Human-in-the-Loop AI in Government: A Case Study

Lanthao Benedikt
Data Science Campus
Office for National Statistics, UK
lan.benedikt@ons.gov.uk

Chaitanya Joshi
Data Science Campus
Office for National Statistics, UK
chaitanya.joshi@ons.gov.uk

Louisa Nolan
Data Science Campus
Office for National Statistics, UK
louisa.nolan@ons.gov.uk

Ruben Henstra-Hill
Data Science Campus
Office for National Statistics, UK
ruben.henstra-hill@ons.gov.uk

Luke Shaw
Data Science Campus
Office for National Statistics, UK
luke.shaw@ons.gov.uk

Sharon Hook
Social Survey Division
Office for National Statistics, UK
sharon.hook@ons.gov.uk

ABSTRACT

In this paper, we present a novel application where Human-Computer Interaction (HCI) meets Artificial Intelligence (AI) and discuss obstacles that need to be resolved on the long journey from research to production. Unlike academia and industries that have been at the forefront of automation for decades, government is a new player in the field, though an important one. We build systems that are used on a large scale, we collect data to inform policymakers. Using the example of the Household Budget Survey, we demonstrate how government agencies can apply Human-in-the-Loop AI to automate the production of official statistics. 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. One major challenge is the human factor. How will the users, who are accustomed to manual tasks, react to the complexity of AI? How should we design the interface to give them a good user experience? How do we measure success? Indeed, one key step towards production is to secure funding, which requires presenting potential success in a way that the stakeholder can understand. Stressing the importance of formulating problems from a practical business viewpoint, we hope to bridge the communication gap and help the research community reach out to more potential users and help solve more novel real-world problems.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction**; • **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

Human-in-the-Loop, Human-Machine Interaction

ACM Reference Format:
Lanthao Benedikt, Chaitanya Joshi, Louisa Nolan, Ruben Henstra-Hill, Luke Shaw, and Sharon Hook. 2020. Human-in-the-Loop AI in Government: A Case Study. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3377325.3377489

IUI '20, March 17–20, 2020, Cagliari, Italy

ACM ISBN 978-1-4503-7118-6/20/03.
https://doi.org/10.1145/3377325.3377489

1 INTRODUCTION

Although it is a relatively new concept, Human-in-the-Loop (HuIL) has been in use in the real-world for some time. Examples include flight simulators[45], helpdesk chatbots[7], photo tagging[36], self-driving cars[16], to name only a few. Not every automation problem requires HuIL, however, applications that require high accuracy certainly do. The production of official statistics is another example where processing time and accuracy are paramount. Official statistics are used by governments to understand economy and society. Accuracy helps paint a correct picture of society, faster results allow faster political decisions.

To demonstrate how government agencies can apply HuIL to replace their legacy production systems, we propose an automation pipeline for the Household Budget Survey (HBS), a survey that exists in almost every country. Our aim is to develop methods that could be generalised and reused by any government agency. The paper is organised as follows.

- `Background`: we describe the typical HBS data collection process and highlight the need for modernisation. We discuss how the legacy system can be replaced with an automation pipeline and identify areas where improvement can be made. A set of metrics is proposed for measuring success.
- `Design and implementation`: we describe the implementation of each module in the pipeline. Highlighting situations where automation is not possible, we explore HuIL strategies to partition tasks between human and machine and design their interactions. The emphasis will be placed on decision points where machine hand-overs tasks to human and vice-versa.
- `User interface`: modernising a business process is more than changing the software, the human factor is a key challenge. If the users dislike the new system, there will be negative impact on productivity and team morale. How can we give them a good user experience, thus gaining their trust? HCI researches often deal with designing public-facing applications for casual users, do they apply to production systems where users are highly trained for the tasks?
- `Conclusion and Future work`: we layout the next steps for going into production and highlight other challenges beyond technological considerations.

Table 1: Example of products and their Classification of Individual Consumption by Purpose (COICOP).

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 2: Comparison between office scanning and mobile app scanning of receipts

Success criteria	Scenario 1 (Office scanner)	Scenario 2 (Mobile app)
Efficiency savings	medium	high
Technological difficulty	low	high
Investment costs	low	high
Data quality	high	low/medium
Respondent burden	low	high
Processing speed	high	low/medium
Human-Machine interactions	medium	high

2 BACKGROUND

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* or the Canadian *Survey of Household Spending*. It collects data on household incomes and spending patterns, which provides crucial information for the calculations 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.

For a household that is selected for the survey, every member above a certain age keeps a diary of expenditure over a 2-week period. Information such as *descriptions of purchases*, *prices*, *shop names*, *purchase dates* and *modes of payment* need to be recorded. To ease respondent burden, government agencies usually collect shopping receipts so the participants do not need to write down all the details. Then, back in the agencies, a team of coders manually type the information from the receipts into the system and manually classify each purchased item to a 5-digit United Nation-defined Classification of Individual Consumption by Purpose (COICOP) code[12]. An excerpt of the COICOP is shown in Table 1 for various milk products. Typically, it takes about 4-5 hours to manually process one single diary and it requires a large team of coders to complete the tasks. One can clearly see the potential for improvements by replacing manual operations with automation, but this raises an important question: *How do we measure success*?

Within the Machine Learning (ML) community, we typically assess and compare model performances by measuring quantities such as *accuracy*, *precision*, *F-score*, *recall*, and plot *Receiver Operation Characteristics* curves, measuring *Area Under the Curve*[8, 15, 23, 27]. However, from a business perspective, such quantities are not very 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*[4, 18], and in the context of official statistics, *respondent burden*[25]. Often, it is about finding a trade-off between these variables. Thus, in this paper, we will use both sets of quality measures, and we will explain how model performance scores can be translated into business performances in the present context.

3 DESIGN AND IMPLEMENTATION

At the beginning of the project, our team of data scientists visited the coding team to understand the nature of their work. Based on our observations, we design a high-level automation pipeline as shown in Figure 1.

(1) `Scanning`: shopping receipts are scanned using either an office flatbed scanner or a mobile phone app.
(2) `Image processing`: receipts are segmented from the scans and image enhancement applied.
(3) `Optical Character Recognition (OCR)`: text is extracted from the receipts and parsed so that useful information is automatically retrieved.
(4) `Natural Language Processing`: OCR is usually not perfect. Misspelled words and abbreviations must be resolved if they negatively affect classification.
(5) `Machine Learning (ML) classification`: we test various feature extraction methods and ML models, combined in a voting architecture to improve performance. Where automation fails, we propose HuIL active learning as a solution.

Let us examine in further detail each module in the pipeline and understand the need for human machine interaction. The focus of this paper is to examine the use of AI in a real-world environment and present the problem from a practical business viewpoint. To this end, we follow guidance from well-established literature on best practice in the field[13, 19, 38].

3.1 Receipt scanning

For data capture, at least two scenarios are possible:

- `Scenario 1`: we collect paper receipts from respondents and scan them in our office with a flatbed scanner.
- `Scenario 2`: we build a mobile phone app that allows respondents to capture images of receipts and upload them directly to the government agencies' Cloud storage.

We assessed the pros and cons of both scenarios, the results are summarised in Table 2. Scenario 1 requires staff to collect and scan receipts, thus *efficiency savings* is estimated to be medium. In scenario 2, we no longer need to collect and scan paper receipts so efficiency savings is higher. In scenario 1, the only *investment costs* is to acquire a flatbed scanner. In scenario 2, there is an important investment in developing the mobile phone app, user testing, setting up a Cloud platform for secured data exchange. In scenario 1, office staff can be trained to consistently produce high quality scans. In scenario 2, since not everyone is technology-savvy, we need to implement basic checks in the app to ensure the photos are of good

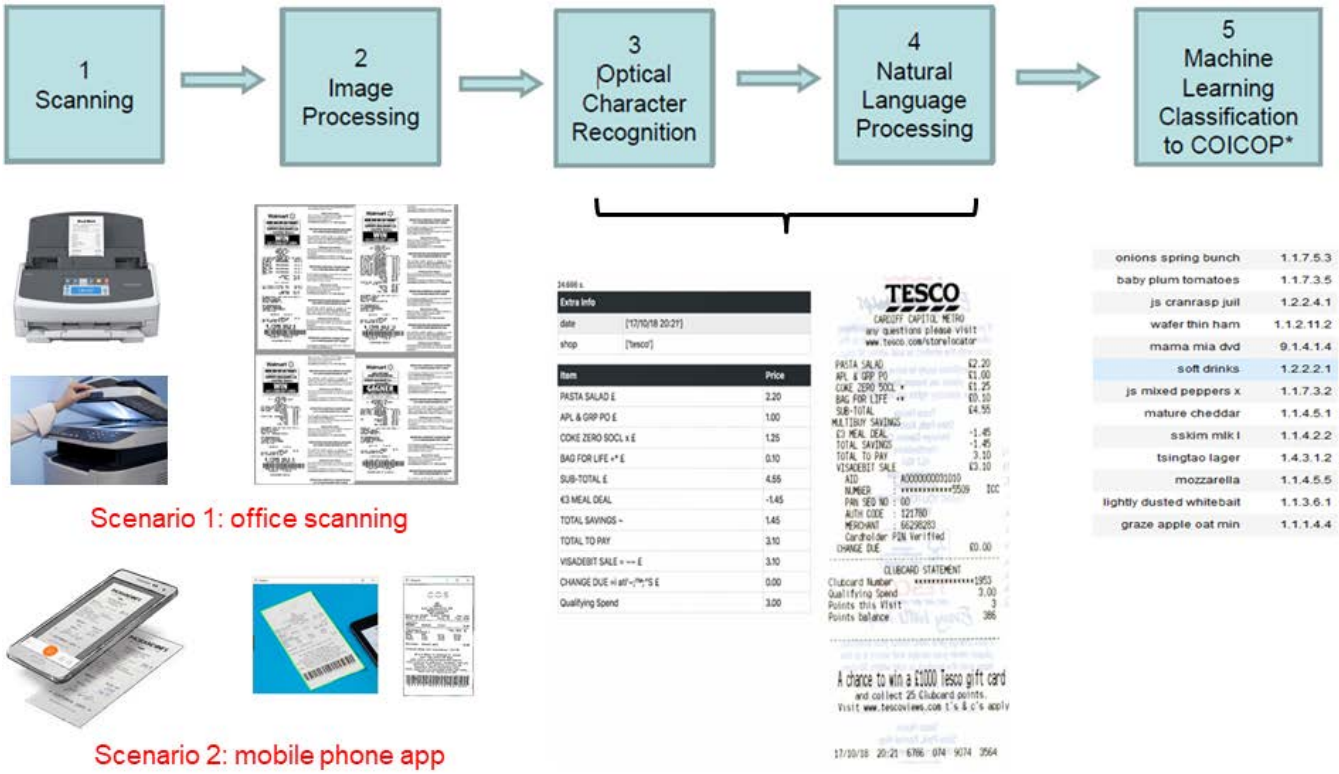

Figure 1: Automation pipeline to replace the legacy data collection system for the Household Budget Survey

quality e.g. lighting, contrast, image resolution, position of receipts, etc. The app feeds back to users, requesting them to take better photos if needed. Because of such human-machine interactions, scenario 2 may increase *respondent burden*. As most HBS surveys already suffer critical falling response rates, further research and tests need to be carried out to mitigate such risk.

Because of the aforementioned considerations, currently, our preferred option is the lower-risk scenario 1. The key point is, the better option from a research viewpoint is not always the preferred option from a business viewpoint. Indeed, scenario 2 offers an amazing research opportunity to explore modern technologies, cloud computing, data security, designing mobile phone app. However, the more sophisticated the technology, the more difficult it is to implement and maintain the new system. Besides, whilst the research community always seeks to push boundaries, businesses are usually a little more cautious regarding novelty because it represents a higher risk due to the *unknown* factor. There is also the cost to upskill staff to novel technologies. For the business to accept such risk, advantages need to be demonstrated somewhere and measured in terms of business performances, which are *data quality*, *efficiency savings*, *processing speed* in this case study.

Nowadays, only a few leaders in the field succeed in productionising state-of-the-art technologies. Many great researches do not go beyond the proof-of-concept stage because novelty is not the main selling point. Governments aim to build production systems. We do not seek to push boundaries but to reuse and adapt current inventions, hence the need to bridge the communication gap.

3.2 Image processing

Having opted for office scanning, many parameters can be controlled. For example, we can decide for all receipts to be consistently scanned at a chosen resolution, in a chosen image format and position, as shown in Figure 1, we decided for each receipt to be scanned such that its front occupies the left hand side of the scan, the back of the receipt is on the right hand side. This makes the automated cropping operation very easy with minimal human intervention, thus saving time.

We carried out tests to identify the best image resolution. Typically, for good OCR, the height of the x letter *x-height* needs to be about 20 pixels[5, 37]. As most text on receipts are of font size 10pt, a resolution of 300dpi yields *x-height* $\approx$ 20 pixels. For this reason, best practices in OCR usually advise for receipts to be scanned at 300dpi[1]. Our tests also show that higher resolutions e.g. 1200dpi yield very low accuracy because it captured more noise and it also takes much longer to process.

The image processing module is implemented in Python. The receipts are cropped from the original scan. The quality of the image is enhanced by applying thresholding algorithms, morphological transformations, removing noise, smoothing edges, etc. Processing time is typically between $\approx$6 and 10 seconds for a receipt. Since we control many parameters during the scanning process, image

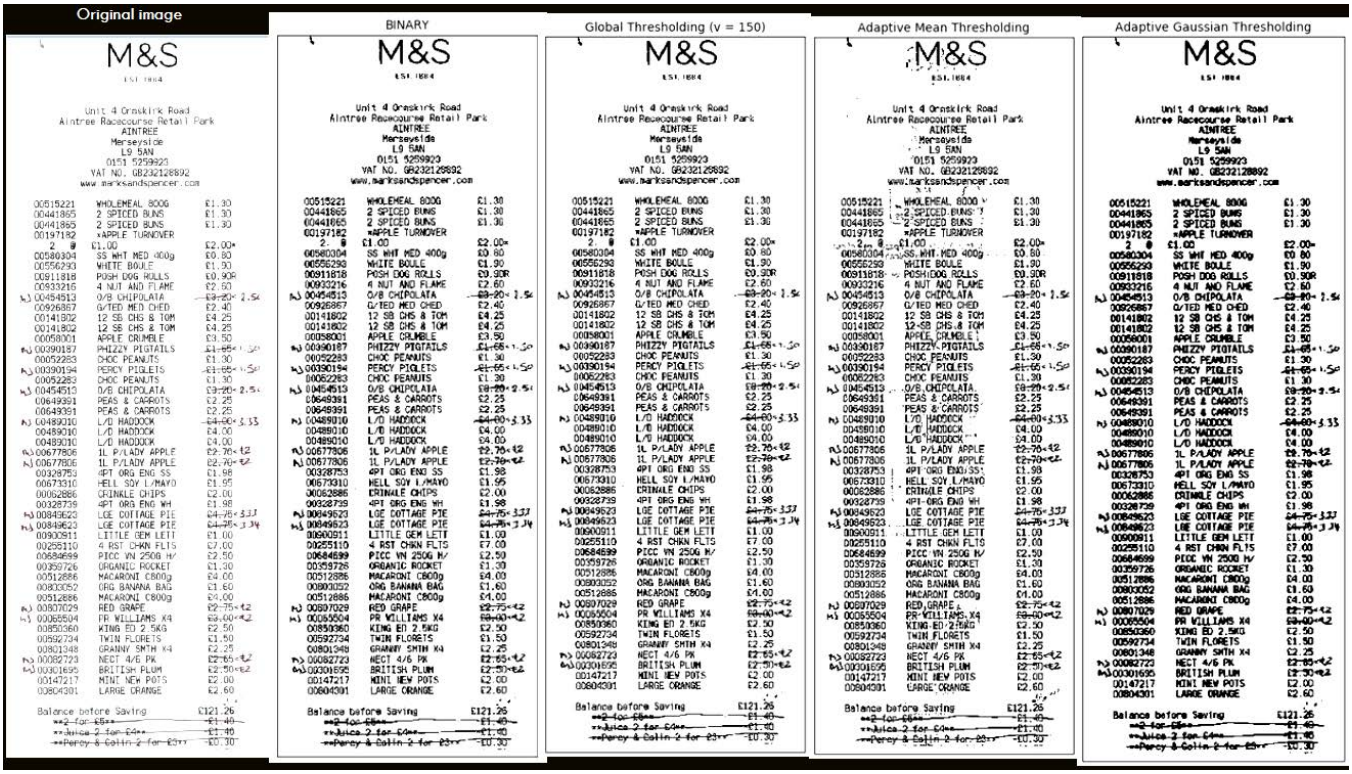

Figure 2: Example of receipt before and after applying image enhancement. From left to right: Original image, Binarisation, Global thresholding (threshold=150), Adaptive mean thresholding, Adaptive Gaussian thresholding.

enhancement is usually not necessary except in particular cases where the receipts are of low quality (faded, creased).

3.3 Optical Character Recognition (OCR)

OCR is the process of locating text in an image and extracting it into an editable format. Early OCR algorithms relied on *pattern recognition* and rule-based *feature detection* to recognise characters[41]. More recent algorithms introduce state-of-the-art data science techniques, for example Recurrent Neural Networks such as Long Short-Term Memory (LSTM)[21, 42]. The LSTM models are trained on images of characters in various Fonts, types (normal, bold, italic, etc.) and image quality. Such models are language dependent because of special characters (e.g. the German umlaut ä, ö, ü) and also because the LSTM model learns sequences of characters that are different from one language to another.

In this paper, we use the open-source Tesseract software package that was originally developed by Google. The latest version 4.0 released in 2018 implements both the legacy engines and the new LSTM recogniser[17, 43]. The data used for training the LSTM networks included a significant amount of degraded images produced by cameras. If LSTM fails on a particular character sequence, it can fall-back to its legacy recogniser to make the decision. For a given document, Tesseract outputs blocks of text together with their bounding boxes' coordinates.

For documents that are well formatted - such as survey forms that use standardised templates - knowing the position of the text is sufficient to extract the desired information. Thus, government agencies have been using OCR to process survey forms for decades. However, for the present application dealing with a variety of receipts, knowing the text position is not sufficient to locate the useful information. Indeed, although the shop name is typically at the top of the receipt, information such as dates, items, prices can be formatted differently from shop to shop. Some receipts have barcodes, some do not, etc. Figure 3 shows an example of a Canadian supermarket receipt and Figure 4 shows how the desired information has been extracted and formatted into a Python dataframe. OCR took about 4 seconds and data parsing about 2 to 3 seconds on a MacBook Pro. A human coder usually takes about 5 to 10 minutes to manually type the same amount of information, barcodes being the most cumbersome to transcribe.

To parse the data and extract useful information, we use both fuzzy matching[39] and regular expressions[3]. For example, to recognise the shop name, we apply fuzzy matching to compare the OCR'ed text against a dictionary of known shops. The operation does not take long since shop names usually appear at the top of the receipt. There are particular cases where the top of the receipt is cut off. In such cases, the program will recognise the shop name in other places such as in the shop's website address or in own-brand products' descriptions. Fuzzy matching accounts for misspelling. The same logic is applied to recognise the payment mode, the headers and the keywords that are used to locate the part of the receipts where the purchased items are.

Figure 3: Example of receipt. It is common practice at Costco in Canada that shopping carts are checked at the exit and the receipt crossed off.

Prices, barcodes and dates are extracted using regular expressions to recognise patterns. For example, the pattern

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

can be used to extract any dates in the form of $yyyy/mm/dd$, $mm/dd/yyyy$ or $yy.dd.mm$. To measure OCR accuracy, we calculate the Levenshtein distance between the descriptions keyed-in by human (the gold standard) and the OCR'ed output. Dates are split into *day*, *month*, *year* and compared against human's recordings. Since text parsing can mistakenly remove relevant lines or omit to remove irrelevant lines, we also count the number of extra and missing lines per receipt as a means to measure OCR correctness.

As mentioned earlier, a dictionary is used to recognise known shops. No dictionary can be exhaustive as new shops open all the time. In such situations, a system that is 100% automated would fail because it cannot find the new shop in the dictionary. In a HuIL system, machine would flag up the shop as *'Unknown'* to alert human who would then update the dictionary. After this

Figure 4: Useful information is automatically retrieved and formatted into a dataframe. The operation takes 4 seconds.

intervention, human would handover to machine to carry on with the automated process. OCR takes about 4 seconds per receipt on average. Data parsing typically takes between 2 to 10 seconds. Even with human intervention to correct OCR mistakes, one can predict potential gains in *efficiency savings* and *processing speed* compared to having humans type out the same amount of information. HuIL, on the other hand, ensures that *data quality* is maintained to the desired level.

3.4 Natural Language Processing (NLP)

There exist situations where even the best OCR engines fail. For example, it is common practice at Costco that shopping carts get checked and receipts crossed off, as shown in Figure 3. The pen stroke makes some characters difficult to recognise, resulting in misspelled words. To solve this problem, NLP literature offers a wealth of Spell Checking and Error Correcting algorithms[11, 22, 24]. Most well-established OCR engines such as Tesseract support common NLP capabilities, for example user-specified *blacklisting*, *whitelisting* of characters, default dictionaries (supporting up to 116 languages) and user-defined dictionaries[42].

In this application, as receipt descriptions contain many shop-specific abbreviations, we replace Tesseract default English dictionary with a user-defined dictionary built from historical records of purchased items' descriptions. This is still under development and our test results have not been convincing so far. For example, the word *'chicken'* is abbreviated on the receipt as *'ckn'* and mispelled as *'c4n'*. Applying automated NLP correction, *'c4n'* is auto-corrected to *'can'*. Although both words *'ckn'* and *"can"* can be found in our bespoke dictionary, *'can'* is a more frequently used word so it is preferably chosen by the algorithm. Using n-gram word embedding is another option we consider, the logic being that the surrounding words can provide more context to the auto-correction[14]. However, receipts are often very succinct, offering little context.

Our test results show that currently our NLP auto-correction cannot achieve the desired *data quality* level, therefore we take a HuIL approach, allowing human to step in and correct misspellings

when needed. We will investigate further NLP methods to improve auto-correction in the future, thus making more efficiency savings. Nevertheless, some level of human intervention will always be necessary, if only for quality assurance. Human coders confirm correct spellings so these can proceed to the next stage of the pipeline.

3.5 Machine Learning classification

The idea is to automatically categorise purchased items into their respective COICOP codes, which can be achieved with supervised Machine Learning (ML)[26]. Producing a labelled dataset to train the models is one of the most resource intensive tasks for humans. Many applications use crowd-sourcing to generate such labelled dataset. However, for this specific problem, labelling data requires domain knowledge. In this paper, we use a dataset of 389,432 entries that have been labelled by experienced HBS coders. The data covers the same period in 2017, 2018 and 2019 to account for seasonality effects. Such dataset is sufficiently large for developing and validating our methods, however, we need to collect larger volume of data going forward if we are to estimate meaningful business performance.

This is a multi-class classification problem where each item must be assigned to one and only one class. The COICOP coding frame comprises about 375 categories of food products[12]. This being a probabilistic process, the model categorises an item to its class with a confidence score that can be used to accept or reject an assignment. Judging by the nature of the data - e.g. *'bread'* and *'milk'* are likely more often bought than, say *'Whisky'* - there is a risk of *class imbalance*, which is magnified in multi-class problems[2]. 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. Therefore, in this study, we use weighted metrics (for accuracy, precision, F-score, recall) to capture a more truthful story[31] and we also define a bespoke performance metric that is more pertinent from a business perspective as follows.

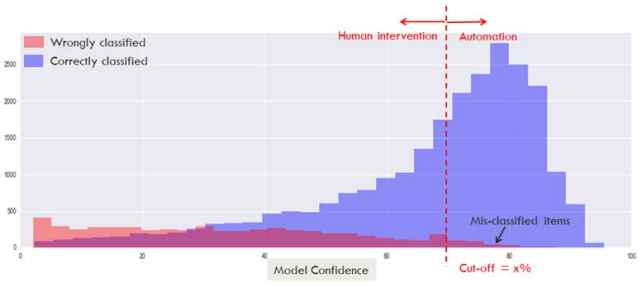

Figure 5: Linking model performance metrics to success measures from a business perspective.

Defining a bespoke performance metric: assuming we split the dataset into a training set (80%) and a test set (20%). We use stratified sampling by class to account for class imbalance[32]. We train the ML models with the training set and use these to classify items in the test set. Since we have the ground truth, we can identify items that are correctly classified and wrongly classified. We plot their histograms by model confidence as shown in Figure 5. For a given model confidence cut-off, say $threshold = x\%$:

- `Accepted`: items that fall on the right hand side of the cut-off are those for which the model confidence $\geq threshold$. They are accepted. It happens that the model sometimes classifies an item to the wrong category with a high confidence.
- `Rejected`: items that fall on the left of cut-off are those for which the model confidence $< threshold$. They are rejected. It happens that the model sometimes classifies items correctly but with a low confidence.

Let us consider a scenario where the business requires a *data quality* level of 2% error rate, which means the number of misclassified items that are accepted must be $\leq 2\%$. We can calculate the value of *threshold* to achieve the desired 2% error rate. Once the cut-off value known, we can calculate the percentage of items that fall on the right of cut-off (percentage of automation). Items that fall on the left of cut-off are either sent to human coders, or be looked up in a dictionary as we will discuss later.

We test three popular feature extraction methods implemented in the Scikit-learn Python package[6]: Countvectorizer (CV), Term Frequency–Inverse Document Frequency (TF-IDF) at word level (TFIDF-w) and at character level (TFIDF-c) and popular supervised ML models e.g. Naive Bayes[10], Logistic Regression (LR)[9], Random Forest (RF)[20], Support Vector Machines[44] and XGBoost[29], with a n-fold cross-validation. We also tested state-of-the-art ngram word embedding methods such as Embeddings from Language Models (ELMo)[33] and FastText[34]. FastText was tested both as a supervised model and a Transfer Learning pre-trained model[40]. Of all tested models, well-established such as LR and RF perform the best. We found that FastText embeddings with Logistic Regression yields an accuracy close to only 68%. ELMo word-embedding shows promising initial results but is computationally more expensive. Generally, state-of-the-art models seem more data-hungry, so while waiting to collect more data, we focus on traditional models that seem to perform better on smaller datasets[28]. To balance out the individual weakness of each model, we implement an Ensemble Learning architecture which applies a *soft voting* concept to select the most likely prediction based on the average predicted probability[35]. Simulation results are shown in Table 3 for the best performing models. Example model characteristics are shown in Figure 6.

Business performance metrics for a range of error thresholds are shown in Table 4 for various ML classifiers. As expected, percentage of automation increases monotonically with the error threshold. A maximum automation rate of 71% can be achieved for an error rate of 5%, using either a Logistic Regression model (LR) in conjunction with a Countvectoriser feature extraction (CV), or a Random Forest model (RF) with a Countvectoriser feature extraction (CV).

One significant problem we often have to deal with is that receipt descriptions are usually very succinct. In some cases, there is so little information that it is impossible for any model to make a correct prediction. For example, a receipt description can simply state *'fresh milk'*, so we don't know whether it is *'whole milk'*, *'skimmed milk'* or *'semi-skimmed milk'* that must be classified to different categories as seen in Table 1. In the solution that we develop, since the words in *'fresh milk'* are very generic, they could belong to anything

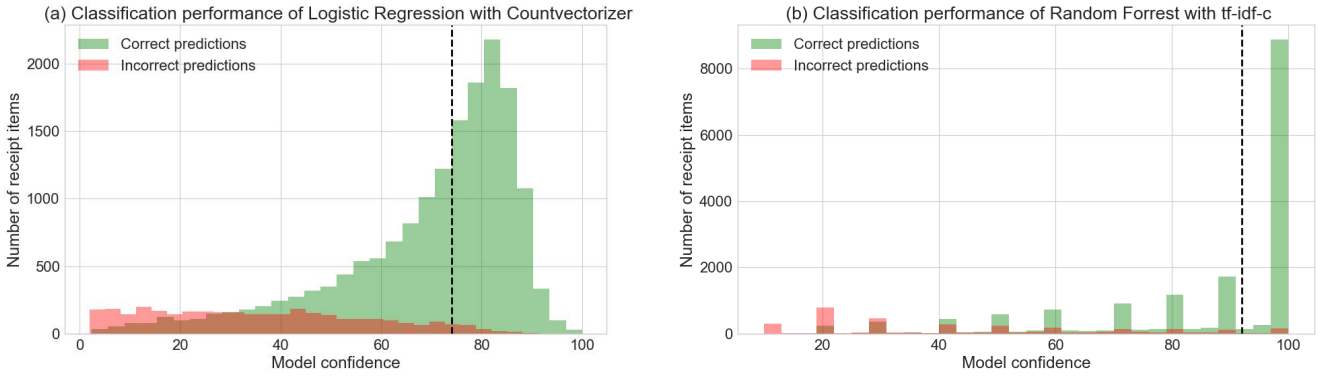

Figure 6: Percentage of automation for a chosen error rate. In this example: the vertical line shows the minimum confidence cut-off (74% for Logistic Regression and 92% for Random Forest) required to achieve 2% error rate.

Table 3: Performance metrics of various Machine Learning classifiers. We use weighted scores to account for potential class imbalance[31].

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	0.79	0.79
LR/TFIDF-c	0.80	0.80	0.81	0.80
RF/CV	0.82	0.82	0.83	0.82
RF/TFIDF-w	0.80	0.80	0.81	0.80
RF/TFIDF-c	0.82	0.82	0.82	0.82
FastText	0.85	0.85	0.85	0.85
Soft Voting	0.86	0.86	0.86	0.86

Table 4: 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 follow different strategies.

Automation rate as a function of error rate (er)				
ML model	er=2%	er=3%	er=4%	er=5%
LR/CV	45%	58%	66%	71%
LR/TFIDF-w	41%	48%	56%	62%
LR/TFIDF-c	43%	53%	61%	66%
RF/CV	0%	53%	65%	71%
RF/TFIDF-w	0%	52%	62%	68%
RF/TFIDF-c	47%	57%	65%	67%
FastText	0%	62%	72%	78%

such as *'fresh meat'*, *'fresh bread'*, *'milk chocolate'*, etc., we expect their TF-IDF scores to be very low and subsequently, the model should make a prediction with low confidence. If the *threshold* is correctly set, the item's confidence score is < *threshold*, therefore it will be flagged up and sent to human to make further decision, as shown in Figure 7. The solution could be then for a coder to contact the survey respondents to verify what kind of milk they bought, which certainly provides high *data quality*, at the risk of increasing *respondent burden*, *production costs* and *processing time*. To circumvent such problem, government agencies usually include a *'Usual Purchases'* page in the questionnaire, asking respondents what kind of *'milk'* they preferably buy, among other common products. Thus, *'fresh milk'* can be imputed with this information. This is an acceptable trade-off between *data quality* and *respondent burden*.

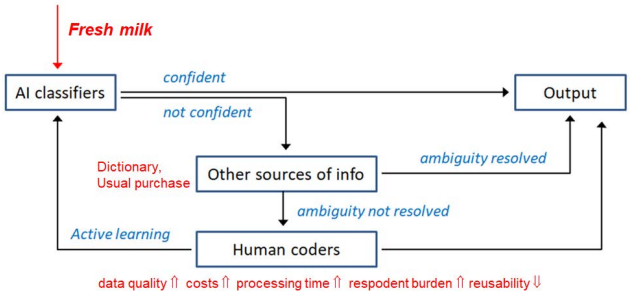

Figure 7: Active Learning: the model prediction confidence is compared against a user-defined threshold. If $confidence \geq threshold$: prediction accepted. If $confidence < threshold$: the item is looked-up in a dictionary or sent to a coder. In the latter case, the model is updated with the new information.

There are also other instances where machine will not perform well, for example, rare products or new products that the models have not seen before. In such case, the model will make a prediction with a very low confidence score, which causes the items to be flagged up. A coder can then manually classify the item, the result will be added to a more up-to-date labelled dataset that will be used to retrain the models overnight to make them more performant. This is the principle of *'Active Learning'* that forms the core of HuIL. Where a system that is 100% automated would fail because it cannot cope with changes, HuIL allows the model to learn from human and stay up-to-date.

4 USER INTERFACE (UI)

Whilst HCI literature is full of great UI examples, most deal with designing public facing applications, aimed for casual users. On the contrary, when one designs interface for a production system as in this case study, survey coders are highly trained and experienced users who repeat the same tasks day in day out over years. The constraints are different, thus common User Experience (UX) best practice may not apply. Another major difference is concerned with *acceptance*. A user visiting a new movie rating website, for instance, is likely enthusiastic and one tries not to disappoint him. On the other hand, employees are likey nervous about AI, worrying about job loss, so their initial reaction is more critical. One needs to win their trust.

To design a system that is fit-for-purpose, we believe the first step is to know the users and to adopt a human-centered design approach. To this end, we visited the coding team to observe them in their daily tasks to understand the current business process. We interviewed them on what they liked or disliked and identified areas for improvement. Our findings are summarised as below.

- The typical coders are not tech-savvy. They are very accustomed to the legacy manual data entry processes and may feel uncomfortable with a new system involving AI. Users are humans with needs such as comfort and low cognitive loads, effort must be made to maintain a sense of familiarity.
- They are experienced and very quick at judging the usability and likeability of the software. The UI creates the first impression so it is extremely important that it hides the sophisticated and wearisome AI machinery.
- As we sketched out together new features, they often attempted to reproduce features in the legacy system and always benchmarked novel ideas against the current system.
- Their tasks are very repetitive. Effort must be made to prevent repetitive strain injuries, for instance, by displaying information in proximity to reduce overall travel time both for eye and mouse movements. We can also make controls accessible by both keyboard and mouse, so users can switch.
- Data collection is a continuous process There is a risk that changes may cause disruption of services so it is better to introduce them gradually rather than one big step change.
- There is understandably a general feeling of concern with the introduction of AI. If not well managed, this can affect the team's morale and productivity. Human-in-the-loop may be a good approach to convey the message that the coders play an active role in the new workflow. The UI should facilitate good human machine interaction and avoid confusion.

With this in mind, we mocked up a first UI as shown in Figure 8. The design decisions were taken in concert with the coders and the software developer who built the legacy system, we reproduced as much as possible the look and feel of the legacy interface, using similar screen colours and layout to create a sense of familiarity. The design is compliant to Don Norman's 6 cognitive design principles[30]: *Visibility*, *Feedback*, *Affordance*, *Mapping*, *Constraint* and *Consistency*. The user story is as follows.

- `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 and click OK.
- `Step 2`: a receipt is then shown on screen. The image is on the left, the information extracted from the OCR output is on the right, as shown in Figure 8. The fields are pre-filled with text extracted from the receipt using OCR. 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 data parsing went wrong, 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.
- `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 smiley appears to confirm success. the next line becomes active and highlighted. The user repeats from Step 7.
- `Step 7b`: if $score < threshold$, a red smiley appears to indicate failure. The user assigns the correct COICOP code and confirms. The results will also be saved into a new training set used to retrain the models to make it more accurate (active learning). The next line becomes active and highlighted. The user repeats from Step 7.
- `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 message appears to inform the user that the task has been successfully completed.

The main purpose of this mockup is to gently familiarise coders to the new system. We can use it for training purpose and for collecting feedback and suggestions, which is a great way to bring the users on board by giving them a say in the way the new system is designed. By taking an active participation in implementing changes, the users will take ownership of the final product and trust the underlying technology. However, this UI design is not meant to be implemented as is. Indeed, whilst it looks acceptable for a short receipt, it may not be so for a long receipt with more than 30 items, for instance. If the screen looks too busy, it may cause confusion and frustration. Therefore, our intention is to redesign the UI in a second phase, displaying one item at a time on screen, as shown in Figure 9. This uncluttered design would improve visibility and is better from an ergonomic viewpoint. Information is displayed in proximity to reduce overall travel time both for eye and mouse

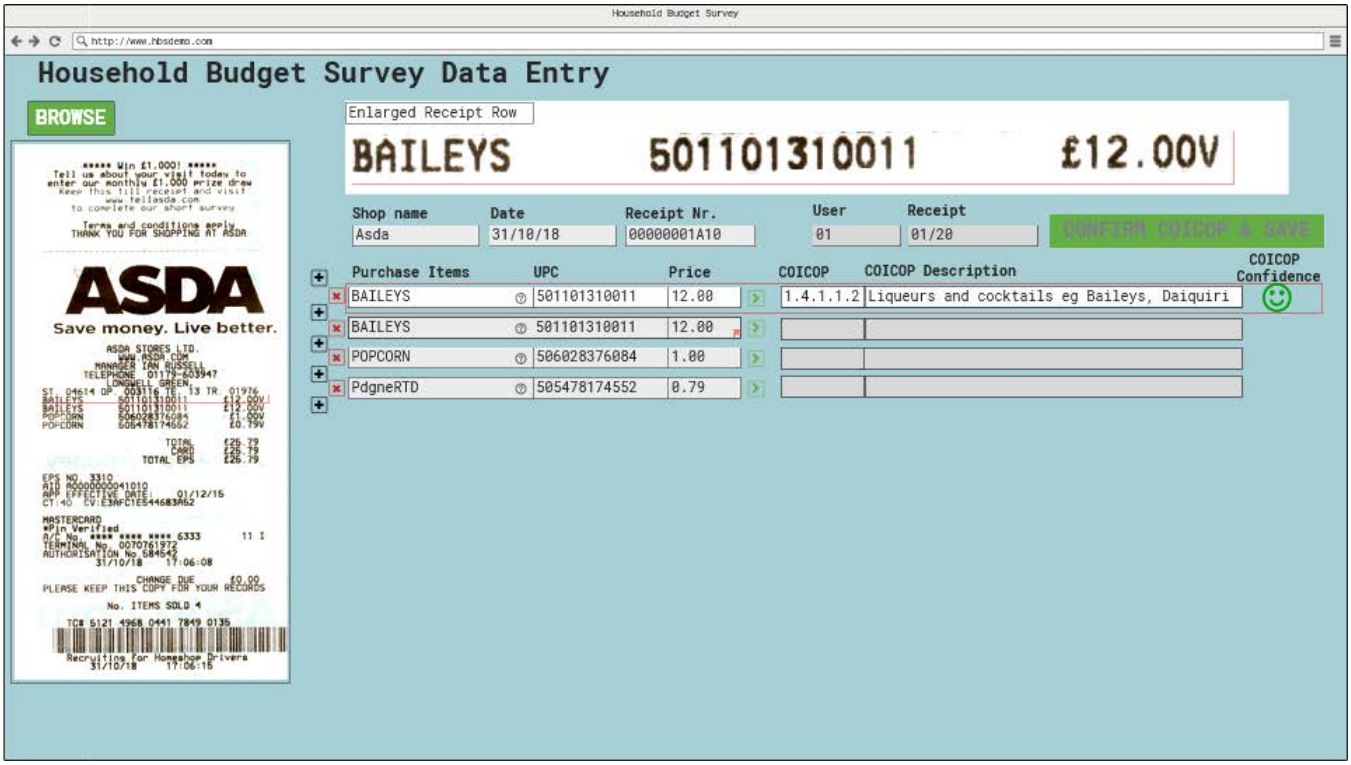

Figure 8: Mockup of the Household Budget Survey Data Entry User Interface. The receipt image is shown on the left, the extracted text on the right (e.g. shop name, date, items). All fields are editable. The currently edited line is highlighted. The snapshot shows the first item checked for OCR errors and classified the item to COICOP. The first item has successfully passed OCR checks and successfully classified.

movements. Controls are accessible by both keyboard and mouse, so users can switch, thus preventing repetitive strain injuries.

Figure 9: New layout to unclutter the screen, showing one item at a time. Information is displayed in proximity to reduce overall travel time both for eye and mouse movements. Controls are accessible by both keyboard and mouse, so users can switch, thus preventing repetitive strain injuries.

5 CONCLUSION AND FUTURE WORK

In this paper, we demonstrate how government agencies can replace their legacy manual processes with automation. We design and implement an end-to-end automation pipeline and discuss success measures. Avoiding reinventing the wheel, we tap into public knowledge by using open-source software and only developing methods when no known solution can be found. We make our methods and codes publicly available so they can be reused by anyone.

Whilst we strongly advocate for automation as a means to make *efficiency savings* and speed up *processing time*, we also acknowledge that there are situations where we need human interventions to maintain *data quality*. Therefore, we propose a human-in-the-loop solution and adopt a human-centered approach to design a user interface that allows human-machine interaction to be closely intertwined at every step of the process.

As we continue to improve our methods, we are also planning the logistics to carry out further tests on larger volumes of data and in various countries. The main aim will be to collect evidence and conduct a cost benefit analysis to support the business case for going into production. This effort will be carried out in parallel to building capability so that in the future, the users will be able to operate and maintain the new system.

REFERENCES

[1] The UK National Archives. 2017. General hints and tips for digitisation for business use. *Guidance and Best Practice* (Sept. 2017). https://www.nationalarchives.gov.uk/documents/information-management/hints-tips-digitisation-for-business-use.pdf

[2] Gustavo Batista, Ronaldo Cristiano Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter* 6, 1 (June 2004), 20–29. https://doi.org/10.1145/1007730.1007735

[3] Philip Bille and Martin Farach Colton. 2008. Fast and Compact Regular Expression Matching. *Theoretical Computer Science* 409, 3 (Dec. 2008). http://arxiv.org/abs/cs/0509069

[4] George A. Boyne. 1992. Local Government Structure and Performance: Lessons from America? *Public Administration*. 70, 3 (Sept. 1992), 333–357. https://doi.org/10.1111/j.1467-9299.1992.tb00942.x

[5] Matteo Brisinello, Ratko Grbic, Matija Pul, and Tihomir Andelic. 2017. Improving optical character recognition performance for low quality images. In *Proceedings of the 2017 International Symposium ELMAR*. IEEE. https://doi.org/10.23919/ELMAR.2017.8124460

[6] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gael Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (June 2013), 108–122. https://scikit-learn.org/stable/_downloads/scikit-learn-docs.pdf

[7] Toxtli Carlos, Monroy-Hernandez Andres, and Cranshaw Justin. 2018. Understanding Chatbot-mediated Task Management. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (May 2018). https://doi.org/10.1145/3173574.3173632

[8] Gavin Cawley and Nicola Talbot; 11(Jul). 2010. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research* (July 2010), 2079–2107. http://www.jmlr.org/papers/v11/cawley10a.html

[9] J. S. Cramer. 2004. The origin of logistic regression. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences* 4, 35 (Jan. 2004). https://doi.org/10.1016/j.shpsc.2004.09.003

[10] Wenyuan Dai, GuiRong XueQiang, and Yang Yong Yu. 2007. Trans-ferring naive Bayes classifiers for text classification. *Proc. of Assoc. for the Adv. of Art. Int. (AAAI 07)* (Aug. 2007). http://new.aaai.org/Papers/AAAI/2007/AAAI07-085.pdf

[11] de Amorim Renato Cordeiro and Zampieri Marcos. 2013. Effective Spell Checking Methods Using Clustering Algorithms. *Proceedings of the International Conference Recent Advances in Natural Language Processing* (Sept. 2013), 172–178.

[12] United Nations Statistics Division. 2018. COICOP Revision. https://unstats.un.org/unsd/class/revisions/coicop_revision.asp

[13] Cranor Lorrie Faith. [n.d.]. A Framework for Reasoning About the Human in the Loop. *Proceedings of the 1st Conference on Usability, Psychology, and Security* ([n. d.]).

[14] Pieter Fivez, Simon Suster, and Walter Daelemans. 2017. Unsupervised Context-Sensitive Spelling Correction of English and Dutch Clinical Free-Text with Word and Character N-Gram Embeddings. *BioNLP* abs/1710.07045, 3 (Aug. 2017). http://arxiv.org/abs/1710.07045

[15] Peter Flach. 2019. Performance Evaluation in Machine Learning: The Good, the Bad, the Ugly, and the Way Forward. *AAAI Press 2019* (Jan. 2019), 9808–9814. https://doi.org/10.1609/aaai.v33i01.33019808

[16] Miriam GilEmail, Vicente Pelechano, Joan Fons, and Manoli Albert. 2016. Designing the Human in the Loop of Self-Adaptive Systems. *International Conference on Ubiquitous Computing and Ambient Intelligence* (Nov. 2016), 437–449. https://doi.org/10.1007/978-3-319-48746-5_45

[17] Github. 2019. Tesseract Releases. https://github.com/tesseract-ocr/tesseract/releases

[18] Keith Hartley. 2009. Value for money in defence: Strategic choices and efficiency savings. *The Chartered Institute of Public Finance and Accountancy. Public Money* 5, 4 (Jan. 2009), 33–38. https://doi.org/10.1080/09540968609387356

[19] Brink Henrik, Richards Joseph, and Fetherolf Mark. 2016. Real-World Machine Learning. *Manning Publications Co. 1sr Ed.* (Jan. 2016). https://doi.org/1617291927, 9781617291920

[20] Tin Kam Ho. 1998. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8, 20 (Aug. 1998). https://doi.org/10.1109/34.709601

[21] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (Jan. 1997). https://doi.org/10.1162/neco.1997.9.8.1735

[22] V. J. Hodge and J. Austin. 2003. A comparison of standard spell checking algorithms and a novel binary neural approach. *IEEE Transactions on Knowledge and Data Engineering* 15, 5 (Sept. 2003), 1073–1081. https://doi.org/10.1109/TKDE.2003.123226

[23] Kononenko Igor and Bratko Ivan. 1991. Information-Based Evaluation Criterion for Classifier's Performance. *Machine Learning* 6, 1 (Jan. 1991). https://doi.org/10.1023/A:1022642017308

[24] Michael P. Jones and James H. Martin. 1997. Contextual Spelling Correction Using Latent Semantic Analysis. *Fifth Conference on Applied Natural Language Processing* (Jan. 1997), 166–173. https://doi.org/10.3115/974557.974582

[25] Sharp Laure. 1981. Respondent Burden: A First Measurement Effort. *Öffentliche Meinung und sozialer Wandel / Public Opinion and Social Change* (Jan. 1981), 194–208. https://doi.org/10.1007/978-3-322-87749-9_17

[26] Machine Learning. 2019. Supervised learning. https://en.wikipedia.org/wiki/Supervised_learning

[27] Sokolova Marina, Japkowicz Nathalie, and Szpakowicz Stan. 2006. Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. *AI 2006: Advances in Artificial Intelligence* (Jan. 2006), 1015–1021. https://doi.org/10.1007/11941439_114

[28] Banko Michele and Brill Eric. 2001. Mitigating the Paucity-of-data Problem: Exploring the Effect of Training Corpus Size on Classifier Performance for Natural Language Processing. *Proceedings of the First International Conference on Human Language Technology Research* (Jan. 2001), 1–5. https://doi.org/10.3115/1072133.1072204

[29] Didrik Nielsen. 2016. Tree Boosting With XGBoost - Why Does XGBoost Win Every Machine Learning Competition? *Master Thesis. Norwegian University of Science and Technology* (Jan. 2016). https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2433761/16128_FULLTEXT.pdf?sequence=1

[30] Donald A Norman. 1990. The design of everyday things. *New York: Doubleday Publishing Group* (Dec. 1990). http://www.nixdell.com/classes/HCI-and-Design-Spring-2017/The-Design-of-Everyday-Things-Revised-and-Expanded-Edition.pdf

[31] Scikit-Learn Python package. 20196. Model evaluation: quantifying the quality of predictions. https://scikit-learn.org/stable/modules/model_evaluation.html

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

[33] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (June 2018). https://arxiv.org/pdf/1802.05365.pdf

[34] Bojanowski Piotr, Grave Edouard, Joulin Armand, and Mikolov Tomas. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5, 20 (Aug. 2017).

[35] Polikar R. 2012. Ensemble learning, Springer. https://link.springer.com/chapter/10.1007/978-1-4419-9326-7_1

[36] Parasuraman R. and Sheridan Thomas. 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans* 30, 3 (Jan. 2000). http://hci.cs.uwaterloo.ca/faculty/elaw/cs889/reading/automation/sheridan.pdf

[37] Nitin Ramesh, AkshaSrivastava, and K.Deeba. 2018. Improving Optical Character Recognition Techniques. *International Journal of Engineering and Technology* 7, 2.24 (Jan. 2018), 361–364. http://dx.doi.org/10.14419/ijet.v7i2.24.12085

[38] Ling Rothrock and S. Narayanan. 2011. Human-in-the-Loop Simulations: Methods and Practice. *Springer* (Jan. 2011). https://doi.org/ISBN978-0-85729-883-6

[39] Ingersoll Grant S., Thomas S. Morton, and Andrew L. Farris. 2013. Taming Text: How to Find, Organize, and Manipulate It. *Manning Publications Co.* (Jan. 2013). https://doi.org/193398838X,9781933988382

[40] Aditya Siddhant, Anuj Goyal, and Angeliki Metallinou. 2019. Unsupervised Transfer Learning for Spoken Language Understanding in Intelligent Agents. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019). https://doi.org/10.1609/aaai.v33i01.33014959

[41] Ray Smith. 2007. An Overview of the Tesseract OCR Engine. In *Proceedings of the 9th International Conference on Document Analysis and Recognition*. IEEE. https://doi.org/10.1109/ICDAR.2007.4376991

[42] Ray Smith. 2016. Training LSTM on 100 languages and test results. https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/7Building%20a%20Multi-Lingual%20OCR%20Engine.pdf

[43] Rice Stephen, Frank Jenkins, and Thomas Nartker. 2013. The Fourth Annual Test of OCR Accuracy. http://www.expervision.com/wp-content/uploads/2012/12/1995.The_Fourth_Annual_Test_of_OCR_Accuracy.pdf

[44] Yichuan Tang. 2013. Deep Learning using Support Vector Machines. *CoRR* (Aug. 2013). http://arxiv.org/abs/1306.0239

[45] Li Wenchao, Sadigh Dorsa, Sastry S. Shankar, and Seshia Sanjit A. 2014. Synthesis for Human-in-the-Loop Control Systems. *Tools and Algorithms for the Construction and Analysis of Systems* 30, 3 (Jan. 2014), 470–484. https://doi.org/10.1007/978-3-642-54862-8_40