

Proceedings of Statistics Canada Symposium 2024: The Future of Official Statistics

Improving the Automated Capture of Survey of Household Spending Receipts using advanced Machine Learning Techniques

by Joanne Yoon and Oladayo Ogunnoiki

Release date: September 8, 2025



Statistics
Canada Statistique
Canada

Canada

Improving the Automated Capture of Survey of Household Spending Receipts using advanced Machine Learning Techniques

Joanne Yoon* and Oladayo Ogunnoiki*¹²

Abstract

The Survey of Household Spending (SHS) conducted by Statistics Canada collects paper diaries and shopping receipts as a source of household expenditure data. An auto-capturing algorithm was created for SHS 2023 to reduce statistical clerks' manual work of extracting important information from scanned receipts of common store brands. The algorithm used Tesseract optical character recognition (OCR) to extract text characters from images of receipts, and it identified store and product entities using regular expressions, also known as regex. The goal of this study was to enhance the current auto-capture algorithm by experimenting with more advanced OCR and machine learning methods. As a result, PaddleOCR, an open-source OCR toolkit, was selected as the new default OCR engine due to its overall performance in recognizing texts, especially digits, accurately across receipts of various qualities. Additionally, entity classifiers based on support vector machines were trained on historical SHS records and existing regex patterns. By using classifiers to categorize different elements present on receipts instead of relying solely on regex patterns, product and store recognition improved. It is expected that this new algorithm will be used for SHS 2025 to improve the auto-capture quality and reduce the manual burden associated with capturing receipt variables.

Key Words: OCR; Machine learning; Automation.

1. Introduction

Survey of Household Spending (SHS) is a Canadian survey that gathers information on the spending habits of Canadians. It looks at how much households spend on food, clothing, shelter, transportation, health care and other items. Respondents complete a questionnaire and a diary with their day-to-day expenses. When filling out the diary, respondents can choose to either write each purchased item in a paper diary or put their shopping receipts in the pocket of their diaries. This results in Statistics Canada collecting between 20,000 to 30,000 shopping receipts per survey cycle. The paper receipts are manually scanned in-house.

To reduce the manual work of extracting important information from scanned receipts, the SHS methodology team has developed an auto-capture algorithm for the receipts from popular stores with a good-quality image. For each receipt, details, such as the store name, transaction date and method of payment, and product variables, including product name, price, discount and deposit, are automatically captured. The algorithm was used in production for the SHS 2023 cycle. This algorithm extracts the text from the scanned receipts using Optical Character Recognition (OCR) and parses relevant information into a structured format. The resulting data is manually validated, and any errors are corrected.

To enhance the scope and performance of the SHS receipt auto-capture algorithm, several improvements were implemented on top of the existing algorithm as described in the article. Specifically, the project aimed to enhance the quality of the extracted information by exploring more OCR engines, incorporating an entity machine classifier, and

*These two authors contributed equally to this work.

¹Joanne Yoon, Statistics Canada, 170 Tunney's Pasture Driveway, Ottawa, Ontario, Canada, K1A 0T6 (Joanne.Yoon@statcan.gc.ca); Oladayo Ogunnoiki, Statistics Canada, 170 Tunney's Pasture Driveway, Ottawa, Ontario, Canada, K1A 0T6 (Oladayo.Ogunnoiki@statcan.gc.ca)

²The views expressed in research papers are those of the authors and do not necessarily reflect the opinions of Statistics Canada or of the federal government.

refining the parsing and data cleaning steps. The evaluations indicate that the new OCR engine, spell corrector and enhanced entity recognition method improved the accuracy of automatically captured variables.

2. Explored methods to improve receipt auto-capture

The auto-capturing pipeline consisted of the following steps as visualized in Figure 2-1:

Optical Character Recognition (OCR): Texts in scanned receipts are converted to machine-readable texts using OCR. It returns both the text and its confidence score on the digitized text so that the confidence score can be used to assess the quality of the scanned receipt. The newly explored OCR engines are elaborated in section 2.1.

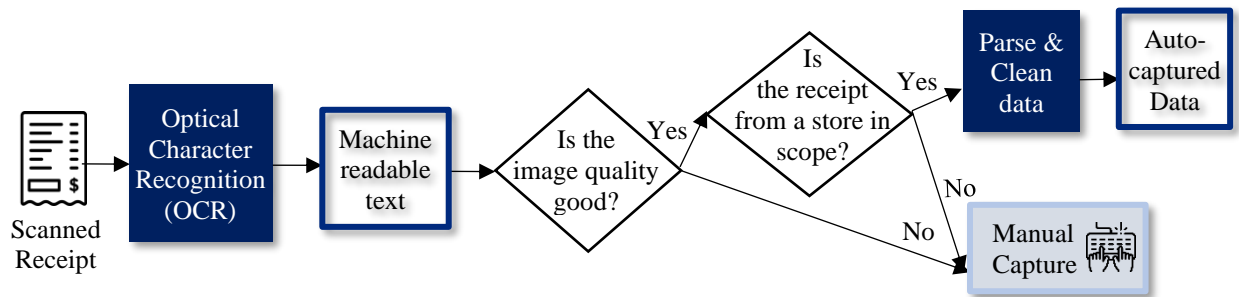
Image Quality Assessment: The image quality is evaluated by verifying that the OCR confidence score exceeds a predetermined threshold, established through prior testing. If the quality is deemed insufficient, the scanned receipt is sent for manual data capture. It is out-of-scope from the auto-captured process because the low confidence score indicates that the OCR may not have accurately interpreted the data.

Store Scope Validation: Due to variations in receipt layouts across stores, the algorithm specializes in processing receipts from certain popular stores with longer receipts. These receipts are more complex and take longer to capture manually. If the image quality is acceptable, the algorithm checks if the receipt is from a store that it knows of before parsing it. Receipts from stores outside this scope are not suited for automated handling, so they are captured manually.

Parsing and Cleaning: For receipts in-scope, the text is parsed, cleaned and structured to be entered into the SHS database accurately. A machine learning entity classifier (described in section 2.2) and a spell corrector (described in section 2.3) were added to this step to improve the data extraction.

Figure 2-1

Visual representation of the auto-capture pipeline



2.1 OCR engines

The initial auto-capture algorithm used Tesseract OCR, but other OCR engines, including EasyOCR, PaddleOCR, KerasOCR, TROCR, and the Donut Transformer OCR, were explored to improve text recognition. The engines were evaluated on all receipt variables, including shop names, dates, and items. Speed was not critically important, but the engine needed to be compatible with and executable on the existing system.

Tesseract OCR (Smith, 2007) was the baseline engine for this study. Tesseract was developed by Hewlett-Packard (HP) in the 1980s as a proprietary software. It became open source in 2005, and Google maintained and updated it. It is a widely used OCR engine because it accurately recognizes and extracts text from images relatively quickly, while also supporting multiple customization options. One of the customizations is Page Segmentation Mode (PSM), defining how the OCR engine divides and processes a layout of text. The project's existing algorithm used PSM 6, which assumes a single uniform block of text since the receipts structure texts to uniform blocks. Tests with other PSM configurations showed that PSM 6 yielded the best performance, with no significant improvements observed in other settings. Despite its general reliability, Tesseract failed to recognize store names from certain store brands, suggesting limitations in handling specific font sizes and styles.

EasyOCR engine (Jaided, n.d.) was tested due to its straightforward, open-source implementation and broad language support. However, it exhibited difficulty in recognizing stylized or small text headers of some store brands. This limitation reduced its suitability for the project's requirements.

PaddleOCR (PaddlePaddle, n.d.) is an open-source practical lightweight OCR system built on PaddlePaddle (short for **Parallel distributed deep learning**), an open-source deep learning platform developed by Baidu Research. It demonstrated impressive performance but required pre-processed images to optimize accuracy. This included cropping images to reduce white space and adjust their height. While this dependency on image pre-processing added complexity, PaddleOCR provided consistent and accurate results, making it the default choice for the project.

TROCR (Microsoft Research, n.d.) is a text model, which uses a Transformer architecture for both image understanding and word piece-level text generation. Since the model relied on a text detection algorithm prior to its recognition step, Tesseract was used in conjunction with TROCR. Although TROCR achieved high accuracy, it was significantly slower compared to other engines and required GPU acceleration for practical use. Processing times were prohibitive, and alternative text detection algorithms needed to be explored for future improvements.

Two additional OCR engines were explored: KerasOCR (Morales, n.d.) is a Keras implementation of Convolutional Recurrent Neural Network for text recognition. Donut Transformer OCR (NAVER AI, n.d.) is a **document understanding transformer** that utilizes an OCR-free end-to-end Transformer model. However, their initial performance on sample receipts was not promising. The Donut Transformer OCR lacked the flexibility to return specific outputs based on the format of its training dataset. These results highlight the importance of selecting engines that align with specific project requirements.

After comprehensive testing, PaddleOCR was selected as the default OCR engine for this project, owing to its consistent accuracy with pre-processed receipt images. Tesseract with PSM 6 and other engines are supported as alternative OCR choices. Users can select another OCR option for their specific use cases.

2.2 Spell corrector

When OCR misrecognized a character of a word due to problems such as poor image quality or vertical inkless lines, a spelling corrector corrected it by using nearby information. A spelling corrector python package based on SymSpell was trained on a domain-specific vocabulary compiled from the SHS database and receipt entity keywords. If a word from OCR was not in its vocabulary, it replaced the word with a word in its vocabulary list that required the least amount of letter changes. It also relied on nearby words when correcting misspellings, but when multiple candidates were identified, the most frequent word found in the SHS database was selected.

2.3 Entity classifier

Regular expressions (regex) are sequences of characters used to define search patterns for matching text within an input text. Originally, regex patterns were used to identify entities, such as a payment mode, in a line of text. This method works well in practice, but machine learning classifiers were explored as a more robust option, capable of learning patterns in the data to handle complex or ambiguous structures. Nevertheless, regexes were preserved to identify dates because the receipts had standard formats for dates that were more accurately extracted using regexes.

To categorize all relevant entities, two different classifiers were developed: a receipt-level classifier and a product-level classifier. A receipt-level entity classifier was first trained to classify each line of text in a receipt into an entity. The entities included Department, Incomplete transaction, Payment mode, Start key, Stop key, Store name, Deposit, Discount, Eco fee, Price per quantity, Product name, Subsidy, Void and Other. The Start key and Stop key were lines that marked the respective start and end of the purchased product list. Lines between the Start key and Stop key were classified again using a product-level classifier that only predicted product entities: Product name, Deposit, Discount, Eco fee, Price per quantity, Subsidy, Void, and Other.

TF-IDF (Term Frequency-Inverse Document Frequency) processed texts to numerical features as input to a machine learning classifier. It is a numerical statistic used in natural language processing to reflect the importance of a term in a document relative to a collection of documents. N-gram terms of two to five characters were used.

Various supervised learning algorithms were trained and compared to identify the best-performing model for our task. A Decision Tree model was first tested because it was an interpretable model that recursively split the data based on feature values, making it a strong candidate for understanding how different features impact classification outcomes. Random Forest improved on the decision tree model by combining multiple trees trained on different sub-samples of the data. Random Forest can reduce overfitting, a common problem with individual decision trees, to improve model stability and generalization. Gradient boosting in XGBoost was an ensemble machine learning technique that built a series of decision trees. The new tree corrected the errors of the previous one using gradient descent to minimize the model's loss function.

The Multilayer Perceptron (MLP) had multiple layers of interconnected nodes to process information using an activation function. The weighted connections between nodes could learn complex patterns in data through iterative training algorithms. It was a valid choice when dealing with intricate datasets where simpler models may fall short.

Three types of support vector machine models (SVM) were trained. SVMs were discriminative classifiers that found the optimal hyperplane in an n -dimensional space, where n was the number of features, that maximized the margin between classes. An SVM with a linear kernel was a baseline model that assumed a linear decision boundary between features. A linear SVM with Stochastic Gradient Descent (SGD) was an efficient alternative for high-dimensional data. It estimated the gradient of the loss per sample, and the model was updated with a decreasing strength schedule. Finally, SVM with a Radial Basis Function (RBF) kernel was selected to capture complex relationships in non-linearly separable data.

3. Results

3.1 Performance of entity classifiers

To train and evaluate entity classifiers, labeled data were gathered from multiple sources. The SHS database included 74,008 Product entries, 13,544 Store entries, and 498 Department entries. Additional training data for other entities came by running SHS's existing auto-capture code on 1,469 receipts from the 2023 SHS, which had already been verified to be correctly parsed. This yielded 20,118 data lines, all which were crucial to train the machine learning models to recognize the various ways different stores format entities such as Deposits and Discounts. This included 12,538 unique lines that were deemed irrelevant for SHS and thus fall under the Other class. The existing regular expression patterns and the auto-generated data from these patterns were also valuable in capturing the full range of entity formats the auto-capture algorithm identified.

Table 3.1-1
F1 scores of classification models

Experimented Models	Receipt Classifier F1 Macro	Product Classifier F1 Macro
Decision Tree	40.4%	57.6%
XGBoost	77.4%	85.2%
MLP	78.8%	90.4%
Random Forest	83.2%	84.0%
Linear SVM with SGD	84.8%	89.8%
SVM with linear kernel	85.5%	90.4%
SVM with RBF kernel	87.4%	91.4%

To assess the model's ability to classify entities despite the class imbalance, each classification model was evaluated using a stratified 4-fold cross-validation method. Table 3.1-1 summarizes the different models tested and their F1 macro scores for receipt-level and product-level entities. F1 macro score was the determinant performance metric

because it weighs the importance of each class equally, meaning even if a class has very few examples in the training data, the quality of predictions for that class is just as important as a class with many examples.

Since the SVM with RBF kernel performed the best with F1 Macro of 87.4% for the receipt entities and 91.4% for product entities, it was used in the project.

3.2 Performance of the overall auto-capture

The performance of the existing and improved algorithms was compared using a test set of 3,620 receipts, which attributed to two months of scanned receipts from SHS. Key metrics included the overall number of receipts that can be auto-captured and the accuracy of receipt and product variables. The receipt-level metrics in Table 3.2-1 showed notable improvement (from 41.3% to 51.5%) in the percentage of auto-captured receipts with the new algorithm. Because of the naturally high OCR confidence scores from PaddleOCR, compared to the variability seen with Tesseract, none of the test receipts were below the confidence threshold to be considered out-of-scope when using PaddleOCR. As a result, the previous 8.0% of the receipts that Tesseract dropped due to low OCR quality were no longer being filtered out with PaddleOCR and were further processed. Additionally, the improved OCR recognized store names with fancy fonts better, leading to a 1.9% reduction in misrecognized stores and thus out-of-scope receipts.

Table 3.2-1
Percentage of auto-captured receipts

Metric	Existing algorithm	New algorithm
Total Receipts	3620	3620
Out-of-Scope – OCR Confidence	291 (8.0%)	0 (0%)
Out-of-Scope – Store Not in Scope	1,833 (50.6%)	1,754 (48.5%)
In-Scope Receipts	1,496 (41.3%)	1,866 (51.5%)

The accuracy of key variables, such as store name, date, and payment mode, (detailed in Table 3.2-2) showed minor but consistent improvements, maintaining high performance. The Payment Mode accuracy was newly measured through this experiment.

Table 3.2-2
Accuracy of receipt variables

Metric	Existing algorithm	New algorithm
Total Receipts In-Scope	1,496	1,866
Store Name Accuracy	1,411 (94.3%)	1,768 (94.7%)
Date Accuracy (Day and Month)	1,226 (84.6%)	1,581 (84.7%)
Payment Mode Accuracy	Not Calculated	1,628 (87.0%)

The product extraction quality also improved. Even though more receipts, and thus more product items, were to be in-scope to be captured, the number of missing items and extra items has decreased. Overall, the number of items properly captured has increased along with the text similarity of item descriptions and the accuracy of product prices. Auto-capture accuracy metrics for four additional variables, Department Name, Discount, EcoFee, and Deposit, were added for this experiment, as shown in Table 3.2-3.

The new additions to the algorithm successfully reduced recognition errors, increased recognition accuracy and improved automation efficiency. Key achievements include:

- **Auto-capture increase:** Receipts auto-captured rose from 41.3% to 51.5%.
- **Item description similarity:** Increased from 93% to 95%.
- **Price accuracy:** Improved from 76.9% to 86.8%.
- **New variables captured:** Added Department Name, Eco Fee, and Discount variables. Tested that they are auto-captured well with 96.2% accuracy on average.

Table 3.2-3
Accuracy of product variables

Metric	Existing algorithm	New algorithm
Total Items In-Scope	12,350	14,755
Items Properly Captured	11,481 (93%)	14,014 (95%)
Items Missing	869	741
Items Extra	3,107	1,747
Price Correct	8,834 (76.9%)	12,171 (86.8%)
Department Name Accuracy	Not Calculated	13,364 (95.4%)
Discount Accuracy	Not Calculated	13,148 (93.8%)
EcoFee Accuracy	Not Calculated	13,924 (99.4%)
Deposit Accuracy	Not Calculated	13,874 (99.0%)

4. Conclusion

This pipeline enhanced the accuracy of extracting receipt content by using advanced OCR methods, spell correctors, and entity classifiers. These improvements also increased both the number of auto-captured receipts and variables. The improved algorithm will be implemented for SHS 2025 in hopes of improving the auto-capture quality and reducing the manual effort required for capturing receipt variables.

Acknowledgements

The authors would like to thank Émilie Mayer for her contributions to this project. The authors are also grateful for the review and constructive comments from Denis Malo, Kimberley Flak, Olfa Khazri and Johan Fernandes.

References

- Jaied AI (2020), EasyOCR: Ready-to-use OCR with 40+ languages supported (v1.7.1) [Computer program], Retrieved from <https://github.com/JaiedAI/EasyOCR>.
- Microsoft Research (2021), TrOCR: Transformer-based Optical Character Recognition (v4.28.1) [Computer program], Retrieved from <https://github.com/microsoft/unilm/tree/master/trocr>.
- Morales, F. (2020), KerasOCR: A High-Level Deep Learning OCR Library (v0.8.5) [Computer program], Retrieved from <https://github.com/faustomorales/keras-ocr>.
- NAVER AI (2021), Donut: Document Understanding Transformer (v1.0.9) [Computer program]. Retrieved from <https://github.com/clovaai/donut>.
- PaddlePaddle (2020), PaddleOCR: End-to-End OCR Solution (v2.6.0) [Computer program]. Retrieved from <https://github.com/PaddlePaddle/PaddleOCR>
- Smith, R. (2007), “An Overview of the Tesseract OCR Engine”, In ICDAR’07, *Proceedings of the Ninth International Conference on Document Analysis and Recognition*. Retrieved from <https://github.com/tesseract-ocr/tesseract>