate methods of machine learning for text classification
9. Design and implement a method for extracting key information from a picture of a receipt
10. Design and implement a text classification system for the text retrieved from invoices and receipts
11. Evaluate the OCR method implemented to extract text from a photo
12. Evaluate the NLP method implemented to classify product items into groups

For the project to be considered a success there will exist:

- A demonstrable working OCR method of extracting text from a picture of a receipt taken on a mobile phone
- A demonstrable working model that can extract an itemised list of purchases from OCR output of a receipt
- A demonstrable working text classification model that can categorise product names with over 80% accuracy

9.1 Evaluation of project objectives

For each of the objectives as listed above, a summary of whether the objective was met and how it was met is provided below.

1. 3 appropriate OCR methods that could be used to retrieve text from an image of a receipt were identified and researched during the Literature Review and problem analysis. TesseractOCR, PaddleOCR, and EasyOCR were identified as being suitable OCR tools for the task. The effect of pre-processing pictures was also identified as a potential method to increase accuracy of OCR.
2. Traditional and modern machine learning models for text classification were researched during the Literature Review. 7 models were identified for experimentation during problem analysis: LR, SVM, NB, a shallow neural network, TextCNN, RNN, and BERT. Additional experiments on pre-processing, tokenisation, and vectorisation were also performed.
3. From objective 1, the most successful OCR tool (PaddleOCR) was selected for the OCR component of retrieving text from a receipt. LayoutLMv2 was discussed for key information extraction as it was identified to be the most current and successful method for KIE to extract information from a document.
4. The 7 models as discussed from objective 2 were experimented with in Chapter 6 and the results influenced the final design of a text classification NLP model to classify products into categories.
5. The effectiveness and results of the OCR tool selected was evaluated in Section 4.5. Numerical metrics were used along with visual confirmation of the OCR method.
6. The NLP method selected to classify products into categories was analysed in Section 6.7 with an in-depth discussion into why each decision for the text classification method was made.

9.2 Evaluation of project aims

Through in-depth research into current OCR methods and text classification methods, this project proposes an end-to-end pipeline to extract product text from an image taken on a mobile phone and classify this text into a category for further use.

As demonstrated:

- An OCR method has been implemented that can extract text from a picture of a receipt taken on a mobile phone. This is demonstrated with real receipts in Chapters 4 and 8.
- A KIE method has been demonstrated to work on a dataset of receipts to extract an itemised list of purchases from the OCR output of a receipt in Chapter 5 and can be modified to work for the proposed pipeline
- A text classification model has been implemented that can categorise product names given the store name that the product was purchased in, and the product name as retrieved from the OCR and KIE components. The model had a validation accuracy of 92.9% which surpasses the 80% aim that was specified in Section 1.4

The success criteria have largely been met and the project has achieved its aims and objectives to an extent. An improvement that could be made would be to modify the KIE method for finetuning on the custom dataset, but due to time constraints, this was not possible. However, the KIE component was demonstrated to work on the CORD receipt dataset successfully which shows that key receipt information can be extracted using the KIE method suggested and explored in Section 5.2.

9.3 Conclusion

Completing this project has been challenging at times, however, despite some roadblocks, the project has managed to achieve its aims and objectives.

Researching current solutions showed that the receipt OCR space is seeing many exciting advancements, with a big focus on improving and adapting transformer technology for NLP and KIE tasks. Simultaneously, even methods that had previously fallen out of favour (CNNs), are seeing reignited interest as new research show significant improvements on previous methods.

The key information extraction space especially is seeing new and innovative solutions that deserve to be discussed in more depth. Whilst a demonstration of the LayoutLMv2 technology on a relevant dataset was shown in Section 5.2, newer more successful KIE models could likely appear in the near future and warrant a renewed discussion of KIE technology for receipt data.

Overall, this project has been a successful and meaningful endeavour into exploring current machine learning methods and techniques for OCR, NLP, and KIE, and has complemented learning from other modules.

9.4 Future work

As discussed in the previous sections, future work could involve revisiting the KIE component of the pipeline to explore the KIE space more. Furthermore, as the pipeline proposed is for a receipt scanner application, research into deployment of such an application could be explored.

Another aspect to consider for continuing work is to continue building the custom dataset. The dataset used for the project was small compared to other datasets (such as CORD and SROIE) but given the popularity of the receipt datasets used for this project, the usefulness of a more robust and abundant dataset could be beneficial to for the ML community. Additionally, methods to improve learning and abstraction could be explored in tandem with increasing the size of the dataset used to train the model. Alternatively, unsupervised machine learning methods could be explored to further the project.

Another worthwhile endeavour would be to solve the problem encountered during problem analysis involving receipts from other countries containing foreign languages. Developing an implementation that could interpret receipts in any language could be useful for those who travel a lot.

Bibliography

[1] 'Veryfi OCR API for Real-Time Data Extraction from Receipts & Invoices', *Veryfi*. https://www.veryfi.com/receipt-ocr-api/ (accessed Feb. 23, 2022).

[2] 'Real-time Receipt OCR API for developers', *TAGGUN*. https://www.taggun.io/ (accessed Feb. 28, 2022).

[3] 'Receipt OCR'. https://nanonets.com/receipt-ocr (accessed Feb. 28, 2022).

[4] Money Advice Service, 'Financial Capability Survey, 2018'. doi: 10.5255/UKDA-SN-8454-2.

[5] 'UK Strategy for Financial Wellbeing | The Money and Pensions Service'. https://moneyandpensionsservice.org.uk/uk-strategy-for-financial-wellbeing/ (accessed May 05, 2022).

[6] 'NEW Receipt OCR API Page', *Veryfi*. https://www.veryfi.com/new-receipt-ocr-api/ (accessed Feb. 28, 2022).

[7] 'Receipt OCR | Data extraction API & Capturing SDK', *Klippa*. https://www.klippa.com/en/ocr/financial-documents/receipts/ (accessed Feb. 28, 2022).

[8] 'Lucidtech: OCR API for Invoices, receipts and other documents'. https://lucidtech.ai/receipt-api.html (accessed Feb. 28, 2022).

[9] 'QuickBooks Self-Employed: Track Mileage & Receipts – Apps on Google Play'. https://play.google.com/store/apps/details?id=com.intuit.qbse&hl=en_GB&gl=US (accessed Feb. 28, 2022).

[10] P. Janík, 'Receipt database with OCR scan', B.S. Thesis, Masaryk University, Faculty of Informatics, 2021. Accessed: Feb. 12, 2022. [Online]. Available: https://is.muni.cz/th/vz51c/

[11] 'ReceiptManager', *GitHub*. https://github.com/ReceiptManager (accessed Apr. 28, 2022).

[12] 'Money Manager Expense & Budget – Apps on Google Play'. https://play.google.com/store/apps/details?id=com.realbyteapps.moneymanagerfree&hl=en_GB&gl=GB (accessed Feb. 28, 2022).

[13] 'Money Manager Expense & Budget'. https://www.realbyteapps.com/ (accessed Feb. 28, 2022).

[14] 'Monefy - Budget Manager and Expense Tracker app – Apps on Google Play'. https://play.google.com/store/apps/details?id=com.monefy.app.lite&hl=en_GB&gl=US (accessed Feb. 28, 2022).

[15] 'Household Budget Survey - Access to microdata - Eurostat'. https://ec.europa.eu/eurostat/web/microdata/household-budget-survey (accessed Feb. 28, 2022).

[16] L. Benedikt, C. Joshi, L. Nolan, R. Henstra-Hill, L. Shaw, and S. Hook, 'Human-in-the-loop AI in government: a case study', in *Proceedings of the 25th International Conference on Intelligent User Interfaces*, Cagliari Italy, Mar. 2020, pp. 488–497. doi: 10.1145/3377325.3377489.

[17] 'What is a Human in the Loop? | Humans in the Loop', Mar. 24, 2021. https://humansintheloop.org/what-is-a-human-in-the-loop/ (accessed Apr. 28, 2022).

[18] de Wolf, 'Optical Character Recognition and Machine Learning Classification of Shopping Receipts', p. 58.

[19] R. Raoui-Outach, C. Million-Rousseau, A. Benoit, and P. Lambert, 'Deep learning for automatic sale receipt understanding', in *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Montreal, QC, Nov. 2017, pp. 1–6. doi: 10.1109/IPTA.2017.8310088.

[20] 'What is Azure Form Recognizer?', *Azure Applied AI Services*. https://docs.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/overview (accessed Apr. 28, 2022).

[21] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. Accessed: May 24, 2022. [Online]. Available: http://neuralnetworksanddeeplearning.com

[22] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization'. arXiv, Jan. 29, 2017. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/1412.6980

[23] Y. Jin and F. Guerin, 'Class Lecture, Topic: "Artificial Neural Networks and Supervised Learning: Shallow versus Deep" COM3013, Department of Computer Science, University of Surrey, Guildford'.

[24] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd edition draft. 2022. Accessed: May 08, 2022. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[25] '1.4. Support Vector Machines', *scikit-learn*. https://scikit-learn/stable/modules/svm.html (accessed May 24, 2022).

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, 'ImageNet classification with deep convolutional neural networks', *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[27] K. He, X. Zhang, S. Ren, and J. Sun, 'Deep Residual Learning for Image Recognition'. arXiv, Dec. 10, 2015. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/1512.03385

[28] A. Amidi and S. Amidi, 'CS 230 - Recurrent Neural Networks Cheatsheet'. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks (accessed May 22, 2022).

[29] 'Long Short-Term Memory'.

[30] A. Vaswani *et al.*, 'Attention Is All You Need'. arXiv, Dec. 05, 2017. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/1706.03762

[31] A. Dosovitskiy *et al.*, 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale'. arXiv, Jun. 03, 2021. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/2010.11929

[32] X. Chen, L. Jin, Y. Zhu, C. Luo, and T. Wang, 'Text Recognition in the Wild: A Survey'. arXiv, Dec. 03, 2020. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/2005.03492

[33] E. H. A and R. N, 'OCR Accuracy Improvement on Document Images Through a Novel Pre-Processing Approach', *Signal Image Process. Int. J.*, vol. 6, no. 4, pp. 01–18, Aug. 2015, doi: 10.5121/sipij.2015.6401.

[34] R. Smith, 'An Overview of the Tesseract OCR Engine', in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, Curitiba, Parana, Brazil, Sep. 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[35] T. M. Breuel, 'The OCRopus open source OCR system', San Jose, CA, Jan. 2008, pp. 68150F-68150F–15. doi: 10.1117/12.783598.

[36] A. Ul-Hasan, 'Generic Text Recognition using Long Short-Term Memory Networks', p. 179.

[37] R. Smith, 'Tesseract Blends Old and New OCR Technology - DAS2016 Tutorial - Architecture and Data Structures - A quick tour of the Tesseract Code', presented at the DAS2016 Tutorial, Santorini, Greece, 2016.

[38] Y. Du *et al.*, 'PP-OCR: A Practical Ultra Lightweight OCR System', *ArXiv200909941 Cs*, Oct. 2020, Accessed: Apr. 29, 2022. [Online]. Available: http://arxiv.org/abs/2009.09941

[39] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, 'Real-time Scene Text Detection with Differentiable Binarization'. arXiv, Dec. 03, 2019. Accessed: May 20, 2022. [Online]. Available: http://arxiv.org/abs/1911.08947

[40] A. Howard *et al.*, 'Searching for MobileNetV3'. arXiv, Nov. 20, 2019. Accessed: May 20, 2022. [Online]. Available: http://arxiv.org/abs/1905.02244

[41] B. Shi, X. Bai, and C. Yao, 'An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition'. arXiv, Jul. 21, 2015. Accessed: May 20, 2022. [Online]. Available: http://arxiv.org/abs/1507.05717

[42] *EasyOCR*. Jaided AI, 2022. Accessed: May 20, 2022. [Online]. Available: https://github.com/JaidedAI/EasyOCR

[43] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, 'Character Region Awareness for Text Detection'. arXiv, Apr. 03, 2019. Accessed: May 20, 2022. [Online]. Available: http://arxiv.org/abs/1904.01941

[44] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. arXiv, Apr. 10, 2015. Accessed: May 20, 2022. [Online]. Available: http://arxiv.org/abs/1409.1556

[45] 'Introduction - Robust Reading Competition'. https://rrc.cvc.uab.es/ (accessed May 18, 2022).

[46] 'Tasks - ICDAR 2019 Robust Reading Challenge on Scanned Receipts OCR and Information Extraction - Robust Reading Competition'. https://rrc.cvc.uab.es/?ch=13&com=tasks (accessed Apr. 28, 2022).

[47] Z. Huang *et al.*, 'ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction', *2019 Int. Conf. Doc. Anal. Recognit. ICDAR*, pp. 1516–1520, Sep. 2019, doi: 10.1109/ICDAR.2019.00244.

[48] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, 'LayoutLM: Pre-training of Text and Layout for Document Image Understanding', in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020, pp. 1192–1200. doi: 10.1145/3394486.3403172.

[49] Y. Li *et al.*, 'StrucTexT: Structured Text Understanding with Multi-Modal Transformers'. arXiv, Nov. 08, 2021. Accessed: May 18, 2022. [Online]. Available: http://arxiv.org/abs/2108.02923

[50] P. Resnick, 'Internet Message Format - RFC 5322'. IETF doi: 10.17487/RFC5322. Accessed: May 18, 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc5322

[51] 'How to Find or Validate an Email Address'. https://www.regular-expressions.info/email.html (accessed May 18, 2022).

[52] Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. V. Jagadish, 'Regular expression learning for information extraction', in *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*, Honolulu, Hawaii, 2008, p. 21. doi: 10.3115/1613715.1613719.

[53] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, 'Neural Architectures for Named Entity Recognition'. arXiv, Apr. 07, 2016. Accessed: May 18, 2022. [Online]. Available: http://arxiv.org/abs/1603.01360

[54] X. Zhao, E. Niu, Z. Wu, and X. Wang, 'CUTIE: Learning to Understand Documents with Convolutional Universal Text Information Extractor'. arXiv, Jun. 19, 2019. Accessed: May 19, 2022. [Online]. Available: http://arxiv.org/abs/1903.12363

[55] A. R. Katti *et al.*, 'Chargrid: Towards Understanding 2D Documents'. arXiv, Sep. 24, 2018. Accessed: May 19, 2022. [Online]. Available: http://arxiv.org/abs/1809.08799

[56] W. Yu, N. Lu, X. Qi, P. Gong, and R. Xiao, 'PICK: Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks'. arXiv, Jul. 18, 2020. Accessed: May 18, 2022. [Online]. Available: http://arxiv.org/abs/2004.07464

[57] T. Hong, D. Kim, M. Ji, W. Hwang, D. Nam, and S. Park, 'BROS: A Pre-trained Language Model Focusing on Text and Layout for Better Key Information Extraction from Documents'. arXiv, Apr. 05, 2022. Accessed: May 19, 2022. [Online]. Available: http://arxiv.org/abs/2108.04539

[58] X. Liu, F. Gao, Q. Zhang, and H. Zhao, 'Graph Convolution for Multimodal Information Extraction from Visually Rich Documents', *ArXiv190311279 Cs*, Mar. 2019, Accessed: Feb. 23, 2022. [Online]. Available: http://arxiv.org/abs/1903.11279

[59] Z. Zhang, J. Ma, J. Du, L. Wang, and J. Zhang, 'Multimodal Pre-training Based on Graph Attention Network for Document Understanding'. arXiv, Mar. 25, 2022. Accessed: May 18, 2022. [Online]. Available: http://arxiv.org/abs/2203.13530

[60] Y. Xu *et al.*, 'LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding'. Jan. 09, 2022. Accessed: May 19, 2022. [Online]. Available: http://arxiv.org/abs/2012.14740

[61] B. Vrusias, 'Class Lecture, Topic: "Traditional NLP" COM3029, Department of Computer Science, University of Surrey, Guildford', Feb. 17, 2022.

[62] B. Vrusias, 'Class Lecture, Topic: "Language Models" COM3029, Department of Computer Science, University of Surrey, Guildford', Feb. 24, 2022.

[63] M. Das, S. Kamalanathan, and P. J. A. Alphonse, 'A Comparative Study on TF-IDF Feature Weighting Method and its Analysis using Unstructured Dataset'.

[64] 'Class Lecture, Topic: "Word Embeddings" COM3029, Department of Computer Science, University of Surrey, Guildford'.

[65] T. Mikolov, K. Chen, G. Corrado, and J. Dean, 'Efficient Estimation of Word Representations in Vector Space', *ArXiv13013781 Cs*, Sep. 2013, Accessed: May 07, 2022. [Online]. Available: http://arxiv.org/abs/1301.3781

[66] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, 'Enriching Word Vectors with Subword Information', *ArXiv160704606 Cs*, Jun. 2017, Accessed: May 07, 2022. [Online]. Available: http://arxiv.org/abs/1607.04606

[67] J. Pennington, R. Socher, and C. Manning, 'Glove: Global Vectors for Word Representation', in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

[68] E. M. Dharma, 'THE ACCURACY COMPARISON AMONG WORD2VEC, GLOVE, AND FASTTEXT TOWARDS CONVOLUTION NEURAL NETWORK (CNN) TEXT CLASSIFICATION'.

[69] T. Joachims, 'Text categorization with Support Vector Machines: Learning with many relevant features', in *Machine Learning: ECML-98*, vol. 1398, C. Nédellec and C. Rouveirol, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142. doi: 10.1007/BFb0026683.

[70] Y. Kim, 'Convolutional Neural Networks for Sentence Classification'. arXiv, Sep. 02, 2014. Accessed: May 20, 2022. [Online]. Available: http://arxiv.org/abs/1408.5882

[71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. arXiv, May 24, 2019. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/1810.04805

[72] Y. Liu *et al.*, 'RoBERTa: A Robustly Optimized BERT Pretraining Approach'. arXiv, Jul. 26, 2019. Accessed: May 22, 2022. [Online]. Available: http://arxiv.org/abs/1907.11692

[73] '3.10.4 Documentation'. https://docs.python.org/3/ (accessed Apr. 29, 2022).

[74] 'Anaconda | The World's Most Popular Data Science Platform', *Anaconda*. https://www.anaconda.com/ (accessed Apr. 29, 2022).

[75] 'TensorFlow', *TensorFlow*. https://www.tensorflow.org/ (accessed Apr. 29, 2022).

[76] 'Keras: the Python deep learning API'. https://keras.io/ (accessed Apr. 29, 2022).

[77] 'Visual Studio Code - Code Editing. Redefined'. https://code.visualstudio.com/ (accessed Apr. 29, 2022).

[78] 'Project Jupyter'. https://jupyter.org (accessed Apr. 29, 2022).

[79] 'Google Colab'. https://research.google.com/colaboratory/faq.html (accessed Apr. 29, 2022).

[80] 'NVIDIA cuDNN', *NVIDIA Developer*, Sep. 02, 2014. https://developer.nvidia.com/cudnn (accessed Apr. 29, 2022).

[81] S. Park *et al.*, 'CORD: A Consolidated Receipt Dataset for Post-OCR Parsing', *Doc. Intell. Workshop Neural Inf. Process. Syst.*, p. 4.

[82] 'What is Kaggle, Why I Participate, What is the Impact? | Data Science and Machine Learning'. https://www.kaggle.com/getting-started/a (accessed Apr. 29, 2022).

[83] 'my receipts (pdf scans)'. https://www.kaggle.com/jenswalter/receipts (accessed Apr. 29, 2022).

[84] O. Rozen, D. Carmel, A. Mejer, V. Mirkis, and Y. Ziser, 'Answering Product-Questions by Utilizing Questions from Other Contextually Similar Products', in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online, 2021, pp. 242–253. doi: 10.18653/v1/2021.naacl-main.23.

[85] 'Amazon-PQA - Registry of Open Data on AWS'. https://registry.opendata.aws/amazon-pqa/ (accessed Apr. 29, 2022).

[86] 'Hierarchical text classification'. https://www.kaggle.com/kashnitsky/hierarchical-text-classification (accessed Apr. 29, 2022).

[87] 'Product Classification and Clustering'. https://www.kaggle.com/lakritidis/product-classification-and-categorization (accessed Apr. 29, 2022).

[88] L. Akritidis, A. Fevgas, and P. Bozanis, 'Effective Products Categorization with Importance Scores and Morphological Analysis of the Titles', in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, Volos, Nov. 2018, pp. 213–220. doi: 10.1109/ICTAI.2018.00041.

[89] A. Althnian *et al.*, 'Impact of Dataset Size on Classification Performance: An Empirical Evaluation in the Medical Domain', *Appl. Sci.*, vol. 11, no. 2, p. 796, Jan. 2021, doi: 10.3390/app11020796.

[90] 'pytesseract · PyPI'. https://pypi.org/project/pytesseract/ (accessed May 21, 2022).

[91] NielsRogge, *Transformers-Tutorials*. 2022. Accessed: May 23, 2022. [Online]. Available: https://github.com/NielsRogge/Transformers-Tutorials/blob/20844e27dac26042277c8725269ed2a9d57d88f0/LayoutLMv2/CORD/Prepare_CORD_for_LayoutLMv2.ipynb

[92] *facebookresearch/detectron2*. Meta Research, 2022. Accessed: May 23, 2022. [Online]. Available: https://github.com/facebookresearch/detectron2

[93] 'ETHICS | Meaning & Definition for UK English | Lexico.com', *Lexico Dictionaries | English*. https://www.lexico.com/definition/ethics (accessed May 08, 2022).

[94] 'Apache License, Version 2.0'. https://www.apache.org/licenses/LICENSE-2.0 (accessed May 21, 2022).

[95] 'UK GDPR Updated for Brexit | UK GDPR'. https://uk-gdpr.org/ (accessed May 21, 2022).

[96] 'What is personal data?', Jan. 01, 2021. https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/key-definitions/what-is-personal-data/ (accessed May 21, 2022).

Appendices

Appendix A – Personal Finances & Digital Apps Survey Responses

Forms ? GL

Personal Finances & Digital Apps

63	03:03	Active
Responses	Average time to complete	Status

1. What age range are you in?

18-24 years old	59
25-34 years old	4
35-44 years old	0
45-54 years old	0
55+ years old	0

2. Are you a student?

Yes	42
No	21

3. Do you use online banking? (either through a browser or a mobile app)

Yes	63
No	0

4. Have you ever used a digital bank? (I.e. Starling, Monzo, Revolut etc)

Yes; my main bank is a digital ...	11
Yes; but not as my main bank ...	27
No	25

5. What do you like about digital banking?

35

Responses

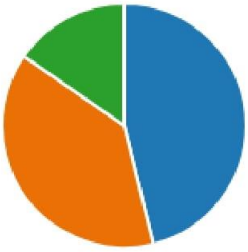
Latest Responses

"The ease of money transfer and seeing how much you have "

15 respondents (**43**%) answered **Easy** for this question.

6. Would you consider using a digital bank?

Yes	12
Yes, but not for my main bank	10
No	4



7. If no or not for your main bank, how come?

14
Responses

Latest Responses

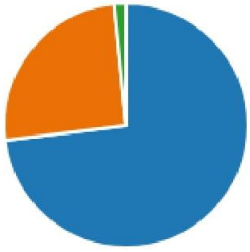
"Don't like the fact I can not walk into a building and speak to someone w..."

"Don't understand it's credibility enough "

3 respondents (**21**%) answered **person** for this question.

8. Have you ever been stressed, anxious, or worried about your personal finances?

Yes	46
No	16
Prefer not to say	1



9. Have you ever been stressed, anxious, or worried about your personal finances?

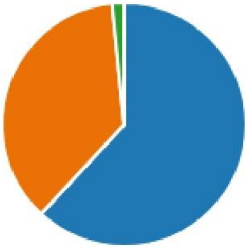
Yes	0
No	0
Prefer not to say	0

10. Do you believe that utilising a budgeting app would help lessen these worries?

Yes	28
No	6
Not sure	12

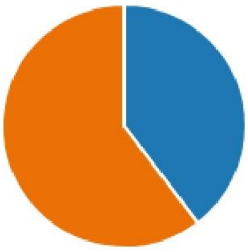
11. Could you last 3 months or more if you lost your main source of income? (I.e. could you live off your savings for 3 months or more)

Yes	39
No	23
Prefer not to say	1



12. Do you know where to go for advice on budgeting your personal finances?

Yes	25
No	38



13. How confident do you feel about making financial decisions? (with 1 being not confident at all and 10 being extremely confident)

63
Responses

6.79
Average Number

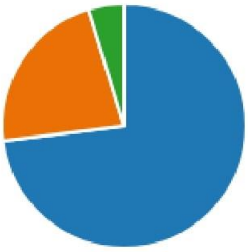
14. How important do you think managing your spending is? (with 1 being not important at all and 10 being extremely important)

63
Responses

9.1
Average Number

15. Do you budget your spending? (in any capacity; this could be on a purchase-by-purchase basis or even a 5 year plan)

Yes	46
No	14
Other	3

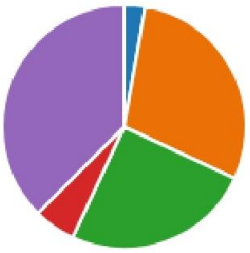


16. How do you plan your budgeting?

Mobile applications	14
Manually/On paper/By hand	30
Other	10

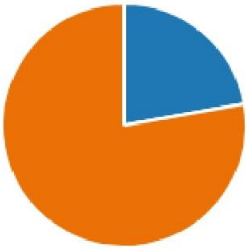
17. What is stopping you from budgeting your spending?

Budgeting seems pointless	2
Too much hassle	21
Too stressful	18
I don't need to budget	4
Other	27



18. Have you ever used a budgeting app to help with your finances?

Yes	14
No	49



19. How helpful was the app in helping you manage your finances? (with 1 being not helpful at all and 10 being extremely helpful)

14	5.64
Responses	Average Number

20. What is the reason for the score you gave?

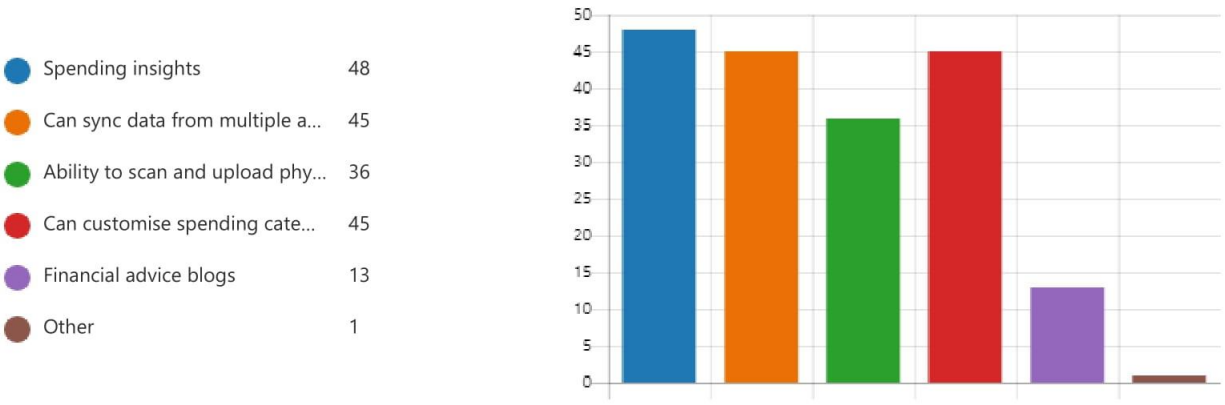
10
Responses

Latest Responses
"Unhelpful they kept suggesting just making more money lol "

3 respondents (**30**%) answered **expenses** for this question.

21. Would you be interested in using a mobile application to help you manage your finances?

22. Select the top 3 features that you would like to see in a budgeting app:

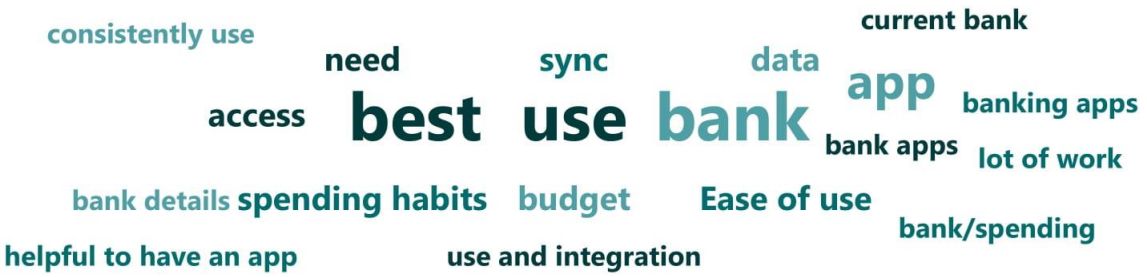


23. What are your reservations, if any, regarding a mobile budgeting app?

20
Responses

Latest Responses

4 respondents (**20**%) answered **use** for this question.



SAGE-HDR (v3.4 19/05/22)

Response ID	Completion date
899668-899650-95415268	24 May 2022, 15:04 (BST)

1	**Applicant Name**	Grace Yingtong Lin
1.a	**University of Surrey email address**	gl00233@surrey.ac.uk
1.b	**Level of research**	Undergraduate
1.b.i	**Please enter your University of Surrey supervisor's name. If you have more than one supervisor, enter the details of the individual who will check this submission.**	Helen Treharne
1.b.ii	**Please enter your supervisor's University of Surrey email address. If you have more than one supervisor, enter the details of the supervisor who will check this submission.**	h.treharne@surrey.ac.uk
1.c	**School or Department**	Computer Science
1.d	**Faculty**	FEPS - Faculty of Engineering and Physical Sciences

2	**Project title**	Automatic Product Extraction, Classification, and Analysis of Receipt Data

3	**Please enter a brief summary of your project and its methodology in 250 words. Please include information such as your research method/s, sample, where your research will be conducted and an overview of the aims and objectives of your research.**	This project seeks to combine research from both fields (OCR and NLP) into a method to extract key information from scanned receipts and invoices and provide insight and analysis into these documents. This work will focus on data from personal sales receipts, rather than in a business context, which is more typical of similar works in this space. Therefore, this work aims to extract an itemised purchase list from a receipt and classify those items into relevant categories. By comparing the latest methods in relevant machine learning research, this project proposes a proof-of-concept for an automated pipeline to extract, classify, and analyse product data from receipts and invoices. A custom dataset of receipts has been collected for this project, with voluntary contributions from others. A budgeting survey was also carried out to gauge the usefulness of an app to help with accounting from extracting physical receipt data.

4	**Are you planning to join on to an existing Standard Study Protocol (SSP)? SSPs are overarching pre-approved protocols that can be used by multiple researchers investigating a similar topic area using identical methodologies. Please note, SSPs are only being used by one school currently and cannot be used by other schools. Using an SSP requires permission and sign-off from the SSP owner**	NO

5	**Are you making an amendment to a project with a current University of Surrey favourable ethical opinion or approval in place?**	NO

6	**Does your research involve any animals, animal data or animal derived tissue, including cell lines?**	NO

8	**Does your project involve human participants (including human data and/or any human tissue*)?**	YES

9	**Will you be accessing any organisations, facilities or areas that may require prior permission? This includes organisations such as schools (Headteacher authorisation), care homes (manager permission), military facilities, closed online forums, private social media pages etc. If you are unsure, please contact ethics@surrey.ac.uk.**	NO

10	**Does your project involve any type of human tissue research? This includes Human Tissue Authority (HTA) relevant, or non-relevant tissue (e.g. non-cellular such as plasma or serum), any genetic material, samples that have been previously collected, samples being collected directly from the donor or obtained from another researcher, organisation or commercial source.**	NO

11	**Does your research involve exposure of participants to any hazardous materials e.g. chemicals, pathogens, biological agents or does it involve any activities or locations that may pose a risk of harm to the researcher or participant?**	NO

12	**Will you be importing or exporting any samples (including human, animal, plant or microbial/pathogen samples) to or from the UK?**	NO

13	**Will any participant visits be taking place in the Clinical Research Building (CRB)? (involving clinical procedures; if only visiting the CRB to collect/drop-off equipment or to meet with the research team (i.e. for informed consent/discussion) select 'NO').**	NO

14	**Will you be working with any collaborators or third parties to deliver any aspect of the research project?**	NO

15	**Are you conducting a service evaluation or an audit? Or using data from a service evaluation or audit?**	NO

16	**Does your funder, collaborator or other stakeholder require a mandatory ethics review to take place at the University of Surrey?**	NO

17	**Does your research involve accessing students' results or performance data? For example, accessing SITS data.**	NO

18	**Will ANY research activity take place outside of the UK?**	NO

19	**Are you undertaking security-sensitive research, as defined in the text below?**	NO

20	**Does your project require the processing of special category1 data?**	NO

21	**Have you selected YES to one or more of the above governance risk questions on this page (Q10-Q20)?**	NO

22	**Does your project process personal data2? Processing covers any activity performed with personal data, whether digitally or using other formats, and includes contacting, collecting, recording, organising, viewing, structuring, storing, adapting, transferring, altering, retrieving, consulting, marketing, using, disclosing, transmitting, communicating, disseminating, making available, aligning, analysing, combining, restricting, erasing, archiving, destroying.**	NO

23	**Are you using a platform, system or server external to the University approved platforms (Outside of Microsoft Office programs, Sharepoint or OneDrive)?**	NO

24	**Does your research involve any of the above statements? If yes, your study may require external ethical review or regulatory approval**	NO

25	**Does your research involve any of the above? If yes, your study may require external ethical review or regulatory approval**	NO

26	**Does your project require ethics review from another institution? (For example: collaborative research with the NHS REC, the Ministry of Defence, the Ministry of Justice and/or other universities in the UK or abroad)**	NO

27	**Does your research involve any of the following individuals or higher-risk methodologies? Select all that apply or select 'not applicable' if no options apply to your research. Please note: the UEC reviewers may deem the nature of the research of certain high risk projects unsuitable to be undertaken by undergraduate students**	NOT APPLICABLE - none of the above high-risk options apply to my research.

28	**Does your research involve any of the following individuals or medium-risk methodologies? Select all that apply or select 'not applicable' if no options apply to your research.**	NOT APPLICABLE - none of the above medium-risk options apply to my research.

29	**Does your research involve any of the following individuals or lower-risk methodologies? Select all that apply or select 'not applicable' if no options apply to your research.**	NOT APPLICABLE - none of the above lower-risk options apply to my research.

30	**Declarations**	• I confirm that I have read the University's Code on Good Research Practice and ethics policy and all relevant professional and regulatory guidelines applicable to my research and that I will conduct my research in accordance with these. • I confirm that I have provided accurate and complete information regarding my research project • I understand that a false declaration or providing misleading information will be considered potential research misconduct resulting in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies • I understand that if my answers to this form have indicated that I must submit an ethics and governance application, that I will NOT commence my research until a Favourable Ethical Opinion is issued and governance checks are cleared. If I do so, this will be considered research misconduct and result in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies. • I understand that if I have selected 'YES' on any governance risk questions and/or have selected any options on the higher, medium or lower risk criteria then I MUST submit an ethics and governance application (EGA) for review before conducting any research. If I have NOT selected any governance risks or selected any of the higher, medium or lower ethical risk criteria, I understand I can proceed with my research without review and

		acknowledge that my SAGE answers and research project will be subject to audit and inspection by the RIGO team at a later date to check compliance.

31	**If I am conducting research as a student:**	• I confirm that I have discussed my responses to the questions on this form with my supervisor to ensure they are correct. • I confirm that if I am handling any information that can identify people, such as names, email addresses or audio/video recordings and images, I will adhere to the security requirements set out in the relevant Data Protection Policy