@HBS > An app-assisted approach for the Household Budget Survey

Optical Character Recognition and Machine Learning Classification of Shopping Receipts

Lanthao Benedikt[1], Chaitanya Joshi[1], Louisa Nolan[1], Nick de Wolf[2], and Barry Schouten[2]

[1]Data Science Campus, Office for National Statistics, United Kingdom
[2]Division of Methodology and Quality, Statistics Netherlands, The Netherlands

project number ESSnet SEP-2105369
February 28, 2020

Abstract

This chapter covers part 2 of Work package 4 in the @HBS project and deals with the processing of shopping receipts. Relevant information such as shop names, dates, purchased items and prices are extracted from receipts and products are classified to their 5-digit Classification of Individual Consumption by Purpose (COICOP). Currently, this is done manually in most countries, which requires several hours to process one single diary and large teams of coders are needed to complete the tasks. We demonstrate how data science techniques and Human-in-the-Loop AI can be applied to automate this process, the aim is time and resource saving on repetitive, labour-intensive tasks which machines are good at, allowing humans to focus on value added tasks requiring flexibility and intelligence. The proposed solution is developed in the context of the United Kingdom, we discuss how the methods can be extended for other countries. Our aim is to make our methodology and codes freely available so that any country can reuse and modify our work to suit their specific requirements. We report not only methods that show potential but also preliminary exploration and failed attempts in the hope that it will help other countries avoid pitfalls.

Acknowledgments

The authors would like to thank Joanna Bulman who leads the UK Living Costs and Food Survey for establishing initial contacts with the @HBS task force and for coordinating Work Package 4.2 with our @HBS partners over the last year. A big thanks to Jo's team, in particular Sharon Hook and Aleksandra Pastuszak, to Andy Watson in the Blaise team and Di Williams' coding team for your domain knowledge inputs and for all the hard work collecting data for our research over the last year. We also would like to acknowledge the support of senior management at the Social Survey Operation Division - Chris Daffin and Alex Lambert - for making this research possible.

We very much appreciate helpful inputs and suggestions from other data scientists at the ONS Data Science Campus, Philip Lee, Arturas Eidukas, Li Chen, Ian Grimstead, Philip Stubbibgs, Ruben Henstra-Hill, Stuart Newcombe, Luke Shaw, Gareth Jones and others. We are very grateful for colleagues at the Campus who kindly donated their personal receipts for our research. We would like to thank Sharon Hill and the Campus Delivery team - Kate Milligan, Lucy Inker-Davies and Wenda Powell - for helping to organise so many project meetings, stand-ups and for managing the project Github repository. We also would like to thank Tom Smith - our Campus Managing Director - and the Campus Project Board for allowing this project to go forward and for your support all along.

Last but not least, we would like to thank our external partners, in addition to our co-authors Barry Schouten and Nick de Wolf at CBS. Over the last year, we have been able to build very strong relationships with our @HBS European partners and beyond. In particular, we have learned much from experiences at the Irish Central Statistical Office and at Statistics Canada. A special thanks to colleagues at StatCan - Emilie Mayer, Denis Malo, Johanne Tremblay, Francois Brisebois, Patrick Gallifa, Monica Pickard, Christian Ritter and many others from the Methodology Division and the Data Science Division. We very much appreciate your interest in our research and for sharing knowledge with us. Our brainstorming meetings have always been very thought provoking, we hope to strengthen this collaboration in the future and expanding it to new partners.

It has been a thoroughly enjoyable and productive experience working with you all. We hope to continue to work together in the future to further share knowledge and lessons learned.

Contents

1 Project overview

1.1 Introduction

The Household Budget Survey (HBS) is the generic name of a survey that is conducted in almost every country in the world, with varying names such as the UK Living Costs and Food Survey (LCF) or the Canadian Survey of Household Spending (SHS). It collects data on household incomes and spending patterns, which provides crucial information for the estimates of the country's Gross Domestic Products (GDP) and Price indices. In many countries, it also provides key indicators on nutrition and food consumption for Health and Environmental Departments.

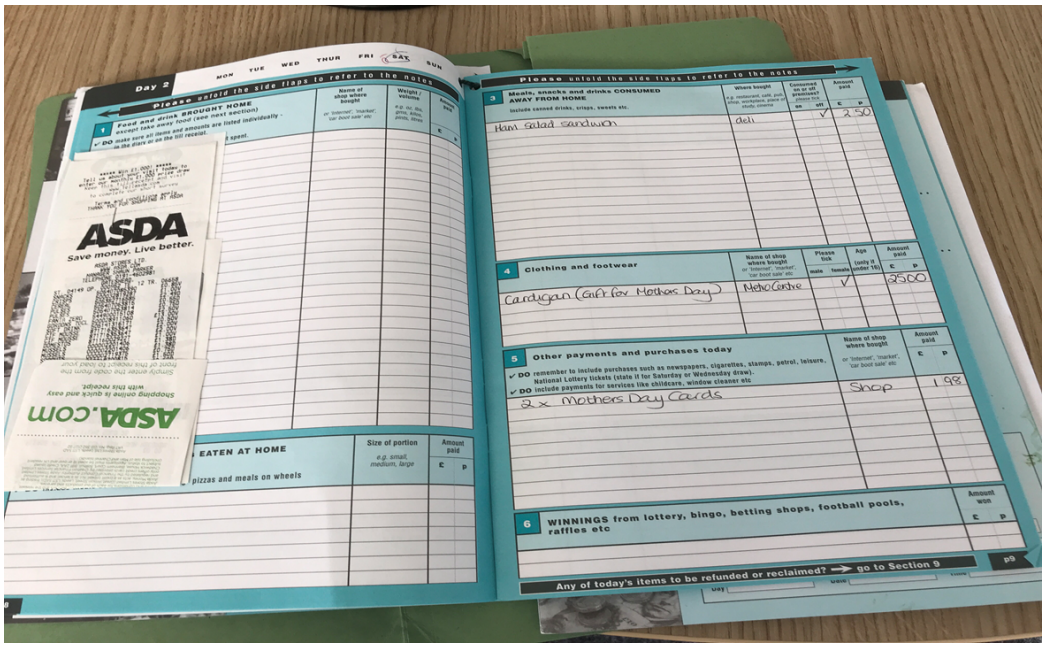


Figure 1: The UK Living Costs and Food Survey (LCF) diary: *purchase descriptions*, *prices*, *shop names* and *dates* need to be recorded over a 2-week period.

In a household selected for the survey, everyone keeps a diary of expenditure over typically 2 weeks. Information such as *purchase descriptions*, *prices*, *shop names* and *dates* need to be recorded. An example of the LCF diary is shown in Figure 1. To ease respondent burden, government agencies usually collect shopping receipts. Then, back in the office, a team of coders manually type relevant information from the receipts into the system and manually classify each purchased item to a 5-digit coding frame. Table 1 shows an example of milk products and their corresponding Classification of Individual Consumption According to Purpose (COICOP) codes.

It takes several hours to manually process one single diary and a large team of coders is needed to complete the task. Stringent government budget cuts over the years have placed tremendous emphasis on the need to make efficiency savings, hence the incentive for exploring modern technologies and Artificial Intelligence (AI) to automate manual operations. Beyond the technical challenge, the introduction of AI also raises a number of important questions: How does automation affect output quality? How can we define and measure success? How does it change the ways we work?

In this report, we describe an automation pipeline and implement a proof of concept to demonstrate how governments can modernise their HBS data collection process. We discuss how success in terms of efficiency savings, processing time and data quality can be formally defined and quantitatively

Product category	COICOP
Pasteurised and homogenised whole milk	1.1.4.1.1
Sterilised whole milk	1.1.4.1.2
UHT whole milk	1.1.4.1.3
School milk	1.1.4.1.4
Welfare milk	1.1.4.1.5
Skimmed milk - incl UHT and sterilised	1.1.4.2.1
Semi-skimmed milk - incl UHT and sterilised	1.1.4.2.2

Table 1: Example of milk products and their corresponding Classification of Individual Consumption According to Purpose (COICOP) codes.

measured. We make our methods and codes publicly available so that any country can reuse and modify our work to suit their specific requirements. The report is organised as follows:

- `Section 1: Project overview` - We describes the current typical HBS process and highlight the need for improvement. We explain the scope of Work package 4.2 in the context of the wider ESSNet @HBS project and set the goals and deliverables for our research.
- `Section 2: Design of the automation pipeline` - We examine how current HBS manual processes can be translated into automated processes and posit that Human-in-the-Loop AI helps make efficiency savings, speeding up processing time whilst maintaining similar or better data quality compared to manual processing. The proposed automation pipeline consists of the following modules: Receipt scanning, Image processing, Optical Character Recognition and Machine Learning classification.
- `Section 3: Receipt scanning` - We discuss pros and cons of various scanning methods and examine parameters that can negatively affect the quality of scanned images.
- `Section 4: Image processing` - We investigate the challenges for obtaining good quality images and describe image processing methods that need to be applied.
- `Section 5: Optical Character Recognition` - We briefly discuss how to select a suitable OCR tool and explain the challenge of how to extract relevant information from raw OCR outputs. We explore two approaches for data parsing and develop an automated procedure to measure OCR performance.
- `Section 6: Machine Learning classification` - This is currently one of the hottest research topics in data science. Do state-of-the-art algorithms perform better than more traditional models? We test and compare a number of popular algorithms.
- `Section 7: Measuring success` - Data scientists measure success in terms of accuracy, precision, F-scores. Such quantities are not meaningful from a practical business viewpoint. We propose to formally define quantitative measures of success in terms of efficiency savings, processing time, data quality and report current test results.
- `Section 8: User interface` - We define the user stories and mockup a User Interface to demonstrate how the pipeline can be implemented.
- `Section 9: Conclusion and Future works` - We summarise the current results and layout the directions for future researches.

1.2 The ESSnet @HBS Project

Modernising the HBS data collection process is an immense task that provides the opportunity for collaborations between National Statistical Institutes. One such joint effort is the ESSNet @HBS

project led by Statistics Netherlands (CBS), which includes Statistics Finland, Statistics Austria, Statistics Slovenia, the UK Office for National Statistics (ONS) and the University of Essex. The project investigates the entire end-to-end data collection process and combines four areas of expertise and methodology as follows:

- `Work package 1 - Coordination`: Provide feedback to Eurostat coordinators and the task force HBS.
- `Work package 2 - App design`: This is the main work package in which the app-assisted approach is developed and the corresponding back-end is specified.
- `Work package 3 - Recruitment and consent strategies`: Review, evaluate and test promising recruitment and data collection strategies for an app-assisted approach.
- `Work package 4 - Data analysis`: This work package includes 2 sub-tasks: 1 - Explore and test the potential of linkage of relevant big data sources (CBS led) and 2 - Develop a proof of concept for a system to process scanned receipts, develop Optical Character Recognition, and automated coding (ONS led).

This document reports findings related to the second sub-task of Work package 4, a.k.a WP 4.2: scanning and image processing of receipts, Optical Character Recognition and automated coding. The work carried out by data scientists at the ONS Data Science Campus was funded by Her Majesty's Treasury, United Kingdom. The ONS team worked in collaboration with an image processing expert from CBS who was funded by Eurostat.

1.3 Objectives and deliverables of Work Package 4.2

The primary objective of WP 4.2 is to automate the manual processing of shopping receipts to make efficiency savings, speeding up processing time whilst maintaining similar or better data quality compared to human performance. In practical terms, we aim to solve the following technical problem: given a large collection of paper receipts from survey respondents, how do we automatically extract relevant information into digital format and automatically classify purchased goods into a 5-digit coding frame? Relevant information to extract from receipts includes purchase descriptions, UPC barcodes, prices, shop names, dates, payment modes.

To answer this question, we propose to design an automation pipeline that comprises the following steps: 1 - Scanning of paper receipts, 2 - Image processing, 3 - Optical Character Recognition (OCR), 4 - Natural Language Processing (NLP) and 5 - Machine Learning classification. We build a proof of concept of this pipeline and benchmark its performance against the legacy system in terms of processing speed, output quality and efficiency savings.

Although this research is carried out in the context of the UK, we aim to develop methods that could be adapted and reused by other countries with minimal modifications. We will discuss how this can be done. As different countries have different constraints and strategies, we do not make recommendations for a one-size-fit-all solution but instead, we explore options to help inform decisions. We report on methods that show potential as well as preliminary researches and failed attempts in the hope that it will help other countries avoid pitfalls. All the methodology and Python codes will be made publicly available. The two main deliverables of the project are:

- The present report in which we propose a high level automation pipeline based on the concept of Human-in-the-Loop AI and we explore various methods for implementing a proof of concept. The primary goal for replacing manual process with automation is to make efficiency savings and speed up processing time without degrading data quality and/or increasing respondent burden, we define methods to quantify and measure success. To the best of our knowledge, no

government agency has so far built such a solution using free open-source software. We hope that our work will pave the way for wider collaborations and help harmonise the production of official statistics across countries.

- The Python codes we have developed for this research will be made available on the ONS Data Science Campus Github repository. Since this work is a proof of concept, the current version of the code is not production ready and may require further software engineering. For example, we have not implemented any exception handling and our code has not been refactored to production standards. However, we have ensured that the codes are thoroughly commented so that other government agencies can easily adapt these to suit their specific needs.

2 Design of the automation pipeline

2.1 State of the art in other National Statistical Institutes

Prior to this work, a preliminary literature review was conducted to discover if other government agencies have undertaken similar works. Far from being exhaustive, this review nevertheless provides insights into the current situation. Like the UK, many countries are still collecting and processing HBS data manually. To the best of our knowledge, no country has so far built an end-to-end automation pipeline from receipt scanning to text classification using free open-source software. Countries that are most advanced in the field typically use commercial software to OCR receipts and diaries such as Sweden (EFLOW), Finland (KOFAX) and Ireland (Teleform), as summarised in Figure 2. Whilst OCR of diaries usually works very well, OCR of receipts is much more challenging. Coding of products to COICOP is typically done using a dictionary, we have not found evidence of Machine Learning classification being used in production.

Why is OCR of receipts so difficult?

OCR typically recognises text from images and output blocks of text, together with the coordinates of their bounding boxes. For documents that are formatted in a standardised way, knowing the location of the text in the document is sufficient to infer the meta-data (i.e. to determine if a block of text is a name, an address, a date of birth, etc.). Thus, government agencies have been using OCR to process survey forms for decades and OCR as a field of research is considered by many as a solved problem. OCR of receipts is difficult due to their variety. Receipts are not standardised, which makes it extremely difficult to infer the meta-data. We are able to extract raw text from images of receipts, but we cannot tell what blocks of text are the dates, the items' descriptions, the prices, the shop names and so on. This requires further data parsing to infer the meta-data. Furthermore, unlike survey diaries that are usually of good quality, receipts may be of bad quality (e.g. faded, crumpled, torn, low contrast), which requires image enhancement to be applied prior to OCR. Developing data parsing methods that should work for any receipt, from any shop, in any country and any language is technically challenging.

Buy or build?

To replace their legacy HBS data collection systems, government agencies have two choices: Buy or Build? Either buy a commercial software or build an in-house solution. Both have pros and cons. Disadvantages of buying commercial software include but are not limited to:

- `Intellectual Property (IPs)`: The software vendors own the IPs. The users are not aware of the underlying methods, which makes it difficult to share knowledge with other agencies. There is a risk that agencies will work in silos using software from different vendors, hence it is difficult to harmonise methodology across countries and strengthen collaborations.

Country	When	Software	Process	Lessons learned
Sweden	Since 2012	Customised version of EFLOW	• Scanning both diaries and receipts in-house • Diaries formatted for scanning • No information on data editing and coding	• Diaries scanning works well • Receipts scanning problematic due to receipt quality and variety of formats
Finland	Since 2016	KOFAX + in-house software for manual editing	• Scanning and coding in-house • Only scan long receipts with ≥ 3 items • Bespoke solution for manual editing	• 80% of data was correctly extracted from the receipts • 20% requiring manual entry (short receipts and written data)
Ireland	Since 2014	Customised version of Teleform	• Scanning and OCR in-house • Manual editing using customised Teleform interface • Coding using dictionary	• Report no significant problems with receipt scanning • Report no significant problems with automated coding
The Netherlands	Explore solution in 2013	Blaise	• Digital diary in Blaise • Receipt scanning by respondents • Data editing with Blaise/Manipula • Automatic COICOP classification	• 50% receipts correctly scanned, of which 75% correct prices extracted • 85% correct COICOP classifications • OCR poor performance, risk of increasing respondent burden • Conclusion: scanning was not a sustainable option

Figure 2: Diary and receipts scanning in some countries. Note: here, *'in-house'* scanning means collecting paper receipts and then scan them back at the office using a flatbed scanner, in contrast to *'app-scanning'* which means the respondent uses the mobile app to make a picture of receipts.

- `Hidden costs`: No off-the-shelf software initially matches all requirements. Agencies depend on the vendors to develop additional features and to implement future improvements. Agencies also depend on the vendors for support and maintenance.
- `Control`: If the vendors cease to exist or change their terms and conditions such that they are no longer suitable, the agencies may suffer interruption of services.

Despite such limitations, buying commercial software could nevertheless be attractive because it requires reasonable investment to obtain a working solution. Commercial software are robust because vendors have specialist teams and many years of experience in software development, compared to government agencies' in-house teams who start from scratch. Figures 3 and 4 show screenshots of the commercial software used at the Irish Central Statistics Office (CSO). OCR is performed with the commercial software Teleform developed by OpenText and coding is done using a dictionary. It took only 9 months and a small team to put in place the new system back in 2014. Data processing time and resource were reportedly divided by half. In contrast, an agency that wishes to build an in-house solution needs to recruit data scientists to build the software and train 'intelligent users' so they can operate and maintain the new system, which represents a significant investment.

In this research, we propose to use the CSO system as a starting point and explore how we can build a similar solution using exclusively free standard open-source software packages and well-established data science techniques.

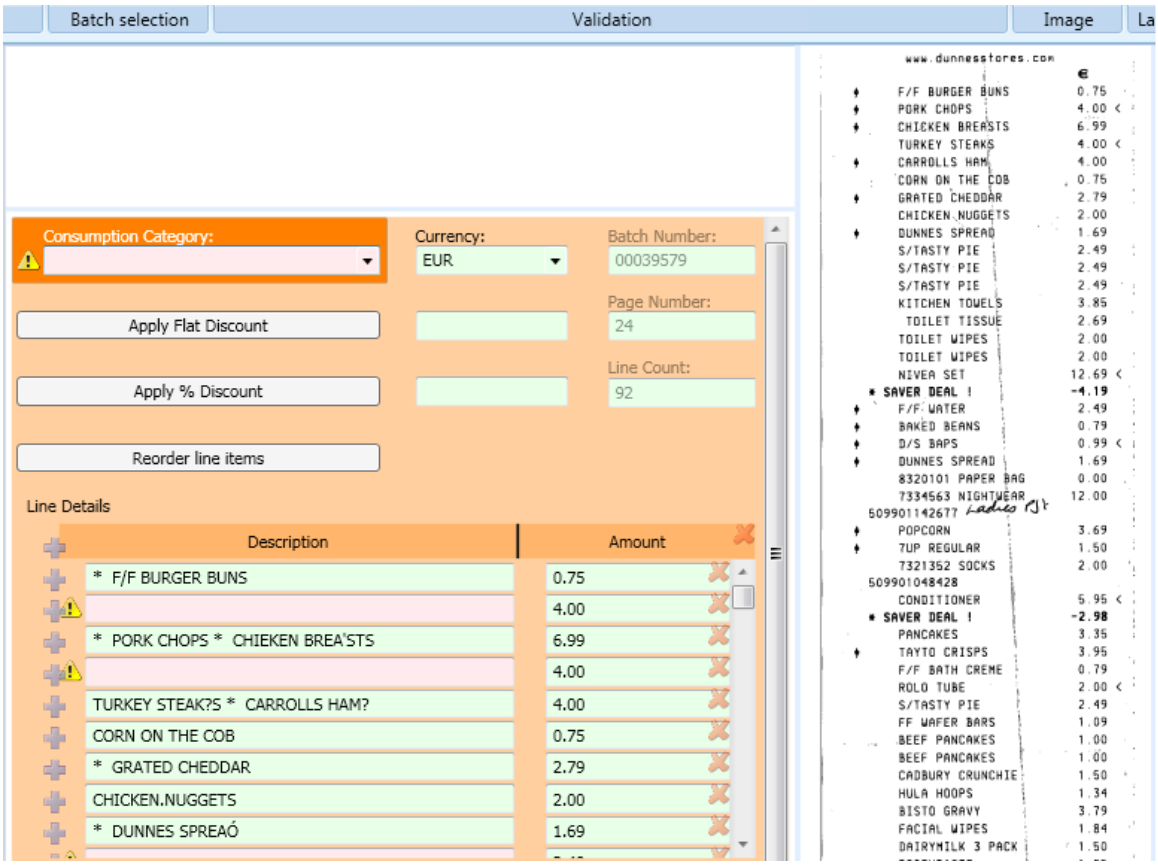


Figure 3: Irish Central Statistics Office HBS data capture solution using OCC and OpenText Teleform: OCR screen.

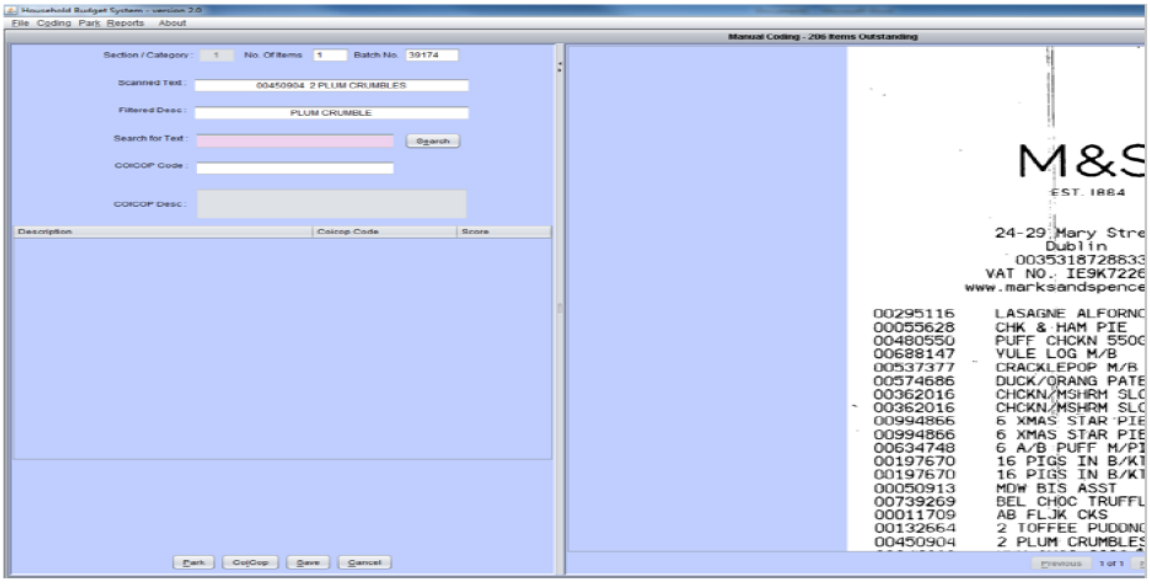


Figure 4: Irish Central Statistics Office HBS data capture system: coding screen.

2.2 The pipeline

We developed the high-level automation pipeline by observing the UK LCF manual coding process, as shown in Figure 5.

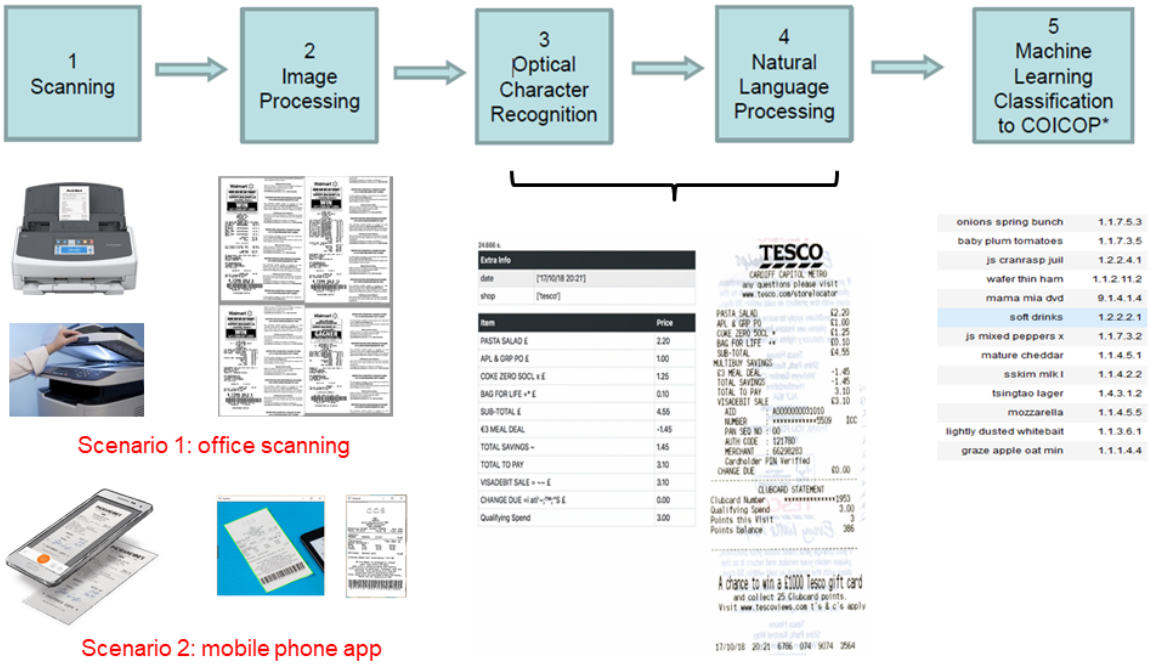


Figure 5: Automation pipeline to replace the legacy data collection system for the HBS. The workflow can be broken down into 5 steps: 1) Receipt scanning, 2) Image processing, 3) OCR, 4) Natural Language Processing and 5) Machine Learning classification. There are two options for receipt scanning: using office scanner or mobile phone app.

1. `Scanning`: paper receipts are scanned into images. There are two options. Option 1 - Paper receipts are brought back to the office and scanned with a flatbed scanner. Option 2 - Respondents capture images using a mobile phone app and send to ONS via Cloud.
2. `Image processing`: receipts are cropped from the scans and image enhancement applied to improve contrast and remove noise. Typical image problems include photos of very bad quality, faded receipts where image contrast needs to be enhanced, stained or crumpled receipts that have shadows on the background, poor lighting, etc.
3. `Optical Character Recognition (OCR)`: text is automatically extracted from the receipts. Data parsing is applied to infer the meta-data such that relevant information can be retrieved.
4. `Natural Language Processing (NLP)`: OCR output may contain misspelled words due to characters being wrongly recognised. One possible way to correct such errors is to apply NLP. This module is explicitely listed for completeness, but the feature is in reality embedded in both the OCR and the classification modules.
5. `Automated classification`: Supervised Machine Learning(ML) models are used to automatically classify items to COICOP codes. Evidence shows that in most ML classification problems, it takes little effort to achieve close to 80% accuracy, but it is increasingly difficult to push for the last 20%. This is a significant challenge for official statistics that require high precision and accuracy. Acceptable error rates are usually agreed between survey teams and their end users, typically less than 5%.

2.3 Human in the Loop AI

There are many situations where automation is difficult if not impossible. For example, item descriptions on receipts can be sometimes very succinct (e.g. *'fresh milk'*, *'bread'*), thus, it does not provide ML models with sufficient information to predict the correct class: is it *whole milk*? *skimmed milk*? or *semi-skimmed milk*? There are also rare items or unseen items (e.g. new products), for which the models will struggle to make correct predictions. This problem is similar to those encountered in some well-known real-world applications such as online photo-tagging, helpdesk chatbot, self-driving cars, etc. Indeed, whilst an autonomous vehicle can drive on a familiar road with little input from the driver, it may not respond well to unseen circumstances such as blocked roads or weather conditions. When this happens, the car hands-over the control to the driver. The controls in an autonomous vehicle are distributed between the car (machine) and the driver (human) using Human-in-the-loop (HuIL), an AI paradigm that relies on human machine interaction [Gil et al. (2016), Faith (2008), Rothrock and Narayanan (2011), R. and Thomas (2000) and Wenchao et al. (2014)]. We propose to adapt HuIL concept to the present application as shown in Figures 6.

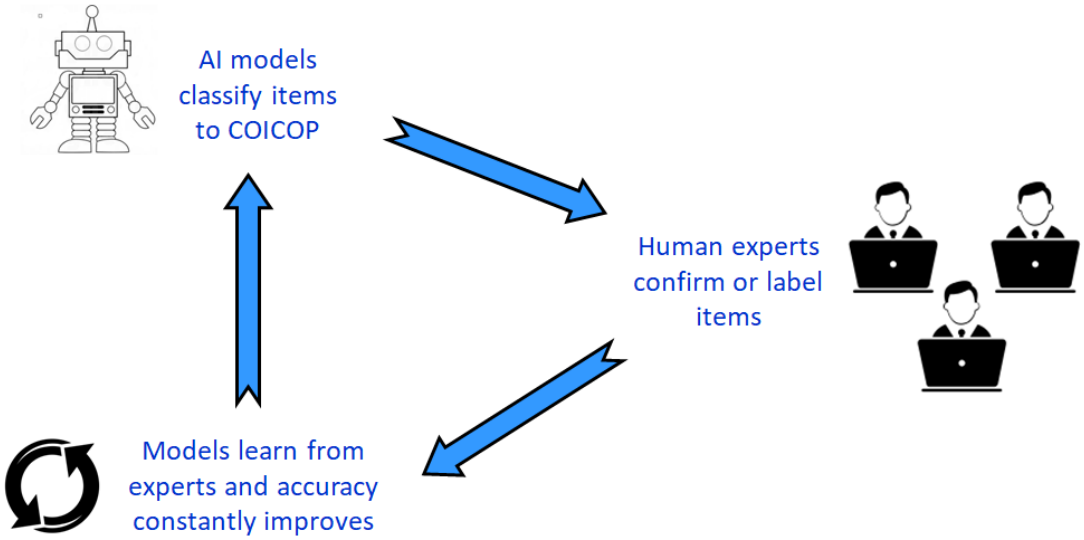


Figure 6: Human-in-the-Loop AI in a human-expertise-centric process. Machine classifies items to COICOP, it performs the tasks very quickly and consistently but will make some mistakes. For example, if the item has not been seen before (i.e. not in the training dataset), the ML model will struggle to identify the correct class. In this case, machine alerts human who then steps in to assign the correct label. The new labelled item is used to retrain the model to make it more up-to-date. Over time, machine learns from human experts and becomes more and more accurate.

The idea of HuIL is to acknowledge that both machine and human have strengths and weaknesses, and it's their pairing – not the supremacy of one over the other – that yields the best results (Lukas Bievald, CEO of CrowdFlower). Indeed, machines are consistent and incredibly fast, but cannot make good judgments in unfamiliar situations. Comparatively, humans are inconsistent and slow, but intelligent and adaptable. HuIL is a branch of AI that brings together the best of both worlds. The advantage is time and resource saving on repetitive, labour-intensive tasks which machine is good at, allowing human to focus on value added tasks requiring flexibility and intelligence, which leads to improvements in both efficiency and quality.

Evidence shows that for most ML classification problems, it takes little effort to achieve accuracy close to 80% but it is very difficult to improve beyond. State-of-the-art researches in data science go to great lengths to develop novel concepts and algorithms, only to gain a few percentages. The resulting solutions tend to be rather complex, involving hyper-parameter fine tuning that requires in-

depth data science knowledge. The program codes are quite complex to implement and maintained, and significant computing power is needed. The key point is, from a business perspective, the more complex the methods, the more difficult it is to build the system. All the more so that IT professionals who are eventually in charge of implementing and maintaining the production system likely do not have data science expertise, which may be a blocker.

One simpler alternative is to opt for a HuIL-based solution. We accept that complete automation is not realistic and design a system where machine and human collaborate: what can be automated is automated, what cannot be automated is handed over for coders to perform the task. The problem then becomes: 1) Build the automation part, 2) Design a mechanism whereby machine alerts human when it needs input and 3) Design an efficient UI to facilitate human machine interaction.

To tackle the difficult problem step by step, we applied Agile project management methods and developed the proof of concept in a number of increments as follows:

- We first focussed on developing a solution that we tested for UK data only, using a small dataset of about 200 UK receipts that we collected from colleagues at ONS. The receipts were from major supermarkets in the UK, namely Tesco, Asda, Aldi, Morrisons, Marks and Spencer, Lidl, Sainsbury's. We prioritised receipts from the major supermarkets because this is where we can make the most efficiency savings. The dataset contained no receipt from restaurants, petrol station, taxi, etc.
- Another prerequisite is that the receipts are in good conditions, although we also tested a small number of receipts that are faded, crumpled and torn to assess the performance of the methods.[1]
- The first proof of concept was developed on receipt images that have been scanned with a flatbed scanner, where a lot of parameters can be controlled. The problem is thus simplified, image processing can be kept minimal, as we will discussed in more details in the next section.
- As the first proof of concept showed promising results, we extended our tests to more receipts collected in other countries e.g. Dutch and Canadian receipts. All are supermarket receipts in good conditions. Languages tested are English, French and Dutch.
- We developed further image processing methods to tackle receipts that are captured by mobile phone app and we propose to compare OCR performances between flatbed scanning and mobile phone app scanning. To this end, we use a small dataset of Dutch receipts that were collected from colleagues at CBS. Images were then captured by staff at both CBS and ONS without specific instructions so we can perform tests in real-world conditions.

3 Receipt scanning

One interesting aspect of this research is to explore and compare various options for scanning paper receipts. What are the pros and cons of scanning receipts using a flatbed scanner, compared to using a mobile phone app? How do image format, image resolution, the quality of the original paper receipts and the scanner setting affect OCR accuracy?

[1] It is worth noting that currently, receipts collected for the UK LCF are sometimes annotated by respondents as well as by interviewers and coders, making it very difficult to OCR them. A separate investigation is being conducted by the survey team to understand how receipt annotations can be simplified.

3.1 Image format

The first parameter we investigated was the image format. We compared the most common image formats for the web and computer graphics [Witten et al. (1994)]: jpeg, gif, bmp, tiff, png and scanned pdf. These all belong to the family of *raster images* that are grids of pixels. Each image is a collection of countless tiny squares. Within the raster image family, there are two types of image compressions, *lossy* (jpeg, gif) and *lossless* (png, tiff, bmp). Jpeg also has a lossless version, but it is not widely supported.

With lossy compression, an image is compressed every time it is saved so that its file size is reduced. This can be achieved by partially discarding information, for example, by reducing the range of colours that the image contains. This is why jpeg files are usually smaller compared to lossless images, which makes jpeg more suitable for online applications. Transmission time, rendering time and storage space on the device are reduced, but this comes at the cost of losing image quality at each save. This is why OCR literature usually prefers lossless (png, tiff) over lossy (jpeg), even more so if the application is supposed to edit and save the images several times during the process. We are thus facing a problem of prioritising file size over image quality, or vice-versa. If receipts are to be scanned in the office using a flatbed scanner, storage space and transmission time are not an issue. However, if respondents are to make photos of receipts and send over to the agencies, png file size may be too large to be stored on the device and to be sent via Cloud.

As we aim to develop solutions that are suitable for both office scanning and mobile app scanning, jpeg is the recommended format. We first developed methods using our small dataset of 200 UK receipts, while looking for a way to test our methods on a larger dataset. Later into the project, we were able to collaborate with Statistics Canada who have a large dataset of about 100,000 receipts collected over many years for the Survey of Household Spending. As it turned out, all Canadian receipts were scanned into pdf format, so we slightly changed our receipt scanning approach to be compatible with Canadian data source. We first scan all UK receipts into pdf, as it is done at Statistics Canada, then we convert the pdf into jpeg. This way, our methods can be applied for both countries. Receipts are scanned one per page, the front of the receipt is on the right, the back of the receipt is on the left, as shown in Figure 7.

3.2 Image resolution

Because raster images are pixel-based, their quality depends on image resolution, which is measured in dots per inch (dpi). The higher the dpi, the better the resolution. To determine whether there is an optimal resolution for OCR, we converted the 200 UK receipts from pdf into jpeg images at the following resolutions: 150dpi, 300dpi, 600dpi and 1200dpi.

Performing OCR on this test dataset, we observed that 1200dpi consistently produced worse OCR results, as shown in Figure 8. It also required longer processing time compared to lower resolutions. The principal reason behind such low performance is that high image resolutions capture more noise, which has a negative effect on recognition. Although we did not observe any significant difference in performance between 150dpi, 300dpi and 600dpi, OCR literature usually recommends 300dpi as the optimal resolution that yields the highest accuracy [Archives (2017)].

In practice, OCR accuracy depends on a combination of two factors: resolution and font size. The lowercase letter x is usually used as a means to predict OCR performance. For good recognition, the height of the letter x must be about 20 pixels. For most UK receipts, characters are typically 10 point font size, which corresponds to 20 pixels at 300dpi. For smaller font sizes, a higher resolution will be required. Too low resolutions may cause speed degradation as uncertainty in character picture produces more recognition variants to process. The commercial OCR software Abbyy recommends

Figure 7: Receipts are scanned into PDF format, the front of the receipt is on the left, the back of the receipt is on the right. Note: this is a sample of Canadian receipt, it is not from the SHS dataset.

the following character sizes, which is applicable for most OCR engines:

- For simple script (e.g. English, French, alphabetic languages) and complex script languages (e.g. Thai, Arabic, Hebrew): recommended size = 20 pixels, minimal size = 12 pixels.
- For logographic script languages such as Japanese and Chinese: recommended size = 25 pixels, minimal size = 22 pixels.

3.3 Quality of the original paper receipts

Receipts are usually printed on thermal paper using a low quality printer, the ink often fades out quickly over time. Although we collected relatively good quality receipts from ONS staff for the research, evidence shows that it is usually not the case for receipts collected from survey respondents. Some proportion of real receipts may be wrinkled, folded, torn, stained, faded. Image processing

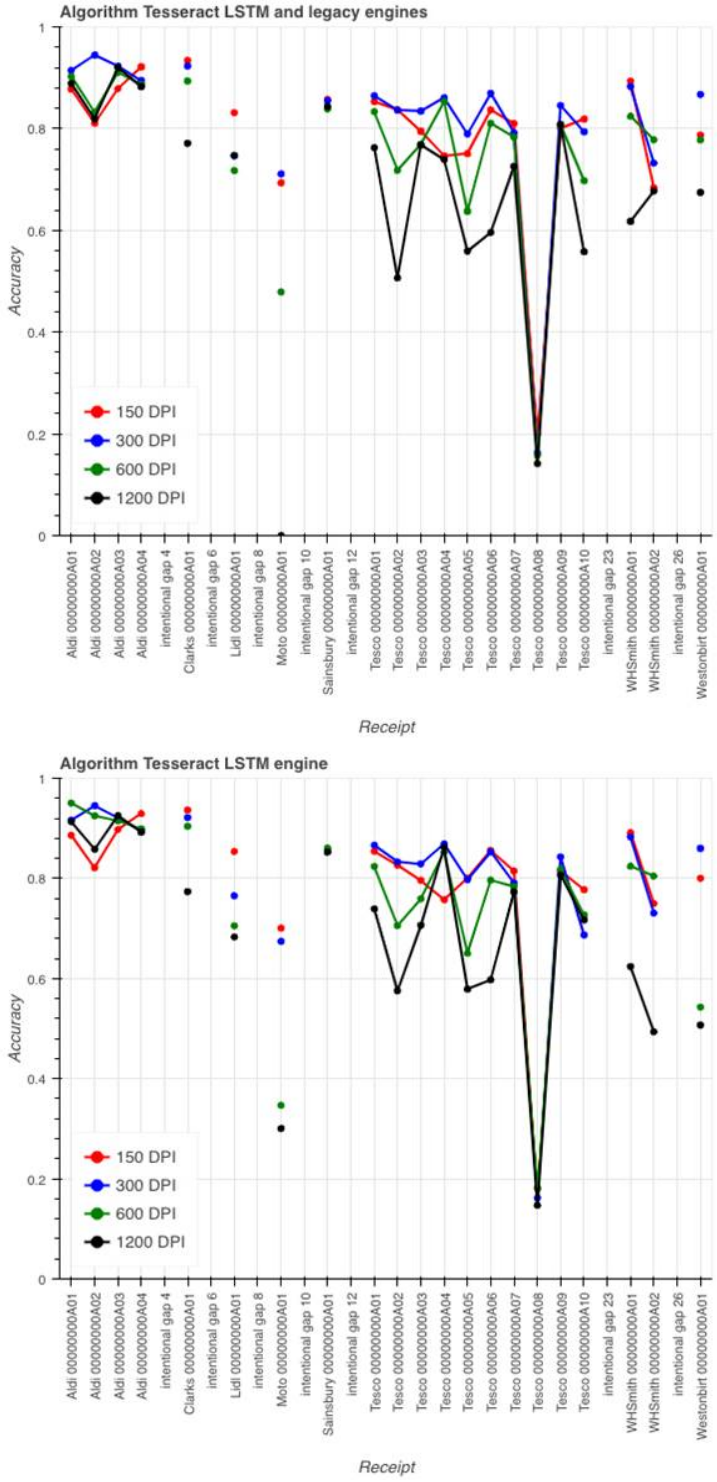


Figure 8: Assessing how image resolution affects OCR accuracy. A small dataset of receipts from various UK supermarkets are scanned at various image resolutions: 150dpi, 300dpi, 600dpi and 1200dpi. Optical Character Recognition is then applied to extract text from the receipts and compared against the gold standard (manual transcripts of the text on the receipts). Accuracy is derived from the normalised string distances between OCR outputs and the corresponding gold standard, Accuracy=1 meaning identical. Resolution 300dpi appears to perform best across supermarkets.

can repair such damage to some degree but not all. This problem is shop-dependent. Indeed, whilst some supermarkets print their receipts with strong ink on good quality paper, others use very thin paper and print on both side, such that the text at the back of the receipt is seen through, adding noise that affects recognition. Discount supermarkets typically use smaller font size to fit more information on the receipts, as well as using faded ink.

Regardless of image format and resolution, if the quality of the original document is really low, there is not much machine can do. One possible solution may be to build a HuIL-based system where coders intervene to transcribe text that machine cannot read. Their combined effort may help guess missing information from degraded data but there will always be situations where the receipt images are so degraded that neither human nor machine can read.

3.4 Scanner settings

Scanners produce a digital representation of the physical paper receipt, the quality of the digital document has an effect on OCR results. There are parameters that can be tuned to improve the scan quality. Since most receipts are in black and white, the scanner does not need to capture colours, which is preferable since coloured images require more storage space. Commercial OCR software often advise to scan documents in grayscale rather than black and white to preserve fidelity. However, in the end, all OCR engines expect a black and white image, so the grayscale image needs to be binarised at some point. Grayscale images provide the flexibility to tune for the optimal threshold so should be preferred to black and white, but this is a minor requirement. Brightness and contrast have an effect on OCR. Figure 9 shows some recommendations from Commercial OCR software Abbyy on how to adjust scanner brightness and contrast.

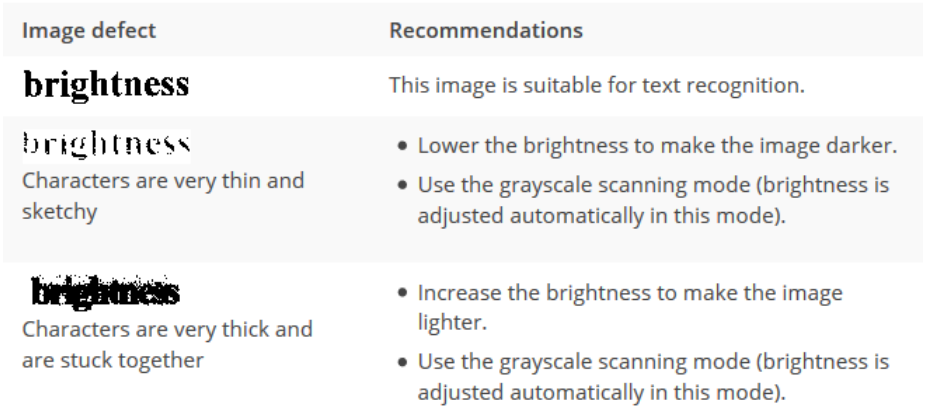
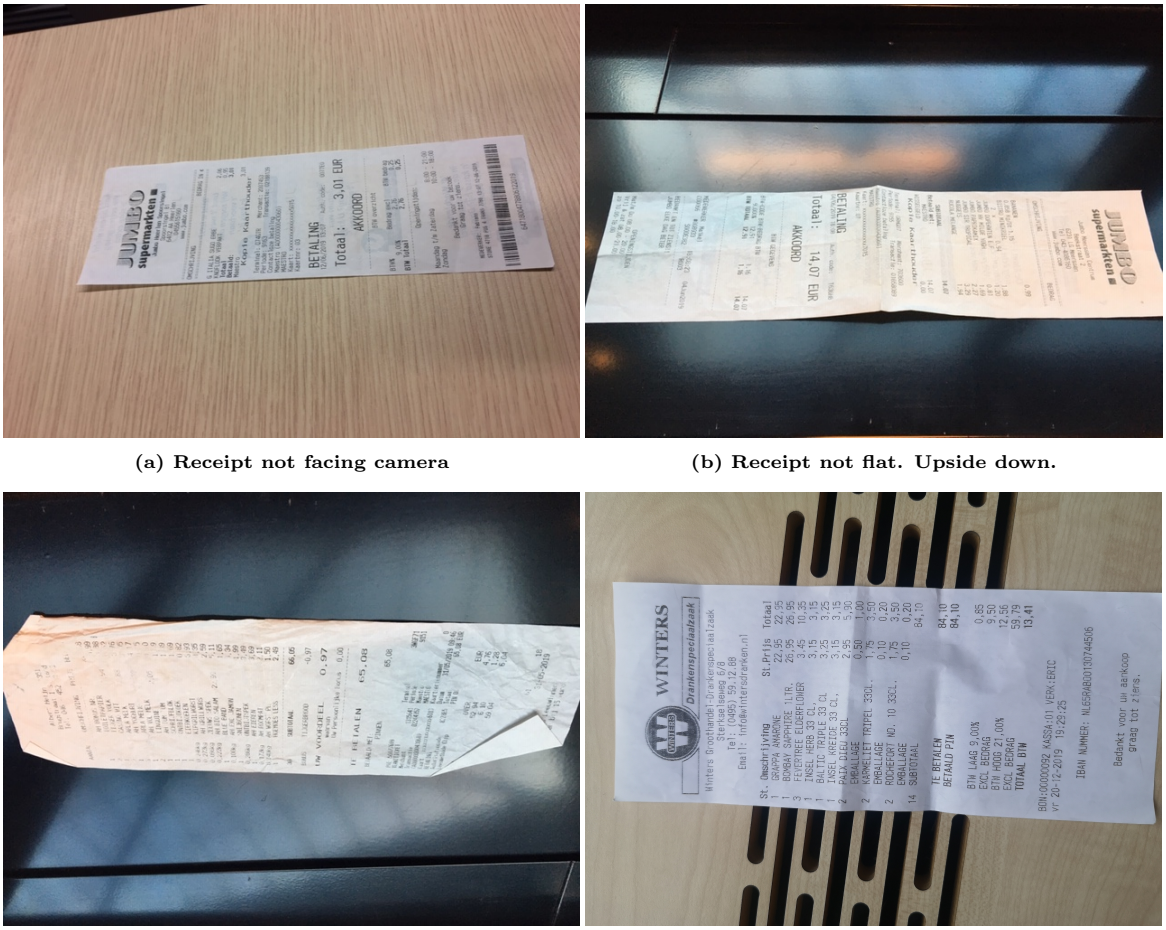


Figure 9: Abbyy recommendations of scanner settings for brightness and contrast.

3.5 Mobile phone app scanning

In previous sections, we investigated parameters that may degrade image quality and OCR accuracy. To great extent, these parameters can be controlled in an office setting where staff can be trained to make good quality scans, a good scanner can be purchased if test results support such decision. However, in this scenario, efficiency saving is less significant and processing time needs to be accounted for interviewers to collect the receipts and office clerks to carry out the scanning.

A more attractive scenario that complies with government Digital by Default strategy is to build a mobile phone app. Respondents make photos of receipts and upload images on a Cloud platform,



(a) Receipt not facing camera

(b) Receipt not flat. Upside down.

(c) Receipt is not rectangle

(d) Non-uniform background

Figure 10: Examples of Dutch receipts that are taken in real-world situations, no specific instruction was given on how the photos must be captured. Several problems can be observed: perspective angle of the receipts, orientation of the receipts, reflective surface, non-uniform surface, receipt is not a perfect rectangle, shadows, poor lighting, receipts are not flat, folds causing distortion of text lines.

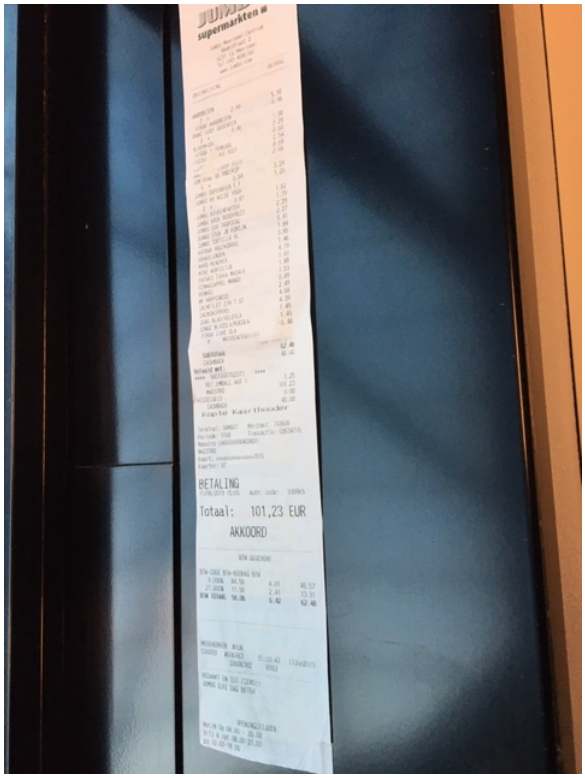
immediately accessible to the agency. Beyond the technological challenges and data security assurance that are out of scope for WP 4.2 and we will not discuss in this report, there are other potential problems that require careful investigation. Two main tasks at hand are: can we automatically crop the receipt out from the image? And is the quality of the cropped image good enough for OCR? Let us examine typical situations where these tasks are difficult. Figures 10 and 11 show examples of Dutch receipts that are taken in real-world situations, no specific instruction was given on how the photos must be captured. Due to data protection concerns, in this report, we only show images of receipts collected from colleagues at CBS, the photos were captured by staff at ONS who do not have specific knowledge of this research, so we expect their behaviour to be close to that of real-world respondents. The problems listed below are however observed by examining the larger dataset of a few hundreds receipts collected from real @HBS pilot tests.

The first cause of potential problems is due to the human factor. Compared to office staff who are trained to make good quality scans, respondents are not necessarily tech-savvy and do not know how the images are going to be automatically processed. Without specific instructions, they may make common mistakes such as positioning white receipts on a white background or non-uniform background, which makes automated cropping more challenging. For long receipts, they may zoom out too far, resulting in unreadable receipts. Poor lighting, poor contrast, shadows caused by objects,

(a) Folds causing shadows.

(b) Blurred image due to camera movements.

(c) Finger hiding text on receipt.



(d) Long receipt not entirely captured.

Figure 11: Examples of Dutch receipts that are taken in real-world situations, no specific instruction was given on how the photos must be captured. Several problems can be observed: missing information due to photo not capturing entire receipt, finger holding receipt overlapping text, blurred image due to movements of camera, long receipts may become unreadable.

hand obscuring text, blurred images caused by unsteady hands are problems we often observe in the dataset. Such problems do not occur when the receipts are scanned with a flatbed scanner.

The second cause of problems is technological. Depending on the quality of the mobile device, photos may be too low resolutions, which negatively affects OCR accuracy. On the other hand, a high quality image may be too large, which takes time to send and may increase respondent burden. In order to crop the receipt out from the image, common method consist of applying edge detection to find the contour of the receipt. The four corners are located and perspective transformation is applied to warped the receipt into a mugshot position. The presence of other objects or patterns on the background may cause difficulty for detecting the receipt in the image. If the receipt is not a perfect rectangle, the four corners are difficult to find. Folds and image angles can cause distortions that need to be repaired. Again, such problems do not exist to great extent if receipts are scanned with a flatbed scanner.

Summary of potential problems in mobile app scanning

- Rotation angles of receipts (sideways, upside down)
- Perspective angles of receipt, characters are distorted
- Zooming, especially long receipts may become unreadable
- Poor lighting, poor contrast
- Receipt not flat, folds can cause shadows and distort lines of text
- Non-uniform background, white background
- Reflective surface
- Blurred image due to movements of camera
- Image does not capture entire receipt, missing information
- Hand holding receipt creating shadow, overlapping text
- Receipt is not a perfect rectangle, folded corner
- Low quality camera producing low resolution images
- High quality images are large in size, file transfer takes longer, storage on device takes up space

As a result, instructions should be given to the respondents so the image quality can be controlled, either by means of written documentation or as basic checks implemented on the device. Care needs to be taken so we do not increase respondent burden. Further research is needed to decide how many problems machine can resolve, and what the instructions should be for the respondents. By using complex image-processing, you are left with a system that is more difficult to build and maintain. Decisions are to made such as, what part of the pipeline could be run on the device and which parts can be run on a server.

4 Image processing

With office scanning, many parameters can be controlled and optimised at data capture so there is no need to implement complex image processing. However, problems that are not caused by the scanning method but by the quality of the paper receipts still need to be repaired, such as

faded receipts and removing shadows caused by wrinkles on receipts, as shown in Figure 12. These problems exist in both flatbed scanning and mobile scanning.

To remove shadows, the idea is to filter the text from the background, then subtract the background from the original image. To filter the text, morphological dilation is applied, followed by median blur with a suitable kernel (here, we use a kernel of size=21). The result is a background that contains only shadows and discoloration. The difference between this background and the original image is then computed. As we can see in Figure 12, the shadows have not been completely removed so we need to apply further cleaning. First, we apply Gaussian blur that is a low-pass filter, which helps rid the image of high-frequency noise. Then, we apply OTSU's thresholding to obtain the final image. Details on the algorithms used can be found in [Beyeler (2017), and Sharma et al. (2019)].

Figure 12: Image processing to repair low quality receipts. From left to right: 1) Original scanned image with dark shadows caused by wrinkles on the receipt, 2) Shadows have been removed but it is still noisy, 3) Gaussian blur then OTSU's thresholding are applied to clean noise.

Basic image cleaning can be applied to improve the clarity of the image, as shown in Figure 13. This includes thresholding methods: simple thresholding that converts the image to black and white using a global threshold for the entire image, more complex thresholding that employ various techniques to tune for the optimal threshold. However, it is very important to stress that high image quality does not always lead to the best OCR result. A clear image where noisy blobs are too visible will negatively affect OCR because the recognition will confound these with real text. It is rather difficult to automatically decide what image processing technique is best for every situation. Some level of human intervention will be needed to decide whether image processing is required to improve OCR.

To automatically crop receipts from the original images captured by mobile phone app, we explored two approaches, traditional image processing techniques and state-of-the-art deep learning.

4.1 Traditional image processing

In order to detect the largest white blob in the image, the color scheme of the image is first converted from Red-Green-Blue (RGB) to Hue-Saturation-Value (HSV). From this color scheme only the Saturation channel is kept to register the white receipt. Using this channel is easier compared

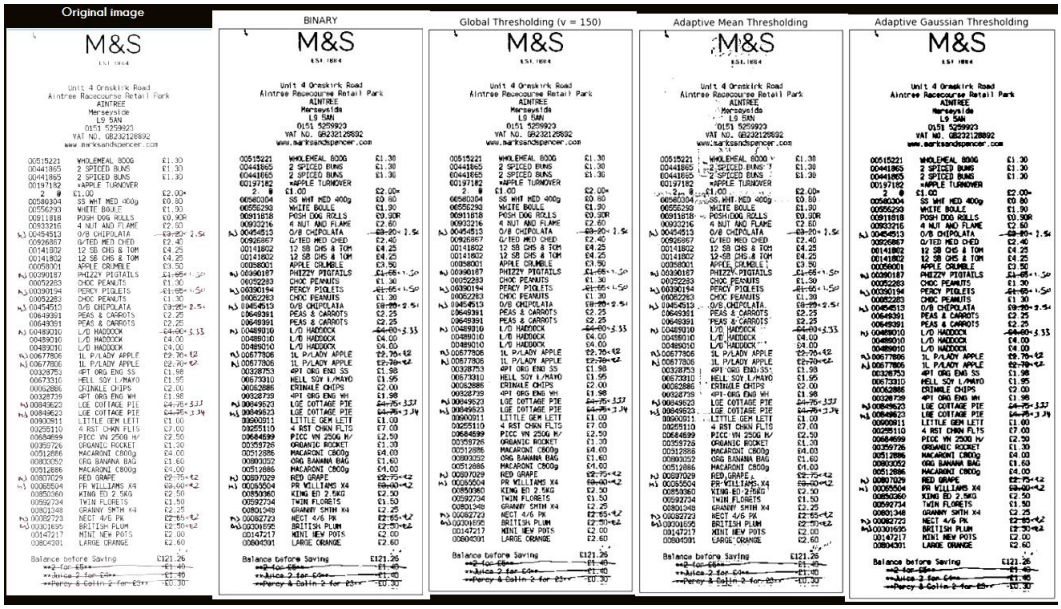


Figure 13: From left to right: Original image, Binarisation, Global thresholding (threshold=150), Adaptive mean thresholding, Adaptive Gaussian thresholding.

to the RGB representation or its gray-scale values, as the saturation channel lends itself well to detecting colors that are close to white (the lower the value the whiter the color is). Images are then re-scaled to a fixed height, this both reduces the amount of pixels that each of the following steps has to process as well as keeping a baseline for all the chosen hyper parameters. Next, bilateral filtering is applied to reduce noise whilst preserving edges. Afterwards a contour detection algorithm is used on the image and the largest blob is kept, because we make the assumption that the receipt is the main object in the photo. The shape of this contour can vary, based on the input, hence the contour is reduced to the four edges/corners that are needed to apply image transformation. The image is transformed to retain only the content within the four edges. As a final step, further enhancements are applied such as shadow removal, and contrasting to make the contrast between black and white stronger. The methods are implemented using the Python package Scikit-image [Sharma et al. (2019)].

4.2 Deep learning

The traditional approach did not always properly detect the contours of a receipt in an image. Reasons such as bad lighting, white backgrounds, or very noisy backgrounds were common problems. To counter these problems an existing convolution neural network was used in combination with transfer learning to yield better results. The method used is an adaptation of a Region-Convolutional Neural Network (R-CNN), that is one of the state-of-the-art CNN-based deep learning model used for object detection, called Mask R-CNN. This adaptation not only returns the bounding boxes of a detected object, but also the specific pixels in the image belonging to this object. The original model was trained to detect 1000 different object, hence the last layer was retrained by using 400 annotated receipts. In each of the receipts the specific bounding boxes were annotated by hand, and from these bounding boxes the including pixels were calculated.

The resulting model was then used to replace most of the traditional image processing steps that involved. In Table 2 you can see that the Deep learning approach performs 14.8% better than the Traditional method when looking at the Intersection over Union (IoU) score, and is also more

Method	IoU score	Detected receipts
Traditional	0.71	89
Deep Learning	0.81	100

Table 2: The traditional method compared to the deep learning method, on 100 previously unused photos. With the Intersection over Union score and whether the method could detect a receipt.

stable as is shown by the fact that it can detect all 100 receipts, compared to the 89 receipts by the traditional method.

For the deep learning pipeline the RGB image only had to be rescaled, after which the algorithm returned to pixels of the receipt. As the algorithm always detects the insides of the receipt, leaving small parts missing out, the image is dilated to incorporated these missing pixels as well. The result is then used to detect the corners and edges of the receipt. To repair skewed images, Hough transformation is applied to detect the most dominant lines, which are then used to compute the intersections between the vertical and horizontal lines. The intersection points found are clustered into four points after which the image is transformed based on the four cluster centres. Finally, shadow removal and contrasting are also applied in this approach to enhance the resulting image. Methods are implemented using the Python packages Scikit-image [Sharma et al. (2019)], M-RCNN [Abdulla (2017)] and TensorFlow [Martin Abadi (2015)].

Figures 14 and 15 show examples of receipts being automatically cropped out from photos. Although traditional image processing techniques work well on simple cases, they struggle on more challenging photos where the receipt is not facing the camera, the contour is not a perfect rectangle and there are various objects in the background. Deep learning performs undoubtedly better in these cases.

In some particular cases where the corners of the receipt in the photo are slightly distorted, the deep learning model may struggle to retrieve the correct shape, resulting in the text being skewed, which degrades OCR accuracy. A further correction needs to be applied as shown in Figure 16. Hough transformation used together with deep learning to correct skewness of text lines.

5 Optical Character Recognition

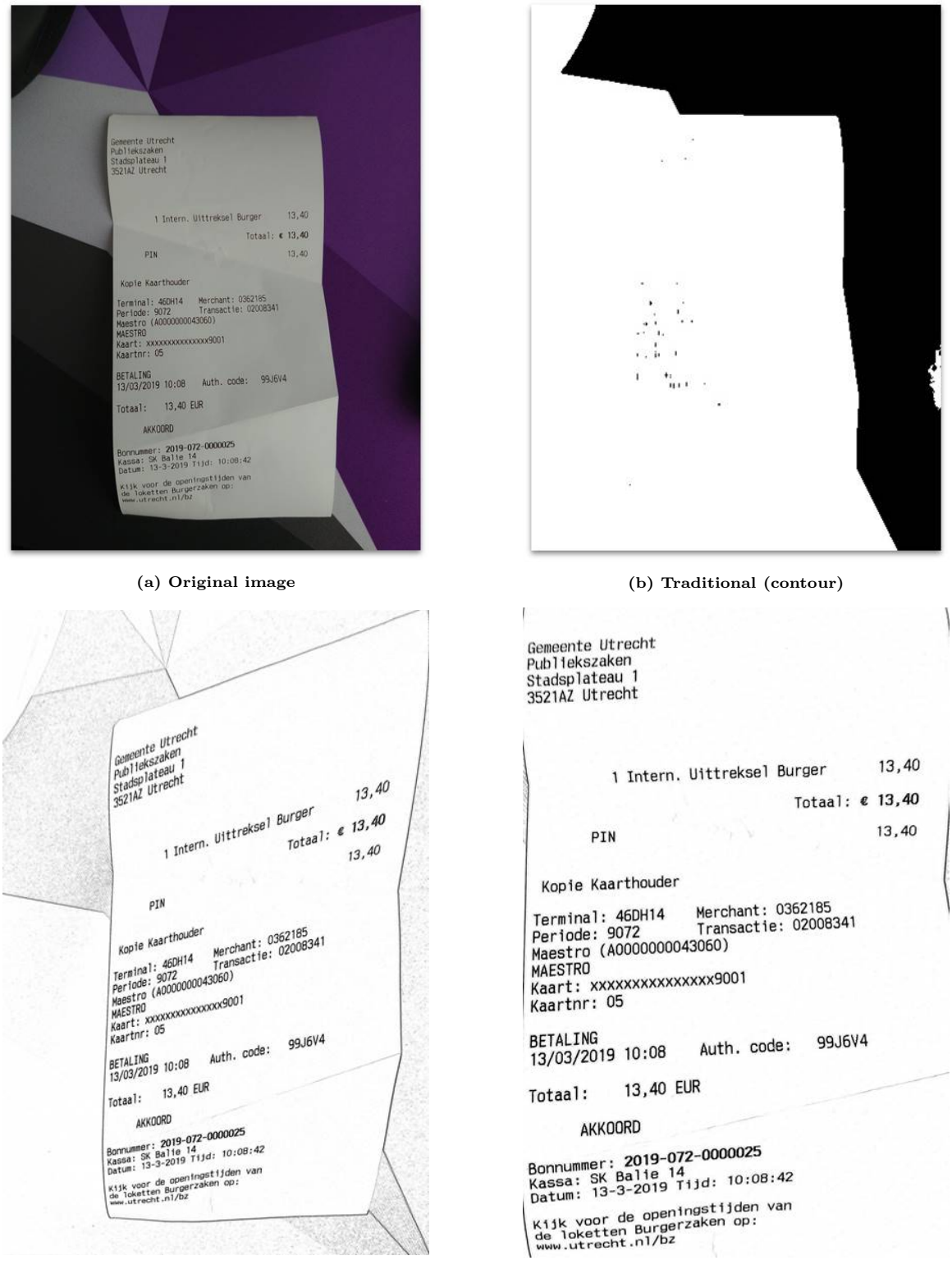
The history of OCR began in 1912 with the *optophone*, a device that converted written text to speech to help the blind read. From as early as the 50's, companies started using OCR to automate data entry for business documents. Algorithms have become better and better over time and there is no shortage of choices nowadays; some OCR software are easy to use, some require more programming to make them work. Some are very expensive, some are free and open-source.

5.1 Selecting a suitable OCR engine

Since we aim for a solution that will be made available for anyone to use, we rule out proprietary software. For security reasons, we also rule out OCR Web Services that use APIs to interface between an external server and client computers inside the government office. Whilst it is possible to assess and manage information assurance related risks, Web services are more complicated to put in place within the time frame of the project. Therefore, we preferred stand-alone software and short-listed three solutions for testing: Tesseract [Smith (2007)], CuneiForm [Tomaschek (2018)], and Calamari [Christoph Wick (2018)] as shown in Table 3.

	Tesseract	CuneiForm	Calamari
Developer	Hewlett-Packard Google	Cognitive Technologies	University of Würzburg
Licenses	Apache	BSD	No license
First release	1985	1996	2018
Latest release	2018	2011	2018
Supported platform	Windows Mac OS Linux	Windows Mac OS Linux	Windows Mac OS Linux
Supported languages	116	23	Unknown
Supported fonts	any printed font	any printed font	Unknown

Table 3: Comparison of three free open-source OCR solutions: Tesseract, CuneiForm and Calamari.



(a) Original image

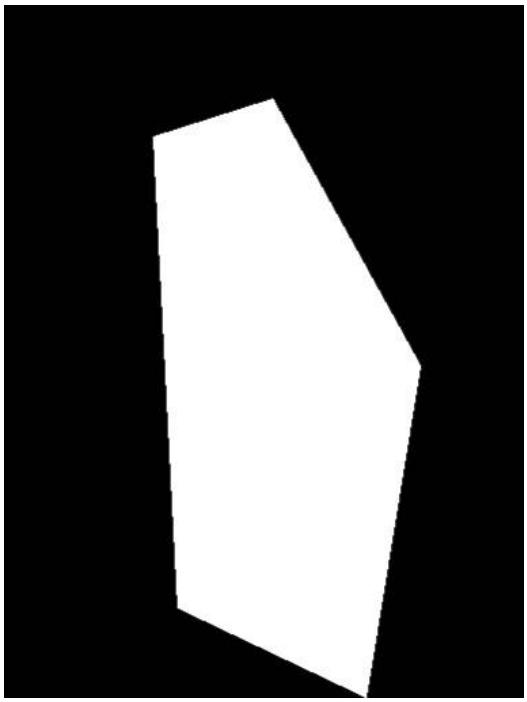
(b) Traditional (contour)

(c) Traditional (transform)

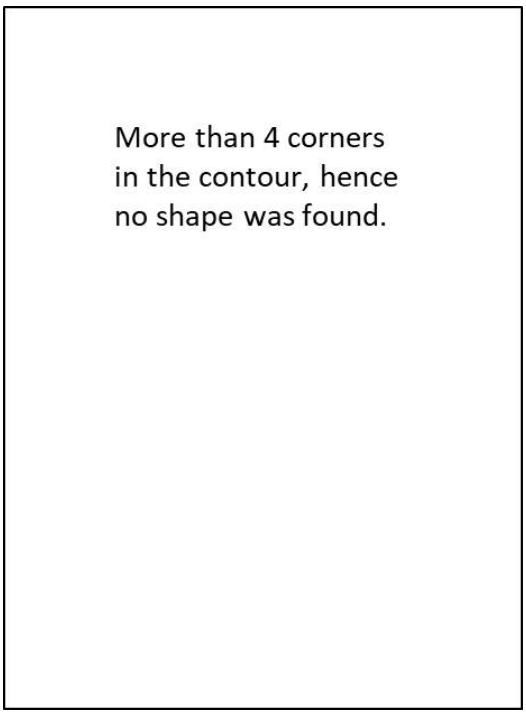
(d) Deep Learning (Transform)

Figure 14: Automated cropping receipt from image. Comparison traditional image processing techniques against deep learning with Region Convolutional Neural Networks.

(a) Original image



(b) Traditional (contour)



(c) Traditional (transform)

(d) Deep Learning (Transform)

Figure 15: Automated cropping receipt from image. Comparison traditional image processing techniques against deep learning with Region Convolutional Neural Networks.

(a) Original image

(b) Deep learning only

(c) Deep learning with Hough

(d) Original image

(e) Deep learning only

(f) Deep learning with Hough

Figure 16: Hough transformation used together with deep learning to correct skewness of text lines.

These three packages were selected because not only they are free and open-source but also easy to import into our Python program. Tesseract is the stand-out winner with a large user community, many discussion forums and free support platforms. Popularity is an important selection criterion for open-source products to ensure that the software is always up-to-date, bugs are quickly spotted and fixed. On the downside, Tesseract is not exactly easy to use. It has more than 400 parameters one can optimise; it is powerful and flexible but requires some programming knowledge to make it work. This is probably why Tesseract is less popular than some OCR Web Services that are more 'plug-and-play'.

The latest version of Tesseract is 4.1.1 released in 2019 [Releases (2019)]. It implements both an OCR algorithm based on traditional *pattern recognition* and a new recogniser, a Recurrent Neural Networks called Long Short-Term Memory (LSTM). The LSTM recogniser is used by default, it can fall-back to the legacy recogniser if it fails. The LSTM model is trained on images of printed text in various fonts, types (normal, bold, italic) and image quality, including a significant amount of degraded images produced by cameras.

In theory, the LSTM character recogniser is language dependent. Indeed, LSTM model learns character sequences that are language-specific, for example, *'schw'* exists in German but not in French. Having said that, receipt descriptions contain many product names that are universal. For instance, *'Schwarzkopf'* and *'Schweppes'* are sold in many countries. In our tests, we first used the model trained for English to process Dutch receipts, then we added the model trained for Dutch, processed the same receipts and compared the results. We observed no noticeable improvement with using Dutch language model, however it is worth noting that we only conducted tests on a small dataset of a dozen of Dutch receipts.

5.2 Data parsing

We know how to extract raw text from receipts using OCR. But data without meta-data is not much information. For instance, some receipts can be very long and contain irrelevant information about prizes, adverts, clubcard advantages and so on. How can we extract from there useful information such as item descriptions, prices, dates and shop names? There are two strategies:

- `Strategy 1`: use image processing to locate relevant information on the receipt image then OCR only these lines.
- `Strategy 2`: OCR all text from the receipt then use string manipulations to extract the relevant information.

Strategy 1

We first explored strategy 1. The image was cut into individual lines of text using Efficient and Accurate Scene Text (EAST), an algorithm commonly used for recognising car number plates from photos. We manually labelled lines that we wished to keep as *'good'* and those we wished to discard as *'bad'*. We computed the pixel intensity profile for every line and used this labelled data to train a machine learning model. For example, the pixel intensity profile of item lines is typically *valley-plateau-large valley-plateau-valley*, as shown in Figure 17. If we label this line as *'good'*, every time the model sees a line with the same profile, it will know that this line should be kept.

Because receipts are different from shop to shop, we had to train one specific model for each shop. About 16 receipts per shop were required to train a sufficiently accurate model. It became quickly evident that this strategy was rather burdensome due to the variety of shops and it would not be

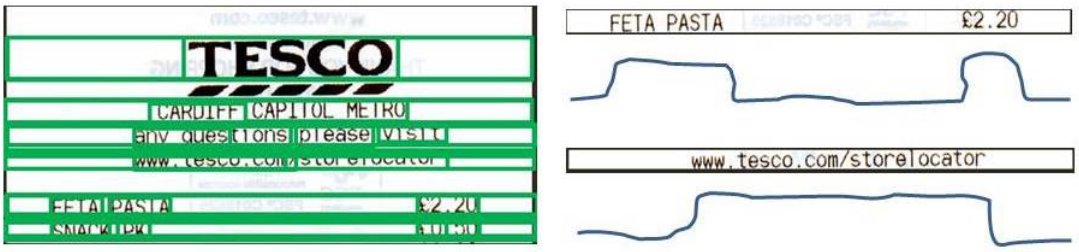


Figure 17: EAST algorithm is used to segment the receipt into strips of text lines then the pixel intensity profiles is computed for each strip.

easy to deploy and maintain an automation system that require one specific model for every shop. Furthermore, our tests showed that whilst the method performed well on extracting item lines, it also captured many irrelevant lines that exhibited similar profile. Another problem was to capture lines that contained dates and payment mode because such lines usually do not exhibit any recognisable structure. In terms of processing time, it took about 25 seconds to process an average sized receipt.

Strategy 2

Therefore, we abandoned strategy 1 to investigate strategy 2. This method uses regular expressions and fuzzy matching to recognise patterns and extract useful information from raw text. For example, we know that in the UK, dates are usually formatted to one of the following patterns: dd/mm/yyyy, dd/mm/yy, d/mm/yyyy, dd/m/yyyy, dd-mm-yyyy, dd.mm.yyyy, etc. Thus, the following regular expression can be used to catch dates that appear anywhere in the raw text:

$$\backslash d\{1,2\}[./-]\backslash d\{1,2\}[./-]\backslash d\{1,4\}$$

The expression above literally means *'looks for any sequence of characters that satisfies the following pattern: 1 or 2 digits, followed by either a dot or a slash or a dash, followed by 1 or 2 digits, followed by either a dot or a slash or a dash, followed by any number of digits between 1 and 4'*. Similar logic can be applied to retrieve UPC barcodes. Regular expressions are easy to implement in Python and run very fast.

To recognise shop names, we use a dictionary of known shops and apply fuzzy matching to detect if any known shop names appear in the raw text. Most Household Budget surveys already have such a list of shop names so this should not require extra effort. The limitation of this method is that it requires some level maintenance to keep the dictionary of shop names up-to-date, new shops need to be added. In our proof of concept, this dictionary is implemented as a simple human-readable text file that does not require any data science or programming knowledge to maintain it. All text is first converted to lowercases to account for varying styles e.g. *TESCO* or *Tesco*, then comparison is done by fuzzy matching to account for spelling mistakes due to incorrect OCR. For instance, if the receipt is faded, *Tesco* may be wrongly transcribed as *Tesc0*. In such case, where an exact string matching would fail, a fuzzy match would match two strings that are sufficiently close.

Detecting item lines is the most challenging problem. We propose to look for keywords. For example, we know that on many receipts, the line preceding the item lines usually contains keywords such as *'Description'*, *'Price'*, *'Quantity'*. With Asda receipts, for instance, the line preceding the item lines always exhibits the pattern 'ST. ⟨ 5 digits ⟩ OP.'. Similarly, we know that the line that immediately follows the last item lines usually contains keywords such as *'Total'*, *'Balance to pay'*, *'Subtotal'* or *'Sub-total'*, etc. Thus, we build a dictionary of keywords and use fuzzy matching to locate where these keywords appear in the raw text. Knowing the positions of the preceding line and

the succeeding line to the item lines, we can segment out anything in between and obtain the item lines. The dictionary is an unordered list of words that is used for all receipts. The model is not shop-specific, hence easier to build and maintain. Figure 18 shows OCR output of an UK receipt, the pseudo-code for extracting the item lines is as below. As a prerequisite, we have a dictionary of keywords. The Start keyword list includes: *'Description'*, *'Price'*, *'Quantity'*, *'GBP'*, *'ST.'*, etc. The Stop keyword list includes *'Total'*, *'Balance to pay'*, *'Subtotal'*, *'Sub-total'*, etc.

Pseudo codes for extracting item lines

⇒ Record start line index as 0 and add this to a list of possible start lines.

⇒ Record stop line index as the index of the last line of text and add this to a list of possible stop lines.

⇒ Read the receipt OCR output text line by line. At each line, apply fuzzy matching to recognise start and stop keywords.

⇒ If a start keyword is found, add the index to the list of possible start lines.

⇒ If a stop keyword is found, add the index to the list of possible stop lines.

⇒ Once all the text has been processed, initiate the start line index to the maximum value in the list of possible start lines (i.e. the last line where a start keyword was found). Initiate the stop line index to the minimum value in the list of possible stop lines (i.e. the first line where a stop keyword was found).

⇒ Perform basic checks such as start line index should be smaller than stop line index, otherwise, look for other possible start and/or stop indices.

In this example, the last text line where a start keyword was found is the fourth line. There are several possible stop lines where keywords such as *'Total'* and *'Subtotal'* were found; the smallest line index is kept, which corresponds to line *'Subtotal 41.29'*. This is an acceptable choice because the line index is greater than 4 that is the index of the start line.

Figures 19 and 20 show a comparison between OCR strategy 1 and strategy 2 for the same receipt. With strategy 1, one purchased item - *'card'* - is missing due to the model not being able to retrieve all correct item lines. With strategy 2, one extra line was erroneously captured, which requires human intervention to remove it. Although the output is not perfect in both cases, it takes less time to remove an extra line compared to adding a missing line, so strategy 2 appears to perform better. Other information such as shop names, dates, barcodes and payment mode are also correctly captured. We have investigated several approaches for OCR and data parsing, and so far, strategy 2 works best. However, this method may not be very robust nor scalable and we will investigate further for a better solution. For example, the blank spaces between blocks of text can be used to help data parsing. However, this is not possible to exploit this feature for now because Tesseract by default collapses all blank spaces to a single one.

In terms of processing speed, it typically takes 2 to 6 seconds to OCR a receipt in relatively good condition, meaning it is not too faded or torn etc. The size of the receipt does not have significant impact on processing time, but the image quality does as Tesseract then re-tunes its parameters to improve the output, which can take up to between 10 and 16 seconds per receipt.

	UPC barcode	RECDESC	Price	Pay-mode	Shopname	Date
0	44203	still water 12 x sooml	1.49	card	Aldi	24.03.18
1	40619	farmhouse seeded batc	0.79	card	Aldi	24.03.18
2	52073	strawberries	2.20	card	Aldi	24.03.18
3	79356	truffle tin	3.99	card	Aldi	24.03.18
4	81434	single origin easter e	4.49	card	Aldi	24.03.18
5	81434	single origin easter e	4.49	card	Aldi	24.03.18
6	7176	secrets easter egg	2.99	card	Aldi	24.03.18
7	71046	herbal shower gel	0.65	card	Aldi	24.03.18
8	63983	mascarpone/ricotta	0.75	card	Aldi	24.03.18
9	51213	pineapple	0.69	card	Aldi	24.03.18
10	51071	large vine tomatoes	1.19	card	Aldi	24.03.18
11	82011	childrens socks 5 pack	2.99	card	Aldi	24.03.18
12	66367	ment /chilli/orange	1.29	card	Aldi	24.03.18
13	63574	single origin bar	1.29	card	Aldi	24.03.18
14	82011	childrens socks 5 pack 2 99 a	.	card	Aldi	24.03.18
15	2271	moist meaty chunks	2.29	card	Aldi	24.03.18
16	80006	antipasti	1.05	card	Aldi	24.03.18
17	74928	avocado twin pack 1 75 a	.	card	Aldi	24.03.18
18	74928	avocado twin pack	1.75	card	Aldi	24.03.18
19	60668	italian mozzarella	0.43	card	Aldi	24.03.18
20	71046	herbal shower gel	0.65	card	Aldi	24.03.18
21	77617	paleo / wholefood bar	0.49	card	Aldi	24.03.18
22	77617	paleo / wholefood bar	0.40	card	Aldi	24.03.18

Figure 18: Receipt of low quality from supermarket Aldi: faded text is faded, shadows, text printed on the background is seen through. Image processing is first applied to improve image quality. OCR is performed to extract raw text then parsing is applied to retrieve relevant information: items, barcodes, prices, shop name and date are extracted.

5.3 Measuring OCR accuracy

So far, we have visually assessed the correctness of the outputs, which is possible as long as we have a small number of receipts to inspect. However, to evaluate the robustness of the methods, we need to test on larger volumes of data, thousands of receipts or more. Therefore, it is no longer possible to assess results visually, we need to formally define a quantitative metric for OCR accuracy and develop an automated procedure to calculate test results.

OCR accuracy can be measured by comparing OCR outputs against the gold-standard, meaning the exact transcripts of the text on the receipts done by human. There are three types of information for which accuracy should be measured in different ways:

1. Information such as dates, prices, barcodes and shop names need to match exactly. Either a shop is identified or it is not, there is no blurred line.
2. There is a greater leeway with item descriptions. A description may be OCR'ed 100% correctly or with some spelling mistakes. If there are not too many incorrect characters in the string, ML models may be able to classify the item correctly. If there are too many incorrect characters, automated classification will fail. We can apply fuzzy matching and use a string edit measure such as the Levenshtein distance to measure the degree of similarity between the OCR output and the gold standard.

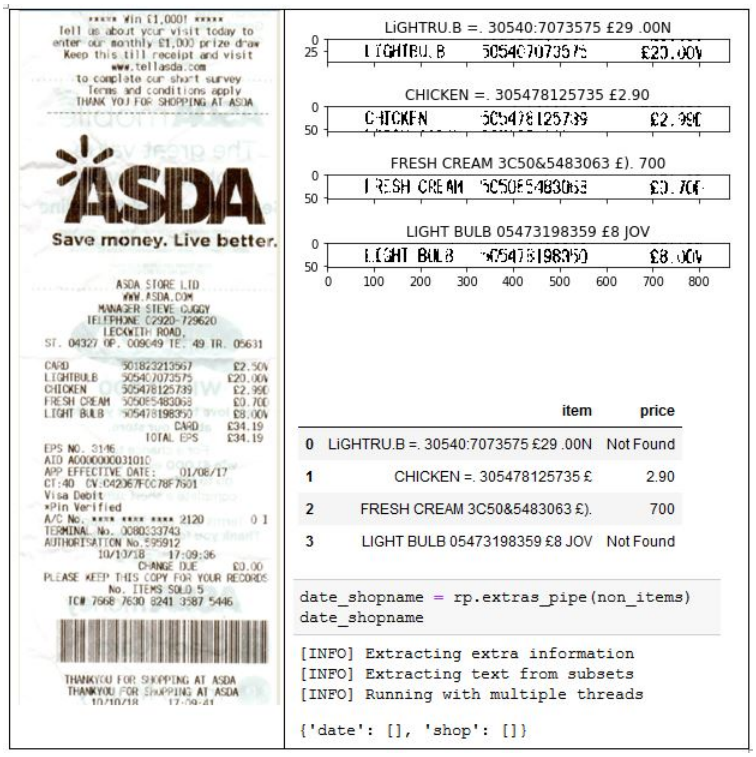


Figure 19: OCR and data parsing - strategy 1: Item *'card'* is missing due to the model not being able to recognise all item lines. Output text is very noisy.

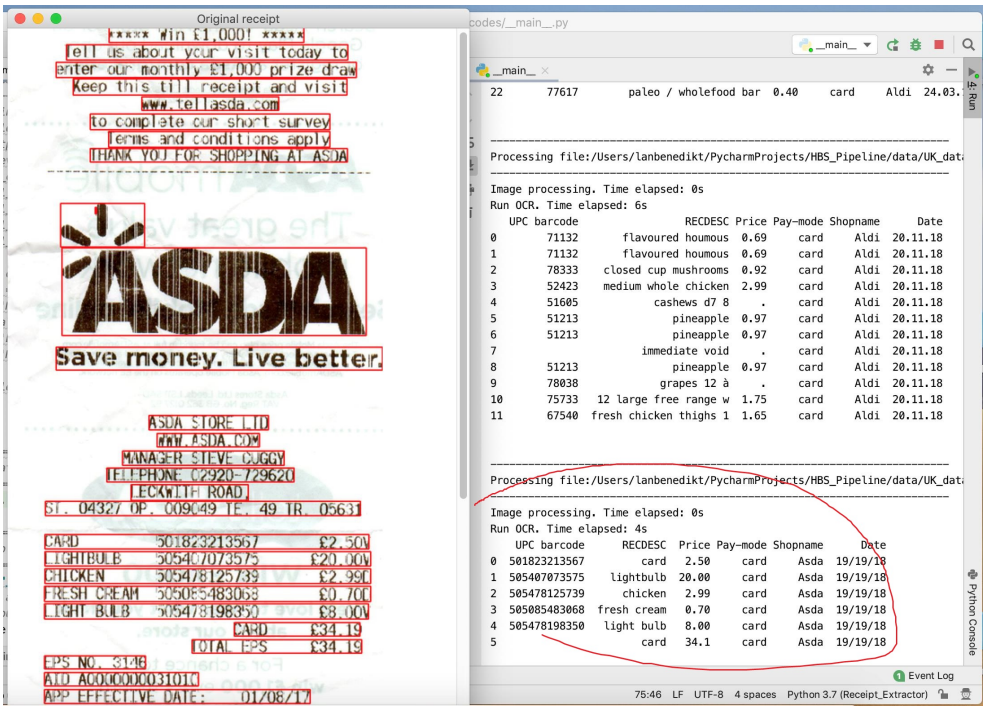


Figure 20: OCR and data parsing - strategy 2: All items have been captured correctly and data parsing performs well. There is one extra line *'card'* that needs to be removed, this requires human intervention.

3. We only wish to retrieve relevant information from a receipt, such as descriptions, prices, shop name, barcodes and date. However, it may happen that the data parsing process does not perform correctly, causing irrelevant lines to be kept or 'good' lines to be discarded. We need to measure that type of errors by counting the number of extra lines and missing lines.

Exact matching: dates, prices, barcodes and shop names

Dates are split into day, month, year and formatted as numerics *dd*, *mm*, *yyyy*. Currency symbols and dot separators are stripped off from prices. The comparison of numeric quantities can be achieved by simply subtracting the OCR output from the gold-standard. To compare shop names, we convert both the gold-standard and the OCR output to lowercase and compare the two strings. In both cases, we define a score of 1 if they match, and 0 otherwise.

Fuzzy matching: item descriptions

Text can be directly compared by applying fuzzy matching and similarity metric such as the Levenshtein distance, which is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions, or substitutions) required to change one word into the other. Formally, The Levenshtein distance between two strings a, b (of length $|a|, |b|$ respectively) is given by $D_{a,b}$ where

$$D_{a,b} = \begin{cases} \max(i,j) \ \text{ if } \ \min(i,j) = 0 \\ \min \begin{cases} D_{a,b}(i-1,j) + 1 \\ D_{a,b}(i,j-1) + 1 \\ D_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} \quad \text{otherwise} \end{cases} \tag{1}$$

where $1_{(a_i \neq b_j)}$ is the indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise, and $D_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b. Figure 21 shows an example of how the distance between two strings is measured.

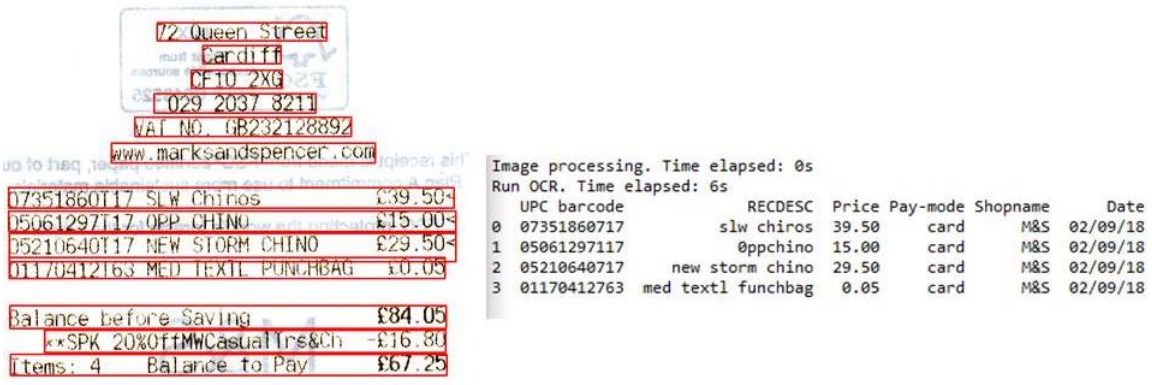


Figure 21: Measuring OCR accuracy with Levenshtein distance.

We can then define a normalised similarity score that takes into account the length of the string as below. If two string are identical, $D_{a,b} = 0$ and $D_{a,b} = 100\%$.

$$S_{a,b} = 100 \times (1 - \frac{D_{a,b}}{max(|a|,|b|)})(\%) \tag{2}$$

Gold standard	OCR output	Levenshtein distance
slw chinos	slw chiros	1
opp chino	0ppchino	2
new storm chino	new storm chino	0
med textl punchbag	med textl funchbag	1

Table 4: Measuring OCR accuracy using the Levenshtein distance.

Extra lines - Missing lines

For each text line in the gold standard, we apply Fuzzy matching to find the equivalent line in the OCR output. The number of missing lines is the number of lines that have not found a match. Similarly, for each text line in the OCR output, we apply Fuzzy matching to find the equivalent line in the gold standard. The number of extra lines is the number of lines that have not found a match.

In terms of error correction, extra lines are quick and easy to repair, one only needs to remove the line. Missing lines need to be added, which takes longer to manually transcribe the missing information. Therefore, we should set the default parameters of the OCR model to a high false positive, so we do not have too many missing lines, even though it may let through more extra lines that need to be removed.

5.4 Scalability of the method

Once the methods implemented and tested on our small dataset of 200 UK supermarket receipts, the next step is to investigate how the solution scales up. To answer this question, we need to test on a greater variety of receipts, in other languages, and on larger volumes. The UK LCF team has recently launched a large scale collection of shopping receipts and are producing the gold-standard transcriptions that we need for quality assurance. This takes a lot of time and resource so we hope to report test results in the near future, unfortunately, this will be after the @HBS project time frame. In September 2019, the ONS Data Science Campus started a new collaboration with Statistics Canada, which gave us the opportunity to explore how our methods could be adapted for Canadian receipts. Due to data protection restrictions, we will discuss the methods using examples of personal receipts that were obtained from colleagues at Statistics Canada. All receipts shown in this report are voluntary data, they are not from the Canadian Survey of Household Spending.

Description of Canadian receipts

Unlike UK receipts that are usually organised into well-defined columns, Canadian receipts appear more spread out. Indeed, most receipts seem to be either organised into loosely-defined columns or do not exhibit any clear patterns at all. Thus, any parsing method that relies on geometrical structures (such as our data parsing method Strategy 1) would fail.

Furthermore, whilst UK receipts are often succinct, Canadian receipts provide very rich additional information such as headers, special offers, tax codes, pricing details, membership offers, all this additional information is embedded within the purchased items text lines. One item may be printed on several lines, as shown in Figures 22 and 23. This particularity requires that we apply more thorough data cleaning to remove text lines that are not relevant for statistical purpose.

Receipts from the same shop can be formatted differently, depending on whether they are in English or in French. One example is dates where a large variety of formats can be found, the order of day, month and year is not unique across the country, leading to possible confusion. For example, *06/09/19* could be 6^{th} September 2019, 9^{th} June 2019, or 19^{th} September 2006, etc. One needs further information such as the survey data collection month and year to ensure dates are parsed correctly.

Therefore, we had to extend the methods we developed for UK receipts to cope with such particularities. We apply the same algorithms to extract shop names, dates, payment modes and item lines as we did for UK receipts. Then, we apply additional data cleaning to filter information we do not wish to keep. As a prerequisite, we keep a list of keywords for headers (e.g. Deli, Meat, Dairy, Produce), special offers (e.g. in-store offer, membership advantages, vouchers). The list used to determine the start line includes keywords such as 'Welcome#', 'ST#', 'Bienvenu#', membership number, etc. The list used to determine the stop line includes keywords such as 'Total', 'Sub-total'. The pseudo code is as follows:

Pseudo codes for retrieving items lines from Canadian receipts

⇒ Record start line index as 0 and add this to a list of possible start lines.

⇒ Record stop line index as the index of the last line of text and add this to a list of possible stop lines.

⇒ Read the OCR output line by line from the top and search for keywords and string patterns to recognise shop name, dates, payment modes, start keywords, stop keywords.

⇒ If a start keyword is found, add the index to the list of possible start lines.

⇒ If a stop keyword is found, add the index to the list of possible stop lines.

⇒ Once all the text has been processed, initiate the start line index to the maximum value in the list of possible start lines (i.e. the last line where a start keyword was found). Initiate the stop line index to the minimum value in the list of possible stop lines (i.e. the first line where a stop keyword was found).

⇒ Perform basic checks such as start line index should be smaller than stop line index, otherwise, look for other possible start and/or stop indices.

⇒ For shops where there is no start keyword to indicate where the item descriptions start, look for the first header.

⇒ Retrieve all lines of text between the start line and stop line identified above.

⇒ Apply the list of keywords to identify and discard irrelevant lines, the remainder are lines where there are only item descriptions.

⇒ Apply regular expression to parse each line into description, price and barcode.

So far, the method seems to work relatively well on both French and English receipts. Preliminary tests run by Statistics Canada have shown promising results. Below is the output from a short test run on a dataset comprising all receipts from one single popular Canadian retailer. The data was collected for the year 2017:

- Total number of receipts: 744
- Total number of items: 5147

Figure 22: Examples of Canadian receipts in English. Note: dummy data for illustration purpose only, not extracted from the SHS survey data.

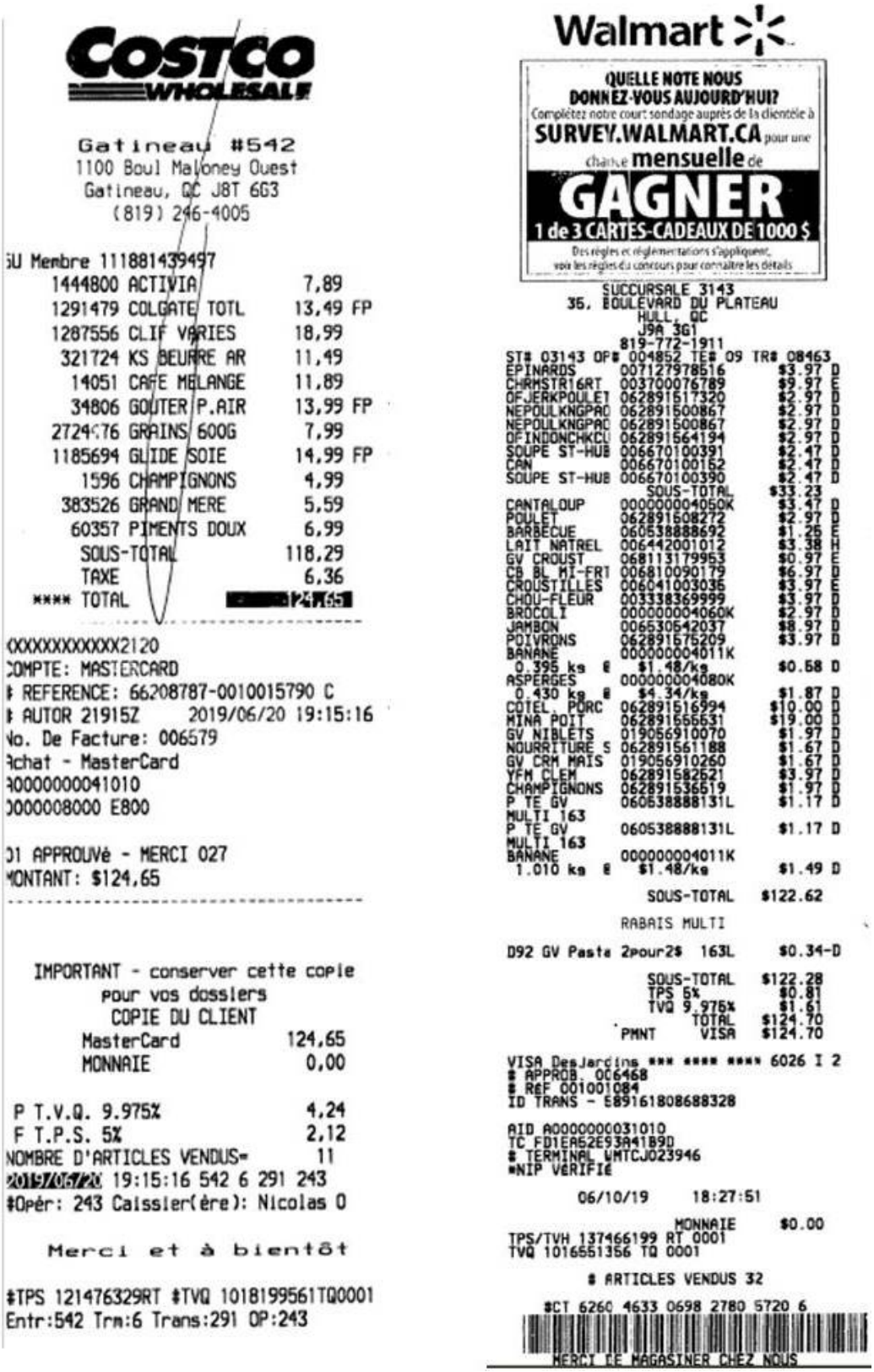


Figure 23: (Examples of Canadian receipts in French. Note: dummy data for illustration purpose only, not extracted from the SHS survey data.

- Total number of items in common (manual coding and OCR): 4951 (96.2%)
- Total number of missing items : 196 (3.8%)
- Total number of extra items : 1900
- Similarity of descriptions of items in common: 97%
- Accuracy of store name : 99.2%
- Accuracy of day of purchase : 87.8%
- Accuracy month of purchase: 89.4%

To assess how well the method performs on other languages than English, we test a number of French receipts. Figures 24 shows example of Canadian receipt in French being OCR'ed. Relevant lines of text are retrieved and parsed into a dataframe of variables including UPC barcodes, item descriptions, prices, shop names, payment modes, dates. The Python codes need to be adapted to handle differences between languages such as price formatting, the dot is used in English as decimal separator instead of comma in French and Dutch receipts.

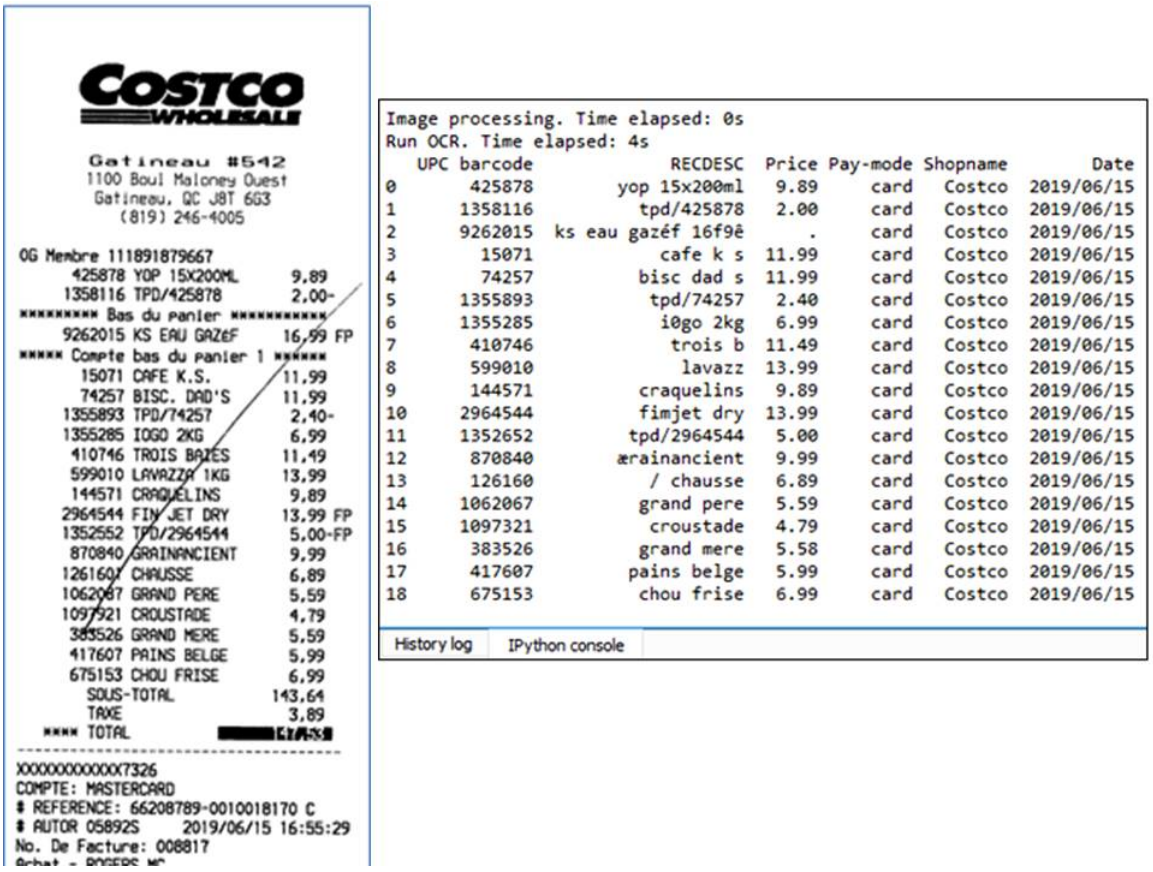


Figure 24: OCR and data parsing of Canada receipt in French: irrelevant lines of text are automatically filtered (e.g. 'Bas du panier', 'Compte bas du panier').

5.5 OCR accuracy flatbed scanner versus mobile app

In this section, we propose to compare OCR performance for various scanning methods. As shown in Figure 25, the same set of receipts are scanned in 3 different ways, using a mobile phone app, a

regular office scanner and a high performance flatbed scanner. Examples of result images are shown in Figure 26, we can see a clear difference in quality between images scanned with a regular office scanner and a high performance scanner. Their respective OCR results will inform decision whether or not there is a need for the agency to invest in a better scanner. In this test, we use Dutch receipts, which allows at the same time to test OCR method on a new sample of non-UK receipts. Because it is time consuming to type out the receipt gold-standard, we limited the test to only 25 receipts. The aim is to demonstrate the concept of how comparison can be done, but the test result is unlikely to be meaningful.

Table 5 shows test results based on a small dataset of 10 Dutch receipts, comparing OCR accuracy across various scanning methods. Although we should not draw conclusion from such a small test, the result seems to indicate that the quality of the scanner does play a role. Mobile app seems to perform surprisingly well in terms of accuracy, which may be explained by the fact that photos were taken by our development team who knew how to make good photos so the text lines were not too distorted. Additionally the average image size of the photo hover around the 1.8MB, while those of the scanner are 500KB on average, which gives the photo significantly more pixels to work with. More extra lines can be found both in office scanning and mobile scanning, which can be explained by the fact that when the images are noisy or in low contrast, Tesseract may struggle more to recognise characters correctly. Therefore, the OCR outputs contain more spelling mistakes. Because our data parsing method relies on keyword matching, it is heavily affected by misspelling. In order to find an indication of the effect of the number of pixels in the photos compared to the scanners, an additional set of 15 annotated receipts was included to compare regular flatbed scanning with the mobile app. For this comparison, the mobile app images were reduced to match the length of the scanner images, while maintaining the proper aspect ratio for the width. Although the set is not large enough to draw conclusions, one can see that the number of pixels has a noticeable influence on the number of extra/missing lines, while the accuracy of properly detected lines stay pretty much equal. Further research into the effects of the pixel count will be left to future work.

	Mean accuracy	Extra lines (%)	Missing lines (%)
Performant flatbed (10)	0.92	+0.12	-0.19
Regular flatbed (10)	0.88	+0.57	-0.42
Mobile app (10)	0.93	+0.76	-0.31
Regular flatbed (25)	0.91	+0.43	-0.22
Mobile app (25)	0.93	+0.24	-0.10
Mobile app Reduced (25)	0.93	+0.48	-0.28

Table 5: Comparison of OCR accuracy across various scanning methods. Test results based on a small dataset of 10 Dutch receipts, with 162 lines of products/prices on receipts. And a dataset of 25 Dutch receipts, with 335 lines. Do note that for *Mobile app* the resolution is significantly higher. *Mobile app reduced* has its resolution reduced to match that of the flatbed scanner.

Further tests need to be conducted on a larger dataset to paint a more truthful picture of how well OCR performs across various scanning methods. The current work serves only as an example to demonstrate how comparative test can be done, to design the concept of the test and to develop the corresponding Python codes.

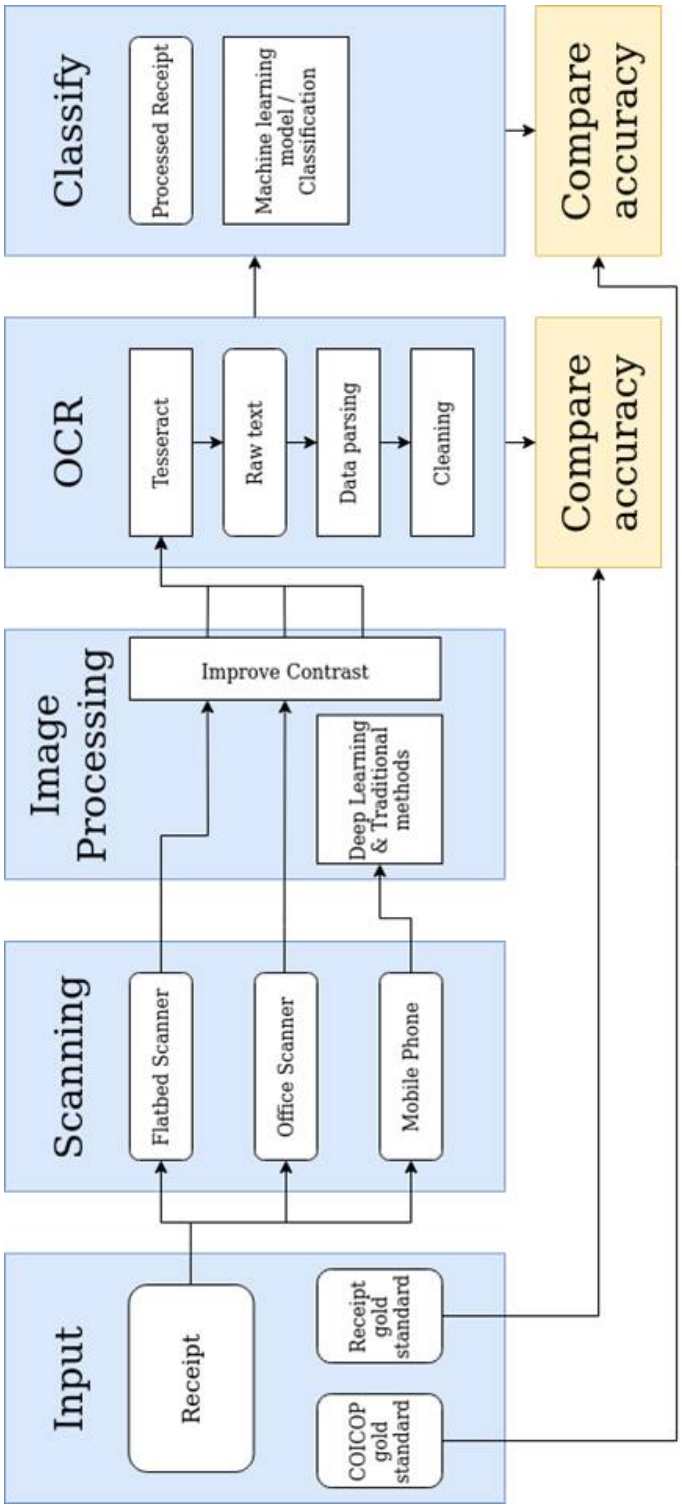


Figure 25: Diagram explaining the test to compare OCR accuracy across various scanning methods. The same set of receipts are scanned in 3 different ways, using a mobile phone app, a regular office scanner and a high performance flatbed scanner. Only basic enhancement are applied to increase the contrast of images scanned with a flatbed scanner, whilst receipts need to be cropped out from photos for mobile phone scans. OCR is applied and accuracies compared across the three scanning modes.

(a) High performance scanner.

(b) Regular office scanner.

(c) Mobile phone app.

Figure 26: Receipt scanned in three different ways for comparing OCR performance across various scanning methods.

6 Machine Learning classification

Text classification is a machine learning approach which can be used to classify sentences, documents or plain text into one or more defined categories or classes. It is a widely used natural language processing task playing a vital role in spam filtering, sentiment analysis, categorisation of news articles and many other domains. There are mainly two machine learning approaches for text classification: supervised approach, where predefined category labels are provided for training, and unsupervised text classification, where the classification needs to be performed entirely without reference to additional labels or classes [Miro?czuk and Protasiewicz (2018)].

For our present task, the goal is to automatically categorise purchased items into their respective 5-digit COICOP codes, which can be achieved with supervised Machine Learning (ML). This is a multi-class classification problem where the assumption is each receipt item is assigned to one and only one class: a fruit can be either an orange or a pear but not both at the same time [Miro?czuk and Protasiewicz (2018)]. However, it is vital to point out that in this case study, the problem is highly specific and labelling data requires domain knowledge. Indeed, the coding frame comprises of over 300 categories where the distinction between one class and another is not always obvious to untrained eyes. For instance, *'Warburtons whole white loaf'* belongs to COICOP category *1.1.1.1.2* while *'white loaf sliced premium'* should be labelled as *1.1.1.1.3*. The difference is based on whether the bread is sliced or unsliced, and therefore common unsupervised classification based methods such as k-means clustering alone will not be suffice for our purpose.

6.1 Feature engineering

The first step in text classification is creating numerical features from text data. These range from basic word counts to more complicated deep learning based methods capturing the context of a word in a piece of text. We used two different methods to create numerical features from text data *i.e.* Count vectorisation and Term Frequency-Inverse document frequency (TF-IDF) [Kowsari et al. (2019)]. Count vectorisation is one of the simplest ways to encode text information into a numerical representation where the number of times every word in the total corpus (collection of all receipt items) is found in a given text (receipt item) is used as the vector. Term Frequency-Inverse document frequency builds the vectors as it attempts to give more prominence to more important words. TFIDF optimises the vector space and changes the impact of specific words depending on how they appear in the receipt item. As one example, a word that occurs very frequently across all classes tells us very little when we find this word in a new receipt item and the significance of this word is therefore reduced. TF-IDF does this by introducing a weighting to the words based on how commonly they appear in all the receipt items. Those less common but potentially more important words will be up-weighted by the inverse document frequency. In this work we have used the Scikit-Learn implementation of the CountVectorizer and TF-IDF at word level (TFIDF-w) and at character level (TFIDF-c) methods to construct feature vector from receipt items [Pedregosa et al. (2011)].

6.2 Supervised learning

In order to match human judgements on receipt items to COICOP classification, a supervised machine learning text classification approach should use features created from the text descriptions of receipt items. A good supervised classifier should learn rules to allocate the data into provided COICOP categories. We explored a range of machine learning classifications models for the purpose of text classification: Naive Bayes model Multinomial, Logistic Regression, support vector machine SVC with linear kernel, the ensemble classifiers such as Random forest, Ada Boost Classifier, Extra Trees Classifier and the Decision Tree Classifier. Again, we have used Scikit-Learn implementation of the above models. Each model comes with a degree of interpretability and has their individual pros and cons in terms of training time, generalisation to unseen data, chances of over-fitting. For instance, Naive Bayes and Logistic Regression are easy to interpret predictive analysis algorithms based on the concept of probability. On the other hand, Decision Tree closely mimics a flowchart. The tree is built up of branches and nodes. At each node, a decision rule will split the data in two and this will then continue to the next node and the next. The random forest is an ensemble model built on decision trees. Support vector machine is computationally more expensive to train and it operates by finding the hyperplane(s) that divides the data into the required categories with the largest margin between the hyperplane and the data. For further details of these methods please refer to [Aggarwal (2014)].

While Scikit-learn offers a wealth of generic machine learning approaches and makes it really easy to experiment with various models and parameter settings, it is not uncommon to train neural networks for the purpose of text classification [Kowsari et al. (2019)]. FastText is one such popular neural net library developed by Facebook. The library is an open source project on GitHub, and provides text classification methods for both supervised and unsupervised learning. FastText has gained a lot of attention in the machine learning community as it is able to learn low-dimensional representations for all features in a text, and then average these to a low-dimensional representation of the full text. In this work we have used FastText for supervised [Joulin et al. (2016)] and unsupervised embeddings for feature engineering [Bojanowski et al. (2016)] for text classification.

6.3 Ensemble voting

In this work we have explored a range of supervised text classification models. However, not all models are suitable for all datasets and have varying levels of complexity and accuracy. The idea behind the ensemble learning is to combine conceptually different machine learning classifiers and use a majority voting scheme (hard vote) or the average predicted probabilities (soft vote) to predict the class labels. Such an approach can be useful to balance out individual weaknesses of different classifiers. In majority voting, the predicted class label for a particular sample is the class label that represents the majority (mode) of the class labels predicted by each individual classifier. In contrast to majority voting, soft voting returns the class label as argmax of the sum of predicted probabilities. As we will show in the text classification performance section, using an ensemble learning strategy to make final predictions leads to an impressive improvement in the performance of machine learning text classification. We have used VotingClassifier which is a Scikit-Learn implementation to incorporate hard/soft voting based prediction making [Pedregosa et al. (2011)].

6.4 Active Learning

Active learning is a special case of machine learning in which an algorithm is able to interactively query human (or some other information source) to obtain the desired outputs at a new classification query. Active learning is a key component in HuIL where human and machine intelligence combine to create more accurate AI. In such systems, humans are involved in every stage of the process by creating a feedback loop from training to testing stages resulting in a more accurate model, as shown earlier in Figures 6. HuIL is a blend of supervised learning (using labelled training data) and active learning (interacting with users for feedback).

Let us describe how the process works by examining an example. If we consider again the case where the description simply says *'fresh milk'*, we don't know whether it is *'whole milk'*, *'skimmed milk'* or *'semi-skimmed milk'*. The confusion comes from the fact that the word *'fresh milk'* comprises two very generic terms that belong to many possible products e.g. *'fresh meat'*, *'fresh bread'*, *'milk chocolate'*, *'cleansing milk'*. We expect ML models to make predictions with low confidence scores and therefore, the prediction is rejected and the item is sent to human for inspection. In the same fashion, ML models would classify rare and unseen products with low confidence scores because such items are either completely absent or exist in few samples in the training set, so the model has not sufficiently learned to recognise them. These cases are flagged up and sent to human for re-labelling, as summarised in Figure 27.

In case of rare and unseen products, the re-labelled data can be used to retrain the models and make them more up-to-date, this is called Active learning. However, in case of ambiguous items such as *'fresh milk'*, the coder has to contact the respondent for clarification, which increases respondent burden, workload and processing time. One way to mitigate this problem is to include a *'Usual Purchases'* page in the questionnaire, asking respondents what kind of *'milk'* they preferably buy, so that *'fresh milk'* can be imputed, as shown in Figure 28. The Usual Purchases can be implemented simply as a look up table. Separate research is being conducted by the LCF team to identify regular products to feature on the Usual Purchases page of the survey questionnaire.

6.5 Classification performance

There is certainly more than one way to assess a machine learning classifier performance and often one single metric is not sufficient to capture the quality. In our present case, machine learning text classification is a multi-class classification problem so an ideal performance metric should reflect

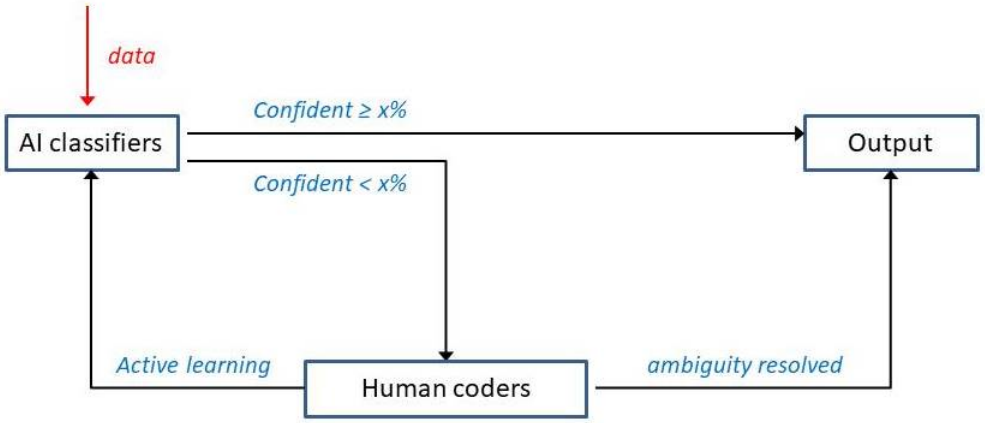


Figure 27: Regular products are classified with high confidence, whereas rare/unseen/ambiguous products are classified with low confidence. If the confidence is less than the cut-off value, the prediction is rejected and sent to human for re-labelling. The new labelled data is used to retrain the models.

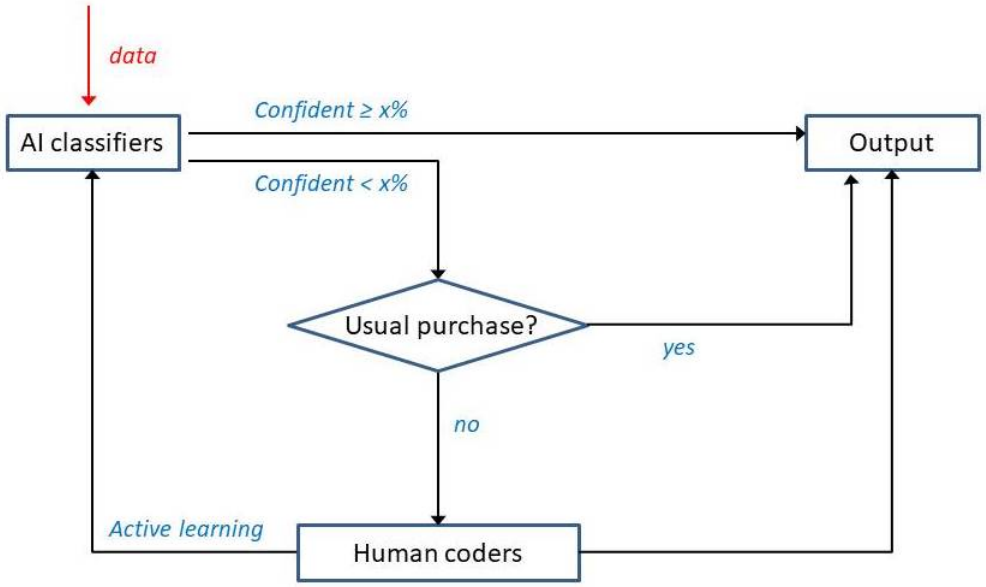


Figure 28: If too many items are sent to the coders, we will not make good efficiency savings and there is a risk of increasing respondent burden. One way to mitigate this problem is to use a dictionary of *'Usual Purchases'* to impute for missing data when it is possible.

the performance of a classifier across all the classes. Furthermore, in most real life situations and certainly in our training datasets we have an imbalanced dataset with unequal amounts of receipt items in each COICOP category. For example, 'bread' and 'milk' are likely more often bought than, say 'vodka' -there is a risk of class imbalance, which is magnified in multi-class classification problems [Leevy (2018)]. The model may perform well on dominant classes but badly on classes that are under-represented in the training set, thus using accuracy alone as a performance metric can be misleading. We must therefore use performance metrics that takes into account this class imbalance. Therefore, in this study, we propose to use multiple metrics (e.g. accuracy, precision, F-score, recall) [Davis and Goadrich (2016)] to capture a more complete picture of the classifier performance and we also define a bespoke performance metric that is more pertinent from a business perspective which we will describe later in this section. We evaluate accuracy, precision, F-1 score, recall in a one-versus rest comparison for each label to assess the quality of the classifier for each item in our data.

Accuracy is one possible metric for evaluating classification models. Informally, accuracy is the fraction of predictions our machine learning model got right. Precision and Recall answer complementary but important questions, precision captures for a given class what proportion of predictions is truly positive. Recall tells us for a given class what proportion of actual positives is correctly classified. There is a trade-off between precision and recall and F1-score is a way to combine precision and recall into a single number. F1-score is computed using a harmonic mean of precision and recall. In a multi-class classification setting, accuracy, precision, F-1 score and recall can all be computed for each individual class and a weighted score can be arrived at for each metric where the metric score of each class is weighted by the number of samples from that class. We made use of classification-report function available in Python's Scikit-Learn library to easily compute accuracy, precision, recall and F-1 score for each class. It is worth pointing out that it is also possible to calculate "macro" averaged score for accuracy and other metrics, which gives equal weights to each class. In problems where infrequent classes are nonetheless important, macro-averaging may be a means of highlighting their performance. On the other hand, the assumption that all classes are equally important might not be always untrue, such that macro-averaging will over-emphasise the typically low performance on an infrequent class. In this work we have therefore reported weighted scores of accuracy, precision, F-1 score and recall where weights account for weights of the class labels in our dataset. We take a dataset that has been labelled by LCF coders and split it into a training set (80%), validation set (10%) and test set (10%). The training set is used to train a ML model, validation set provides an unbiased evaluation of a model fit on the training dataset while tuning models' hyperparameters and test data is used to provide an unbiased evaluation of a final (fitted) model.

It can often be the case that some parts of the dataset set are easier or harder to classify than other parts, so the above train-val-test split can not always guarantee an unbiased performance assessment of a machine learning model. We therefore also make use of k-fold cross validation technique to avoid the risk of getting a misleading view of the performance of a classifier. The process of cross validation is equivalent to shuffling the data, dividing it into N equally sized chunks and then training the model on N-1 of these parts and using the last part as test data to compare against. The training and testing is then repeated while choosing a different chunk as the test data, and using the rest as training data until the entire data set has been used as test data once. The result for each part of test data (k^{th} fold) is then returned as an array that can be averaged to get a single performance metric. Of course, this comes at an increased computational cost but avoids the danger of reporting misleading performance of a machine learning model. It is possible to use *cros-val-score* function offered by Scikit-Learn to easily evaluate cross validations scores for performance metrics of different ML models.

Almost all machine learning models require a series of hyperparameters to operate and supervised machine learning classifiers we have considered in this work are no exception. Scikit-Learn machine learning classifiers come with default values for hyperparameters which are not always optimal.

Table 6: Performance metrics of various Machine Learning classifiers. We use weighted scores available from Scikit-Learn's classification-report to account for class imbalance [Pedregosa et al. (2011)]. Best performance metrics are highlighted in bold.

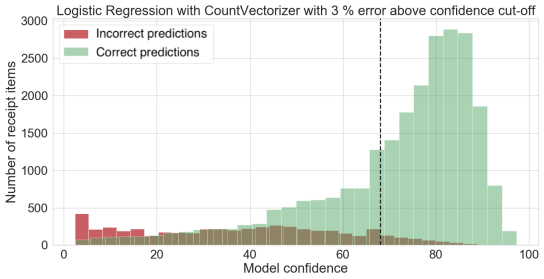
Machine learning classifiers performance				
ML Model	Accuracy	F-1 score	Precision	Recall
LR/CV	0.83	0.82	0.83	0.83
LR/TFIDF-w	0.79	0.78	0.80	0.79
LR/TFIDF-c	0.81	0.80	0.81	0.81
RF/CV	0.83	0.83	0.83	0.83
RF/TFIDF-w	0.80	0.80	0.81	0.80
RF/TFIDF-c	0.83	0.83	0.83	0.83
NB/CV	0.74	0.72	0.76	0.74
NB/TFIDF-w	0.73	0.70	0.74	0.73
NB/TFIDF-c	0.73	0.71	0.73	0.73
DT/CV	0.82	0.82	0.83	0.82
DT/TFIDF-w	0.80	0.80	0.80	0.80
DT/TFIDF-c	0.81	0.81	0.81	0.81
SVM/CV	0.84	**0.85**	0.84	0.84
SVM/TFIDF-w	0.80	0.80	0.81	0.80
SVM/TFIDF-c	**0.85**	0.84	**0.85**	**0.85**
FastText	**0.85**	**0.85**	**0.85**	**0.85**
Soft Voting	**0.85**	**0.85**	**0.85**	**0.85**

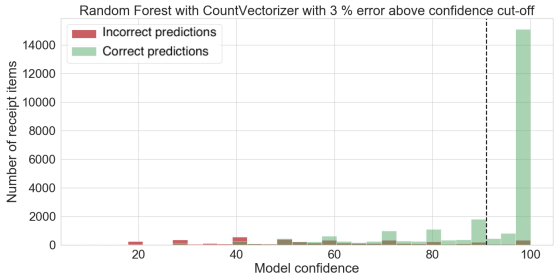
Scikit-learn offers *GridSearchCV* to perform an exhaustive search on a dictionary of parameter values. This is then followed by a k-fold cross validation test to retrain the model on k different configurations of data per configuration of parameters. For a model configuration with x parameters with y possible values each, we would need to train the model $(k * x * y)$ times in total, which means it grows in complexity exponentially very fast. The cross validation score is used to compare every configuration against each other and the top scoring configuration is returned as the result. Scikit-learn also provides *RandomizedSearchCV* which is computationally less expensive to operate as it performs a random search for an optimal set of parameter values.

In the interest of minimising computational costs, we experimented with 5-fold cross validation and GridSearchCV on a small subset of ML models (Logistic Regression and Random Forest) and we found the performance metrics for those models were very similar to the metrics reported in this work where we have evaluated ML models on a test dataset following a train-val-test split with no additional hyperparameter optimisation. Performance metrics of various Machine Learning classifiers on the test dataset is shown in Table 6. It is evident from Table 6 that the performance of Support Vector Machine model in conjunction with a Countvectoriser feature extraction (CV) and TF-IDF at character level (TF-IDF-c) perform optimally. Also, Logistic Regression model in conjunction with a Countvectoriser feature extraction (CV) or a Random Forest model (RF) with a TF-IDF at character level (TF-IDF-c) perform competently well. It should be noted that even without an elaborate hyperparameter optimisation performed on the full training dataset, majority voting is able to ensure that the weakness in the individual performances of various classifier can properly complement each other as a whole and together yield better performance compared to the individual models alone as shown in Table 6.

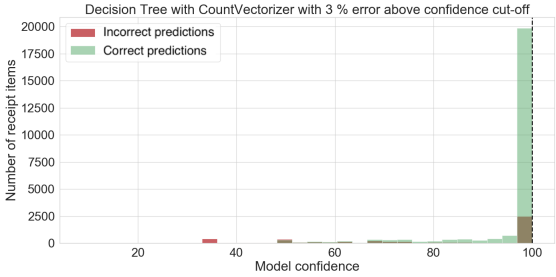
In addition to experimenting with various models and parameter settings readily available in Scikit-learn, we have also tested state-of-the-art ngram word embedding methods such as Embeddings from Language Models (ELMo) [Peters et al. (2018)] and FastText for text classification [Joulin et al. (2016)]. FastText was tested both as a supervised model and for obtaining pre-trained word embeddings for converting text data into vector model model [Bojanowski et al. (2016)]. We found that FastText embeddings with Logistic Regression yields an accuracy score of 68%, while ELMo word-embedding with Logistic Regression shows promising initial results but is computationally more expensive- so we did not explore them further. FastText as a supervised model performed exceedingly well and the corresponding performance metrics on the test data are provided in Table 6. FastText's neural network architecture is capable of learning similarities between different yet related features. For example, the model might learn that *skm mlk* and *skim milk* are related, and should classify them to the same class.

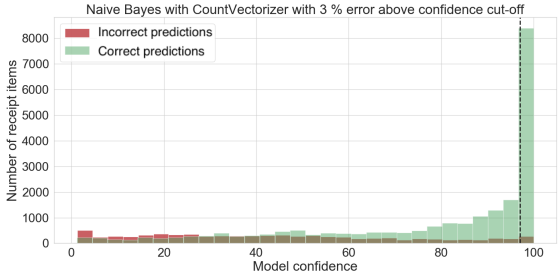
In the end, we believe that choosing the best model for text classification depends on a number of factors, low computational cost to train the model, scalability, acceptable performance metrics, interpretability and ease to use to mention a few. An ideal text classifier which meets business needs should score high on most if not all of these factors. If we measure the processing times of individual tasks - e.g. the time to OCR one average receipt, and to classify N items to COICOP - we can deduce the overall processing time taken by AI to perform the tasks, which can be then compared to the processing time by human. Of course, these are only estimates that may be sufficient to help build a business case to take the research further. More truthful measures will be collected by conducting real pilot test.



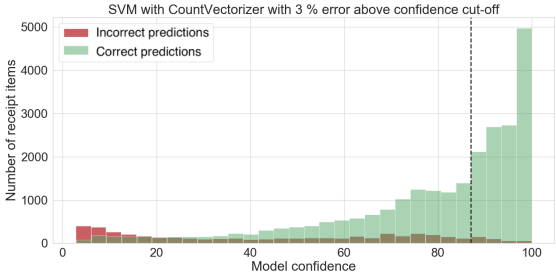


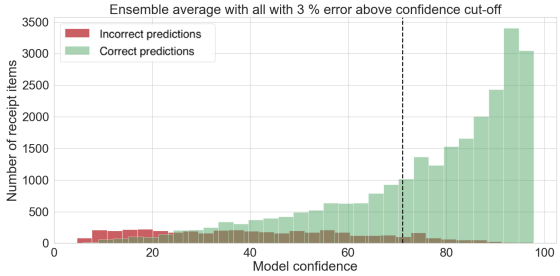
(a) Percentage of automation with 3% error rate: 59% **(b) Percentage of automation with 3% error rate: 57%**





(c) Percentage of automation with 3% error rate: 28% **(d) Percentage of automation with 3% error rate: 0%**





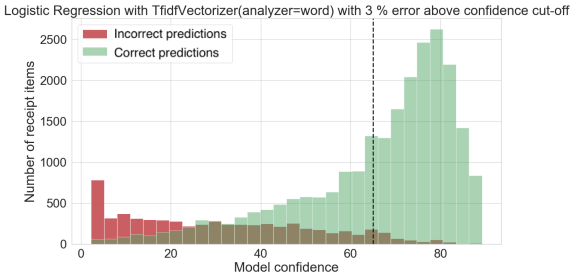
(e) Percentage of automation with 3% error rate: 44% **(f) Percentage of automation with 3% error rate: 61%**

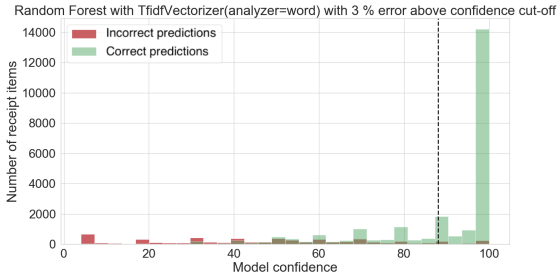
Figure 29: Percentage of automation for a chosen error rate for different machine learning models and feature extraction using CV.

7 Measuring success

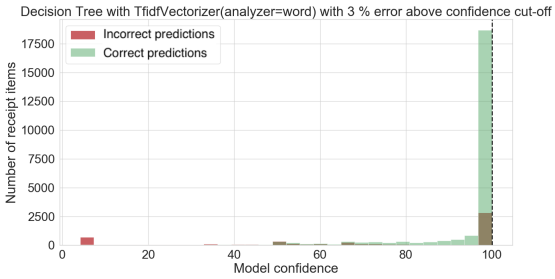
7.1 Formal definition of success

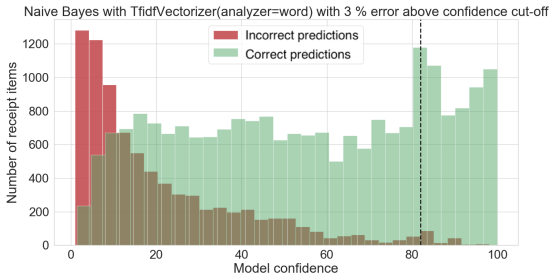
Within the data science community, assessing and comparing different classification models using measures such as *accuracy*, *precision*, *F-1 score*, *recall* make a lot of sense. From a business perspective, however, such quantities are not meaningful. For a business to invest in replacing its legacy system, potential benefits are usually measured in terms of *efficiency savings*, *production costs*, *processing time*, *data quality*, and in the context of official statistics, *respondent burden*. Often, it is about finding a trade-off between these variables. Is there a way to translate model performance into business measures? We have thus designed a model performance metric more relevant to the business needs which can be designed as follows. Different machine learning classification models are trained and their individual performance is evaluated on test dataset. For each individual model, the final prediction outcomes is separated into two categories: (1) Items where model predictions match human labelling and (2) Items where model predictions is different to human labelling. Every prediction is associated with a model confidence score, we plot the histogram as shown in Figure 32.



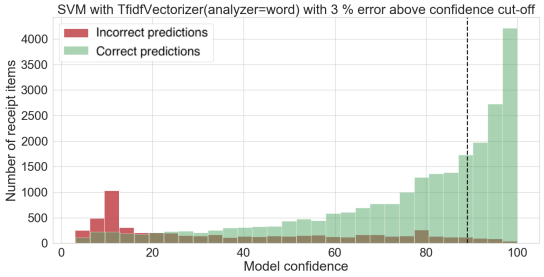


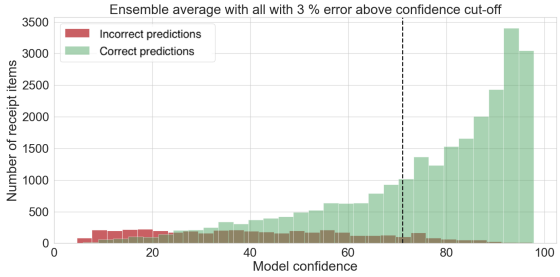
(a) Percentage of automation with 3% error rate: 52% **(b) Percentage of automation with 3% error rate: 60%**





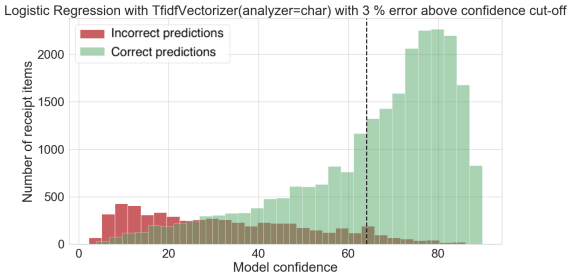
(c) Percentage of automation with 3% error rate: 18% **(d) Percentage of automation with 3% error rate: 0%**



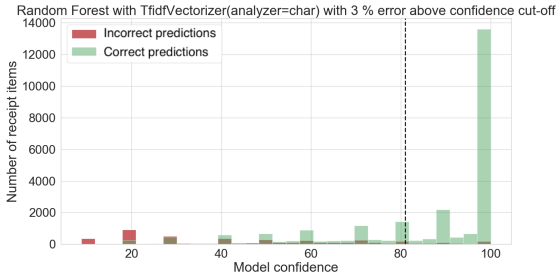


(e) Percentage of automation with 3% error rate: 34% **(f) Percentage of automation with 3% error rate: 61%**

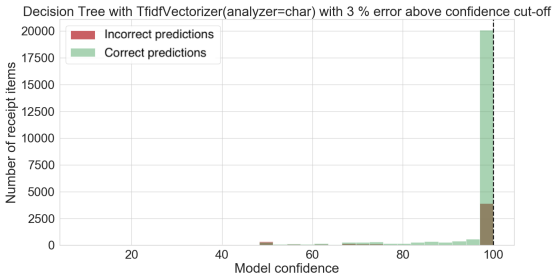
Figure 30: Percentage of automation for a chosen error rate for different machine learning models and feature extraction using TF-IDF-w.



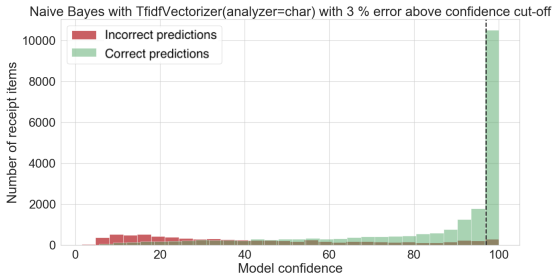
(a) Percentage of automation with 3% error rate: 54%



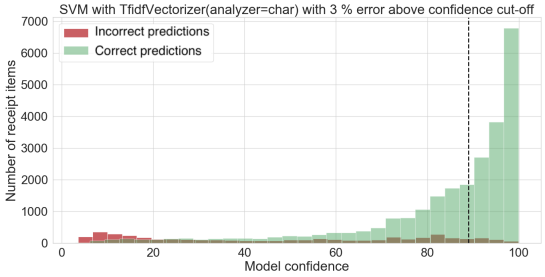
(b) Percentage of automation with 3% error rate: 61%



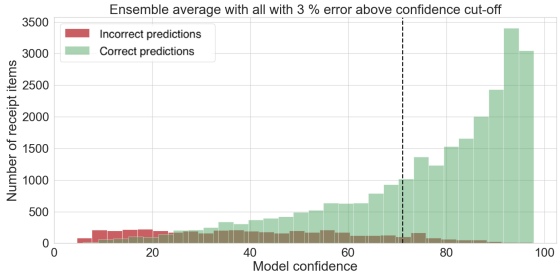
(c) Percentage of automation with 3% error rate: 35%



(d) Percentage of automation with 3% error rate: 0%



(e) Percentage of automation with 3% error rate: 49%



(f) Percentage of automation with 3% error rate: 61%

Figure 31: Percentage of automation for a chosen error rate for different machine learning models and feature extraction using TF-IDF-c.

If we now define a confidence cut-off, say $x\%$: items that fall on the right hand side of the cut-off line are automatically accepted, items that fall on the left hand side are sent to human. We can see from the graph that there is a small proportion of mis-classified items that slip through. In the context of the UK, the LCF team produces data for other government agencies, with whom they agree on an acceptable level of data quality measured in terms of error rate. If we assume that the coders always classify items correctly, anything on the left of the cut-off line is 100% accurate because they are checked by human. The proportion of mis-classified items on the right hand side should not exceed the error rate agreed with the end users. Knowing this error rate, we can determine the cut-off value and estimate the percentage of automation.

Figure 32: Linking model performance metrics to success measures from a business perspective.

7.2 Test results

Business performance metrics for a range of error thresholds are shown in Table 7 for various ML classifiers. As expected, percentage of automation increases monotonically with the error threshold. Logistic Regression model (LR) in conjunction with a Countvectoriser feature extraction (CV), or a Random Forest model (RF) with a TF-IDF at character level (TF-IDF-c) result in 69% automation for an error rate of 5%. A maximum automation rate of 78% can be achieved using a FastText model for an error rate of 5%. On the other hand, soft voting based VotingClassifier outperforms every other model with maximum automation rate achievable in the low error rate regime ($\leq 3\%$). Example model characteristics for different machine learning classifiers in conjunction with CV, TF-IDF-w and TF-IDF-c are shown in Figure 29, Figure 30 and Figure 31 respectively.

8 User Interface

"People ignore design that ignores people."
— Frank Chimero, Designer

8.1 The human factor and user story

Replacing a legacy systems is more than merely replacing the software, the human factor is a key challenge. How will the coders, who are accustomed to manual tasks, react to the complexity

Table 7: Percentage of automation for a given error rate. This is a business decision to find a trade-off between *data quality* (error rate) and *efficiency savings* (% of automation). Different government agencies may agree on different error threshold. Maximum automation rate achievable for a given error threshold is shown in bold.

Automation rate as a function of error rate (er)					
ML model	er=1%	er=2%	er=3%	er=4%	er=5%
LR/CV	31%	51%	59%	65%	69%
LR/TFIDF-w	14%	45%	52%	58%	64%
LR/TFIDF-c	16%	47%	54%	61%	66%
RF/CV	0%	0%	57%	65%	67%
RF/TFIDF-w	0%	52%	60%	64%	70%
RF/TFIDF-c	0%	50%	61%	67%	69%
NB /CV	0%	21%	28%	33%	37%
NB/TFIDF-w	10%	16%	18%	31%	38%
NB/TFIDF-c	0%	27%	35%	42%	47%
DT/CV	0%	0%	0%	0%	0%
DT/TFIDF-w	0%	0%	0%	0%	0%
DT/TFIDF-c	0%	0%	0%	0%	0%
SVM/CV	7%	34%	44%	52%	58%
SVM/TFIDF-w	10%	22%	34%	44%	50%
SVM/TFIDF-c	22%	38%	49%	59%	65%
FastText	0%	0%	**62**%	**72**%	**78**%
Soft Voting	**36**%	**53**%	61%	67%	73%

of AI? If the coders dislike the new system, there will be a negative impact on productivity and team's morale. The UI creates the first impression so it is extremely important that it hides the sophisticated and wearisome machinery. A good UI humanises technology, builds relationship with the users so they will trust the underlying technology through good experience.

To design a system that is fit-for-purpose, we believe that the first step is to understand the users' needs and priorities, therefore we adopted a human-centered design approach. To this end, very early into the project, our data scientists visited the coding team to observe them in their daily tasks to understand the current business process. We mock-up various UI designs with input from User Experience (UX) architect and consult with software developers on feasibility. Then we presented the mockups to a user focus group to gain feedback. Figure 33 shows an example of UI, we take inspiration from the Irish CSO templates and incorporate inputs from the LCF coders and the software developer who built the legacy system. We reproduced as much as possible the look and feel of the LCF legacy interface, using similar screen colours and layout to create a sense of familiarity. The user story is as follows.

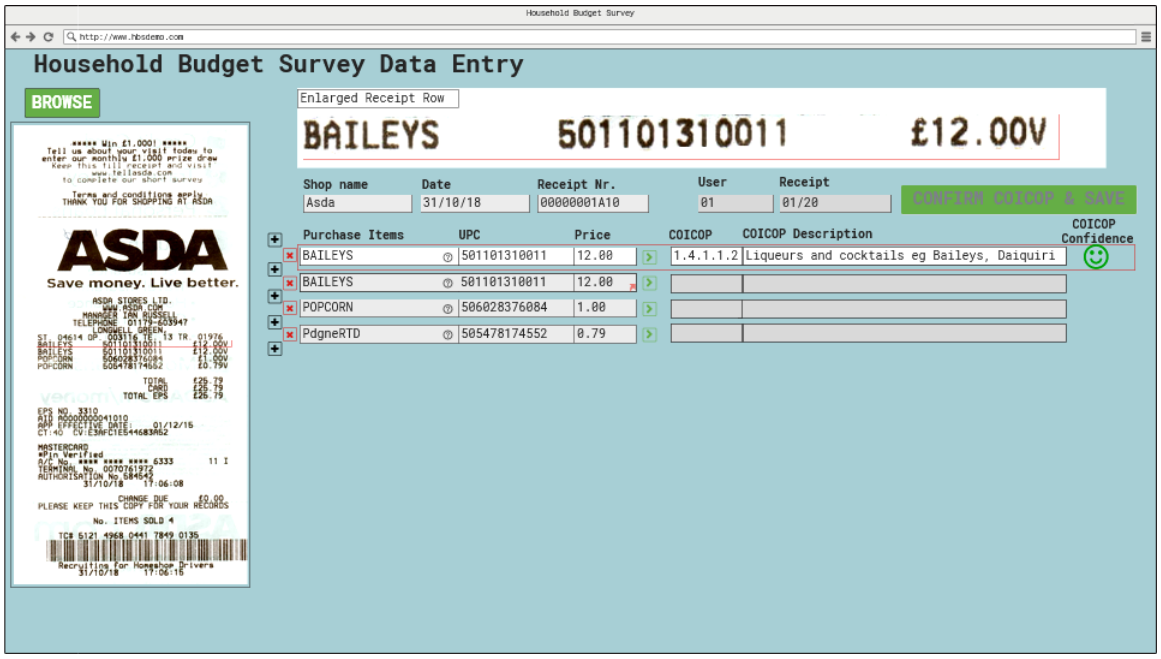


Figure 33: Mockup of the Household Budget Survey Data Entry User Interface. The receipt image is shown on the left, the text fields on the right are prefilled with text extracted from OCR (e.g. shop name, date, items). All fields are editable so the user can correct OCR errors if needed. The currently edited line is highlighted. The snapshot shows the first item being checked for OCR errors and classified to COICOP.

- `Step 1`: the user enters a Welcome screen that shows a Browse button. A message invites the user to browse to the folder where the receipt images are stored. The user browses to the said folder, all images are read into a list. The UI will loop through all receipts in the list.
- `Step 2`: the first receipt is shown on screen. The image is on the left, the information extracted from the OCR output is on the right, as shown in Figure 33. All fields are editable so the user can make corrections if the OCR results are not correct. The currently edited field is highlighted, starting at the top line, showing shop name, date, etc.
- `Step 3`: if there are misspelled words in the OCR output text, the user can correct this. If irrelevant lines are captured, the user clicks on the 'x' button to delete them, and if lines are missing the user clicks on the '+' button to add an empty line to then fill in the missing information.

- `Step 4`: if the image is too small, a message is flagged up in red *'Receipt Unreadable'* and the portion of the receipt currently edited is enlarged, as shown in Figure 34.
- `Step 5`: the user checks if the OCR'ed text is correct, or make corrections otherwise.
- `Step 6`: once OCR corrections are completed, the user clicks on the arrow to classify the item to the COICOP code. This runs the ML classifiers behind the scene.
- `Step 7a`: the models predict the 5-digit code with a confidence score. If $score \geq threshold$ (as discussed in the previous section), a green smiling face appears to confirm success. The next line becomes active and highlighted. The user repeats from Step 6.
- `Step 7b`: if $score < threshold$, a red angry face appears to indicate failure, as shown in Figure 35. The user assigns the correct COICOP code, a green smiley appears to confirm success, as shown in Figure 35. The results will also be saved into a new training set used for updating the models (active learning). The next line becomes active and highlighted. The user repeats from Step 5.
- `Step 8`: when there is no more line to edit, the button 'Confirm COICOP and Save' becomes active and clickable. The user clicks on 'Confirm COICOP and Save'. The next receipt appears on screen. The user repeats from Step 2.
- `Step 9`: the user repeats the same process until there is no more receipts to process. A final message appears on screen to inform the user that the task has been successfully completed.

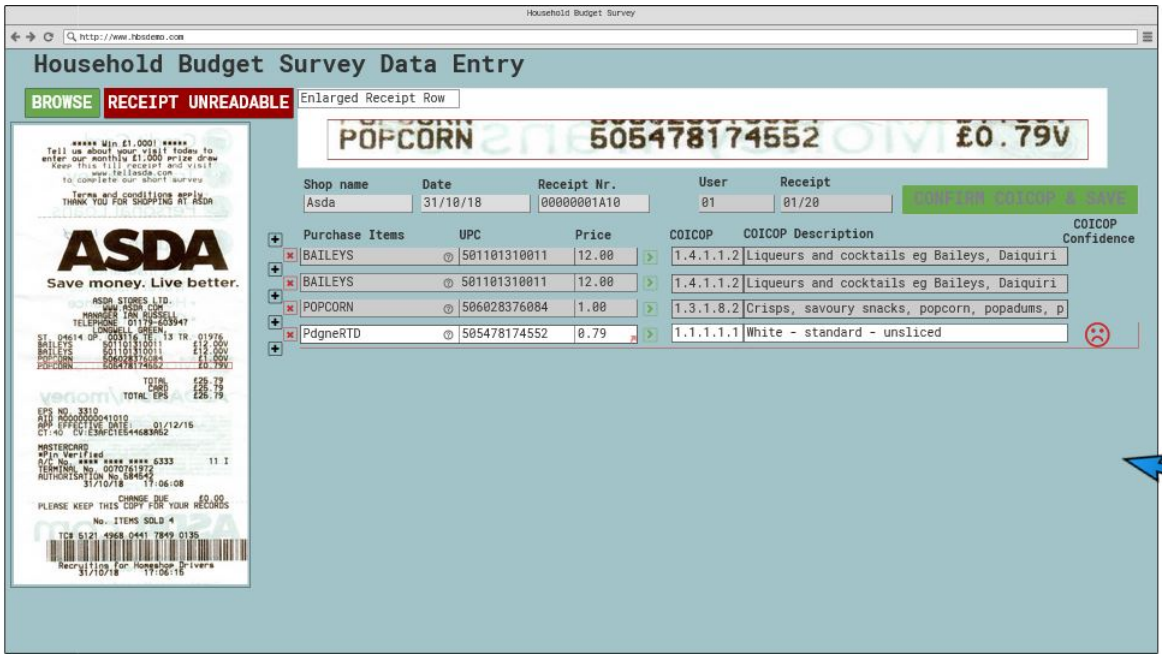


Figure 34: Example screenshot where OCR did not perform correctly. The item description is erroneous, causing the subsequent Machine Learning (ML) classification to fail. As the ML model cannot recognise the item, it assigns a COICOP code with low confidence (lower than the cut-off value). The item is flagged up, requesting human input.

8.2 Design principles

The UI mockup is deployed on Balsamiq Cloud, making it easy for coders to access it online to test and provide feedback. The main purpose is to gently familiarise the users to the new system,

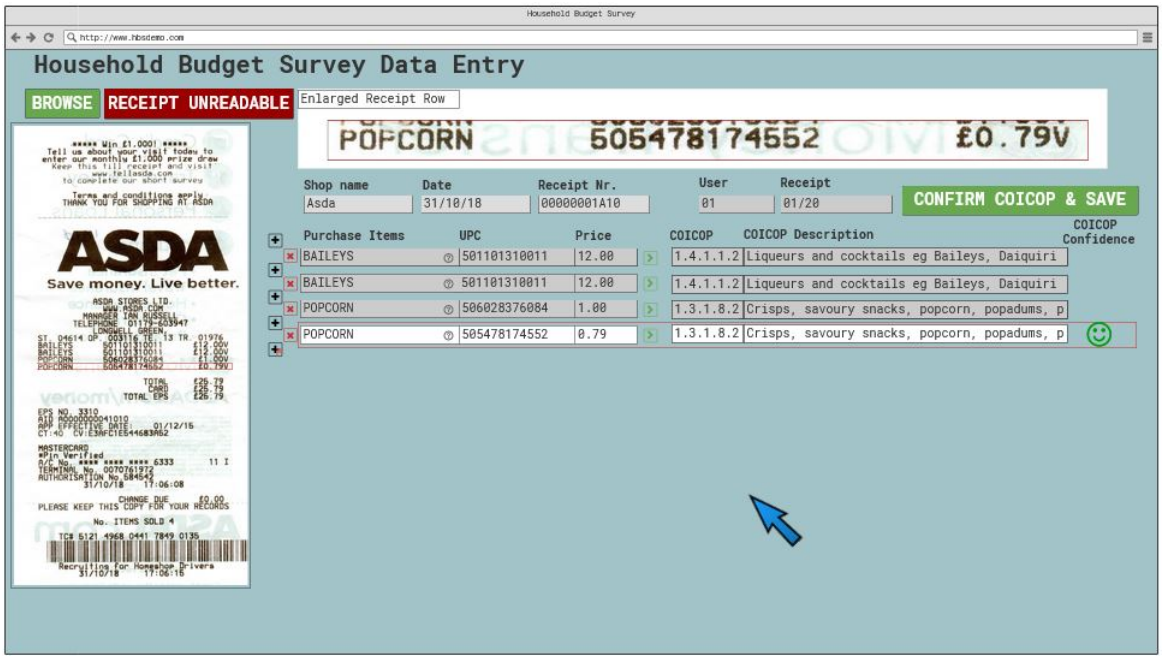


Figure 35: The coder intervenes to manually correct the misspelled description. This time, the ML model can recognise a known item and assigns the correct COICOP code with high confidence. A green smiley appears to confirm success.

which is a great way to ensure we develop a system that is fit-for-purpose. By taking an active participation in implementing changes, the users will take ownership of the final product and trust the underlying technology. We follow best practice in UI design as below [Norman (1990)].

1. `Visibility`: Users need to know what all the options are, and know straight away how to access them. Features should not be hidden 'out of sight'. The UI needs to be kept as simple as possible, every element must serve a purpose.

2. `Feedback`: Every action needs a reaction. Indications could be a sound, a moving dial, a button changing colour to let the users know their actions have been taken into account.

3. `Affordance`: The shape of a feature allows the users to know how to use it. For example, a button invites clicking (to afford means 'to give a clue'). Draw attention to key features using:Color, brightness and contrast. Avoid including colors or buttons excessively. Text via font sizes, bold type/weighting, italics, capitals and distance between letters. Users should pick up meanings just by scanning.

4. `Mapping`: The relationship between controls and their effects. For example, the up and down arrows represent the up and down movement of the cursor. Respect the user's eye and attention regarding layout; focus on hierarchy and readability. Put controls near objects users want to control.

5. `Constraint`: Restricting the kind of user interaction that can take place at a given moment.

6. `Consistency`: The same action has to cause the same reaction, every time. The design should be consistent across the entire application. Consistent sequence of actions, identical terminologies and platform conventions should be followed throughout the application.

7. `Workflow`: Minimise the number of actions for performing tasks but focus on one chief function per page; guide users by indicating actions. Ease complex tasks by using progressive disclosure.

8. `Efficiency`: The screens should load and display content within acceptable amount of time. The more than expected time a user has to wait, more stress is built into the human body causing long term damage. The interface should also have functionality for advanced users. While being non-obtrusive to novice users, accelerators or shortcuts should be available for experienced users.

9. `Visual appeal`: Minimalist and aesthetic design helps user easily consume the data and hence there is less stress on the human mind and increases the 'feel good' factor in the user. Principles of contrast, repetition, alignment and proximity come naturally to humans.

10. `Cognitive loads`: User's memory load should be minimized. All information that a user needs from the application to perform a task should be displayed or easily retrievable.

9 Conclusion and Future Works

In this document, we reported findings from our research conducted as part of Work package 4.2 of the @HBS Project. We seek to automate the processing of shopping receipts and classification of products to COICOP, the aim is to make efficiency savings, speeding up processing time whilst maintaining similar or better data quality. We proposed an end-to-end automation pipeline that comprises the following modules 1-Receipt scanning, 2-Image Processing, 3-Optical Character Recognition, 4-natural Language Processing, 5-Machine Learning classification and explore various options for implementing each module. Our experiments show that in order to maintain the data quality level required for official statistics, pure automation is not realistic. Whilst it is relatively easy to develop AI models that perform at 80% accuracy, it is increasingly difficult to push for the last 20%. As the algorithms become more complex, the system becomes more difficult and more expensive to build and maintain. To mitigate such drawback, we propose Human-in-the-Loop as an alternative AI concept where machine and human intelligence combine, the result is time and resource saving on repetitive, labour-intensive tasks which machines are good at, allowing humans to focus on value added tasks requiring flexibility and intelligence.

The current research aims to build a proof of concept. Preliminary OCR tests were carried out on a small dataset of about 200 receipts obtained from ONS colleagues. Classifications to COICOP were tested on a separate dataset of about 400,000 product items obtained from the UK LCF team. Both tests have shown promising results. The next step is to evaluate if the methods scale up to larger volumes of data and larger variety of receipts. We believe that the concept of the pipeline will hold regardless of the complexity of the problem, but the underlying methods for each modules could be further improved. For example, at the moment, we are unsure how robust is the approach for data parsing using keywords. This method performs best compared to all approaches we have tested so far, but this requires further investigations. We are in the process of handing over the project to the survey team and help build data science capability. We recommend that the team keep testing our methods on larger datasets, new problems will be identified and resolved. This way, we incrementally develop a system that is fit-for-purpose.

The primary motivation for this work is to make efficiency savings and speed up processing time without increasing respondent burden or degrading data quality. In this research, we have performed *'lab simulations'* to show the potential of the solution, but pilot tests need to be carried out in real-world conditions so we can collect realistic numbers. We have also made assumption and simplification that are not entirely realistic. For example, we benchmarked AI algorithms against human performance, model accuracy was measured using human outputs as the ground-truth, which ignored the fact that humans also make mistakes. For future research, we recommend that the survey team collects information on human error rates so we can conduct a more truthful comparison.

Last but not least, we wish to share knowledge and collaborate more widely with other agencies as we have already started to do so with some National Statistical Institutes. Our solution was developed mainly in the context of the UK but we have made effort to keep it as generic as possible so the methods can be adapted for other countries. All the Python codes will be made publicly available on the ONS Data Science Campus Github repository as well as this report where we report on solutions that have shown potential as well as failed attempts, in the hope that it will help others avoid pitfall.

References

Abdulla, W. (2017). Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`.

Aggarwal, C. C. (2014). *Data Classification: Algorithms and Applications*. Chapman & Hall/CRC, 1st edition.

Archives, T. U. N. (2017). General hints and tips for digitisation for business use. *Guidance and Best Practice*.

Beyeler, M. (2017). Machine learning for opencv: Intelligent image processing with python. *Packt Publishing Ltd.*

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Christoph Wick, Christian Reul, a. F. P. (2018). Calamari - a high-performance tensorflow-based deep learning package for optical character recognition. *Guidance and Best Practice*.

Davis, J. and Goadrich, M. (2016). The relationship between precision-recall and roc curves. *Proc. of the 23rd International Conference on Machine Learning*, pages 233–240.

Faith, C. L. (2008). A framework for reasoning about the human in the loop. *Proceedings of the 1st Conference on Usability, Psychology, and Security.*

Gil, M., Pelechano, V., Fons, J., and Albert, M. (2016). Designing the human in the loop of self-adaptive systems. *International Conference on Ubiquitous Computing and Ambient Intelligence.*

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., and Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4).

Leevy, J.L., K. T. B. R. e. a. (2018). A survey on addressing high-class imbalance in big data. *J Big Data*, 5:42.

Martin Abadi, A. A. (2015). Tensorflow:large-scale machine learning on heterogeneous distributed systems. *Google Research*.

Miro?czuk, M. M. and Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36 – 54.

Norman, D. A. (1990). The design of everyday things. *New York: Doubleday Publishing Group*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

R., P. and Thomas, S. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans*.

Releases, T. (2019). Tesseract releases. *https://github.com/tesseract-ocr/tesseract/releases*.

Rothrock, L. and Narayanan, S. (2011). Human-in-the-loop simulations: Methods and practice. *Springer*.

Sharma, A., Shrimali, V. R., and Beyeler, M. (2019). Machine learning for opencv 4: Intelligent algorithms for building image processing apps using opencv 4, python, and scikit-learn. *Packt Publishing Ltd.*

Smith, R. (2007). An overview of the tesseract ocr engine.

Tomaschek, M. (2018). Evaluation of off-the-shelf ocr technologies. *Bachelor Thesis - Masaryk University*.

Wenchao, L., Dorsa, S., Shankar, S. S., and A., S. S. (2014). Synthesis for human-in-the-loop control systems. *Tools and Algorithms for the Construction and Analysis of Systems*.

Witten, I., Bell, T., Emberson, H., Inglis, S., and Moffat, A. (1994). Textual image compression: Two-stage lossy/lossless encoding of textual images. *Proceedings of the IEEE*.