

AI Driven Accounts Payable Transformation

**Tarun Tater^{1*}, Neelamadhav Gantayat^{1*}, Sampath Dechu¹, Hussain Jagirdar¹, Harshit Rawat²,
Meena Guptha², Surbhi Gupta², Lukasz Strak², Shashi Kiran², Sivakumar Narayanan²**

¹IBM Research

²IBM Services

[ttater24, neelamadhav, sampath.dechu, hussain.jagirdar1, harsrawa, meenamga,surbhgup,[lukasz.strak@pl.ibm.com],
shkiran6 sivakumar.narayanan]@in.ibm.com,

Abstract

Accounts Payable (AP) is a resource-intensive business process in large enterprises for paying vendors within contractual payment deadlines for goods and services procured from them. There are multiple verifications before payment to the supplier/vendor. After the validations, the invoice flows through several steps such as vendor identification, line-item matching for Purchase order (PO) based invoices, Accounting Code identification for Non- Purchase order (Non-PO) based invoices, tax code identification, etc. Currently, each of these steps is mostly manual and cumbersome making it labor-intensive, error-prone, and requiring constant training of agents. Automatically processing these invoices for payment without any manual intervention is quite difficult. To tackle this challenge, we have developed an automated end-to-end invoice processing system using AI-based modules for multiple steps of the invoice processing pipeline. It can be configured to an individual client's requirements with minimal effort. Currently, the system is deployed in production for two clients. It has successfully processed around ~ 80k invoices out of which 76% invoices were automatically processed with low or no manual intervention.

Introduction

The finance function is no longer a cost arbitrage for a client. They are now looking for innovative solutions that not only deliver cost benefits but also deliver value. For organizations that are looking at cutting costs, automation holds the key (Furth 2005; Bohn 2010). The question is, how and what to automate? Traditionally, the trend has been to identify repetitive tasks that can be automated using automation scripts and robotics. Slowly, businesses are moving towards looking at automation opportunities involving high cognitive load. One of the major tools to achieve this has been machine learning(ML) components that can integrate with automation scripts to deliver at par or better than human-level performance while being more efficient. One such area where ML is making inroads is accounts payable, thus turning them into a profit center from a cost center. While technology surely holds the key, as rightly pointed out by (Bohn 2010), the dilemma industry watchers face is that there's no "one

size fits all" solution which leaves individual companies and departments to cobble up tools and solutions which might improve speed and accuracy.

Accounts payable is one of the finance functions which is extremely manual intensive (Schaeffer 2002; Cosgrove 2013). The process deals with procuring raw materials, services, and goods from listed vendors and paying invoices submitted by the vendors within contractual payment deadlines. Steps for managing the accounts payable process involves receiving invoices from vendors, matching vendor details from invoice to the business records using a set of dynamic business rules, and posting the invoices in the Enterprise resource planning (ERP) for payment. This process may also involve some conditional steps depending on the client and type of invoices. Such steps include performing business rules check for invoice compliance, invoice line item matching i.e., the line item description on the invoice is matched against that on the Purchase Order (PO), and goods receipt to perform a three-way match, and applying a tax code to each invoice item. Managing accounts payable is also a very important and daunting task since the invoices need to be paid accurately and on time to maintain a long-term relationship with the vendors (Schaeffer 2002) while avoiding any penalties. The challenges are amplified since it also needs to be ensured that duplicate, incorrect, and fraud invoices are not paid as that would result in losses, and retrieving such payments would bring in more costs.

The industry as a whole is moving towards the infusion of Artificial Intelligence (AI) and Machine Learning models in different business processes. But, the machine learning models come with a caveat. The predictions made by a model are not always right albeit offering very high accuracy. Thus, explanations corresponding to the predictions are important. Also, an automation system might look at some thresholds which if crossed would lead to the automation of a step in a process. Furthermore, the machine learning models need to adapt to changing rules for some processes. For instance, a tax code might change depending on the policies. Hence, there is an increasing need for the models to learn and re-evaluate the parameters and/or thresholds that they consider for a particular prediction as well as automating a process.

In this research, we aim to automate the complex process of invoice processing and auto-post as many invoices as possible without no or low manual intervention using AI-

*These authors contributed equally.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

enabled micro-services. This results in better efficiency and cost-saving resulting in better client, and vendor satisfaction and engagement.

Account Payable Process

In the current business-as-usual, the accounts payable process is largely manual intensive. Since no single platform is available to enable touch-less invoice processing from ingesting of invoices to payment, the majority of the steps in the invoice processing pipeline are done by agents. When a new invoice is received, the accounts payable agent or accountant manually reads and validates if all required information/fields are available on the invoice. Then, the agent checks whether an invoice is a Purchase order (PO) based or Non-Purchase Order(Non-PO) based. A PO-based invoice is an invoice for goods where a purchase order was placed before procurement. Whereas, a Non-PO based invoice is one where the services were used and then the organization was billed against it. If the invoice is PO-based, the agent then validates if the PO number mentioned in the invoice is present in the PO data. Other such validations include whether the invoice is billed to the receiving company and the billing address matches. There can be more validations around invoice amount, invoice dates, etc. depending on the requirements. After these sanity checks, the agent identifies the vendor matching the vendor name, address, and banking details from the invoice to the vendor data. Then, he/she enters all the details into the system manually. After that, for a PO invoice, he/she matches the invoice line items to PO line items and goods receipt considering line item descriptions, quantity, price, etc. Subsequently, in the case of a Non-PO invoice, he/she assigns accounting codes to process the invoice in the system, along with determining the tax code. The invoice is then posted to ERP for payment. Also, many of these documents and data may need to be retrieved from different applications and data stores. Some of these checks have been automated by Robotic process automation (RPA) depending on the client and complexity of invoices.

Currently, it is difficult to have a system that can be easily used by different clients for varying invoice processing setups. Along with this, the steps involved in invoice processing are manually intensive, cumbersome, and pose challenges because of the following issues:

- *Catalog Mismatch* : Buyers and Sellers maintain their catalogs which results in different terminology for the same item. This creates a problem while processing invoices since the line item descriptions in the invoice may not match with the description in PO for the same item. On the other hand, dissimilar items may be categorized as the same items and wrong payments may be made.
- *Abbreviations* : Shortened form of names, (Ex: Butter vs Bttr, etc.) introduced by either buyer/seller prevent traditional automation techniques such as RPA to perform line-item matching (fuzzy match).
- *Duplicate Invoices* : Some invoices may be resubmitted by vendors because of delay in payment or other issues. This causes the same invoice to be in the system multiple

times which might result in the organization paying the invoice amount multiple times.

- *Tax code and Regulations* : The tax code for an item(s) can change over time with changes in regulations. This creates an issue as the previous data would then become redundant.
- *Generalizable Solution across clients* : Different clients / industries have different invoice compliance rules. Some companies might also need only a subset of tasks due to the nature of invoices. For example, companies that deal with products follow only PO-based invoices where they need not follow steps such as Accounting Codes prediction. This makes it hard to find a generalizable approach for different accounts.
- *Knowledge Retention* : Processors and/or Associates are heavily dependent on Subject Matter Experts (SME) or clients for solving queries. Since there is no system to capture this user information and feedback, it can result in knowledge erosion on account of attrition.

There is no cost-effective platform for enabling touch-less accounts payable which is generalizable and easily configurable across different setups for invoice processing to the best of our knowledge.

Contribution

In this paper, we present a configurable end to end system for automating the Account Payable process using AI with the following key aspects:

1. We propose a novel way to orchestrate the invoice payment process using a flow authoring tool to meet different design requirements by various clients which are easily configurable with a drag and drop UI.
2. AI-enabled semantic similarity module for matching invoice line items to PO and goods receipt enabling three-way line-item matching. A hybrid classifier involving an information retrieval(IR) and rule-based solution to predict the tax and Accounting Ledger information.
3. Automatic Tuning of confidence thresholds based on agent's feedback at different confidence levels. The feedback mechanism can be incorporated with any of the individual AI-enabled modules helps to incorporate inter-agent agreement and fine-tuning the model in accordance with the received feedback.

Our system is successfully deployed for multi-national electronics distributor for 3 of its markets for more than a year and has already processed more than 70k invoices. It was recently also deployed for a multinational retail organization and has processed more than 10k invoices which highlight the usability of the proposed system.

System Architecture

Figure 1 describes different components that are available and a possible flow that can be orchestrated in the system. We will discuss about each component in detail in this section.

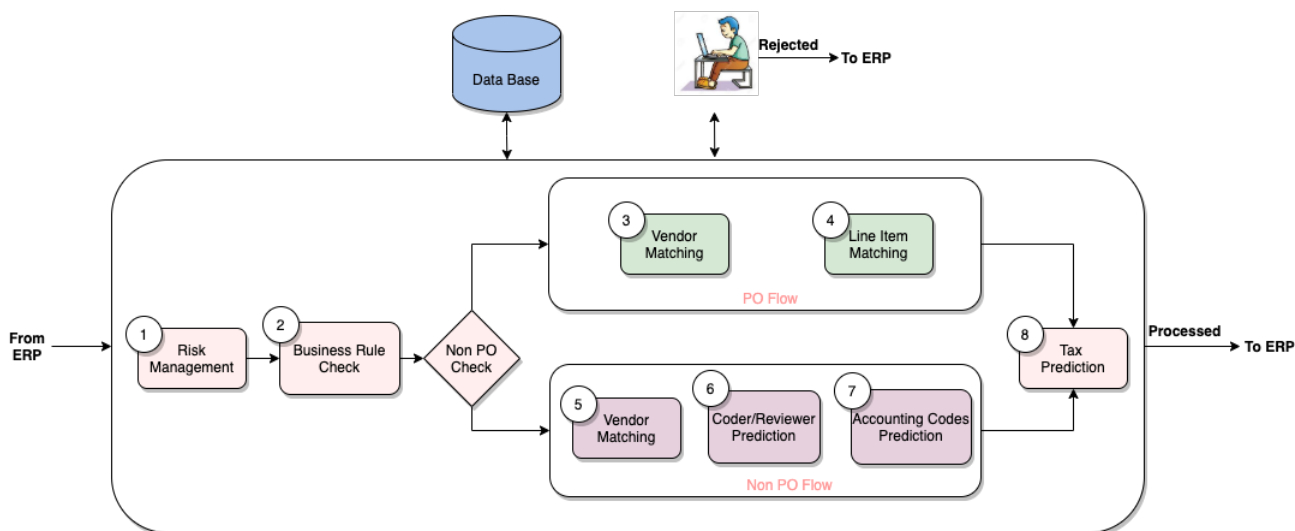


Figure 1: **System architecture.** We have a multi-stage architecture for our platform. The order of the different modules within the processing system (Risk Management, Business Rules Check, etc.) can be customized according to business requirements.

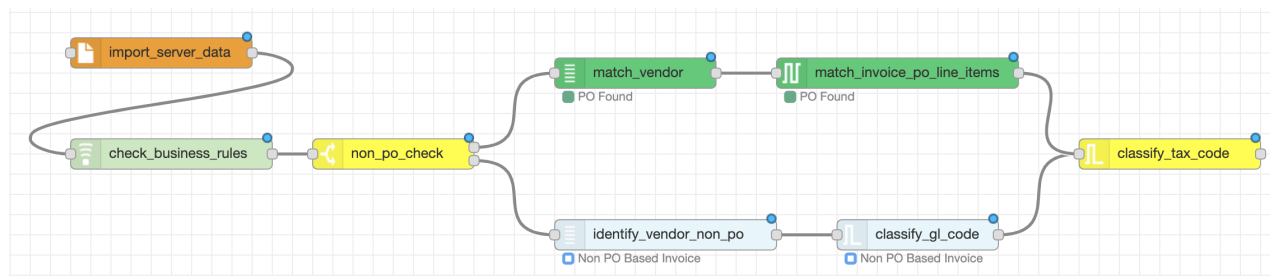


Figure 2: **Orchestrator flow :** One of the workflows used in a deployed system

Orchestrator

Orchestrator is mainly used for the deployment and management of various services in our system. It provides a way to select or deselect a particular service and also provides options to set default parameters for a particular service. Users can manage the default thresholds for all the AI components and also can define business rules such as allowed deviation in price etc. They can also be changed at any time enabling clients to perturb the thresholds as the system processes more and more invoices over time.

We used Node-red¹ - an opensource browser-based flow editor for this purpose. Our implementation contains different node-red nodes pertaining to each service. The nodes contain default values for the services which can be configured at the time of deployment. Figure 2 shows an actual flow orchestrated using node-red for a client. This design also helps to integrate any new module as per the requirement with minimal effort for integration.

Risk Management

Duplicate invoice payments are a common phenomenon in accounts payable. On average companies make anywhere

between 0.1 to 0.5 percentage duplicate payments(Wagner 2018). This can translate to huge amounts for a large company. This percent translates to a whopping \$500,000 to \$2.5 million amount for a large company with an annual turnover of \$500 million. With limited manpower and resources, it is very difficult to validate and verify all the invoices.

Even though some companies are Sarbanes-Oxley Act (SOX)² compliant and use a leading ERP system for invoice processing, they make duplicate payments. ERP systems can detect exact duplicates but they cannot detect a tinkered duplicate. Most organizations validate invoices based on the invoice number, invoice date, and amount. It's effortless to bypass these built-in controls, either accidentally or intentionally. Following are some of the scenarios where one might end up making a duplicate payment:

- Typographical errors on the invoice number.
- Vendor might send another invoice with a different date or a perturbed invoice amount, no matter how small.
- Duplicate Invoices may be submitted via different channels and locations.

¹<https://nodered.org/>

²<https://www.dnsstuff.com/what-is-sox-compliance>

- Utility payments (recurring payments) may arrive as a statement and not as an invoice. General AP practice is to enter them as *account#+month+year* e.g., 12345Jan2021. This increases the likelihood of potential duplicates if the AP accountant enters it as 12345Jan2021 vs 12345January2021.

It is extremely resource intensive to manually search for potentially duplicate invoices. We propose an automated risk detection solution to monitor transactions continually for duplicate and fraudulent invoices. Our system looks for duplicate invoices from the historical invoices on four parameters namely - invoice number, vendor name, amount, and invoice date. Following is the implementation of the system for identifying the individual parameter score:

Invoice Number Identify invoices with invoice numbers that closely match. Being an alpha-numeric string, our fuzzy match between two invoice numbers takes string sequence and number field separately. For instance, INVNO12345678 and INV12345678 are an example of duplicate invoice number as INVNO and INV could be abbreviations for invoice number and invoice respectively. Fuzzy matching is more powerful for detecting duplicates than only using exact matching. Similarity analysis is applied to invoice numbers to identify typos and transpositions. We implemented Levenshtein distance metric (Yujian and Bo 2007) based string similarity implemented in python³.

Vendor Name The next validation is to detecting duplicate vendors based on names, addresses, tax ids, bank accounts, and other attributes. We indexed the vendor names along with their attributes and used a fuzzy search technique to identify a closely related vendor. For the fuzzy search, we used FuzzyWuzzy (SeatGeek 2011) a python-based module to search for a given string.

Invoice Amount For pre-processing, we remove all the characters as well as special characters. Our algorithm gives a high score if the amount matches completely or if the deviation percentage lies below a certain threshold. We even index it as a string type for a fuzzy match with tight edit distance for cases where typos could occur. Eg. 1908 vs 1980.

Invoice Date After regularising the string, we index the invoice date as a string as well as in the date object. The risk score encompasses the fuzzy match between the string and the exact match between the dates.

We identify individual scores for these parameters and combine them to arrive at an overall score. We categorize all the invoices greater than a certain threshold as malicious and route them to a practitioner for further validations.

Better visibility into duplicate invoices and payments can help companies detect cash leakage, fraud, and misuse so that they can recover funds. Furthermore, the right tool will enable them to prevent duplicate payments in the first place, while combining and analyzing data from multiple payment platforms and providing workbench tools to build a strong audit trail. Further validations to determine the authenticity of invoices are done in the next step.

³<https://github.com/ztane/python-Levenshtein>

Business Rule Check: Despite some suppliers' adoption of digital/electronic invoices, many Accounts Payable (AP) processing organizations are impeded by paper-driven invoice processing. For an invoice to be processed accurately, it is required to extract the information from the invoice precisely. Accuracy of current Optical Character Readers (OCR) is good in the case of digital files whereas it is moderate to low in the case of scanned/manually written invoices, because of the quality of scan and the document.

PO Number Validation: This step is more relevant for PO-based invoices where the need is to check the authenticity of the PO number. This is done by referencing the PO number from the header or sub-header of an Invoice. In our system, this validation is done by referring the PO number back to the ERP's PO table or PO system to determine its validity. In the event of a match between the invoice and ERP system, invoice processing continues. However, if there is a mismatch, it is generally up to the AP staff (along with Procurement) to determine if the invoice submitted reflects the transaction details of a different PO or if it is entirely errant.

Tax, Freight, & Total Amount Validation: These fields are the most error-prone fields in a given invoice, so the system includes various validations to filter out counterfeit invoices. Following are some important rules:

- Filter out invoices with total amount zero.
- Compare the total amount of invoice against the sum of all the line item amounts.
- Review freight charges and tax charges. The sum of the tax amount, freight amount, and total amount should be equal to the invoice amount.


Business rules also include any other checks mandated by the client.

PO Invoice VS. Non-PO Invoice Categorization

This is an essential step for vendors dealing with both PO-based and Non-PO-based invoices. This step determines how the invoice needs to be processed. PO-based and Non-PO based invoices go through different steps before payment. For PO-based invoices, steps might include 2/3 way matching, and for Non-PO based invoices, steps may include Accounting codes classification & Approval routing. The inclusion of a PO Number on the Invoice is a typical way to distinguish between PO Vs. Non-PO invoices, though this is not always the case. Many clients follow *No PO No Pay* policy where only a approved list of vendors or invoices below the fixed threshold can be allowed under the Non-PO category. Hence, in some cases, determination may need to be made on other criteria like vendor details, shipping details, etc. to correctly assign it to the right process flow.

Vendor Matching

Most organizations require a vendor validation step in their AP process to ensure that the vendor submitting the invoice has an Approved Vendor status. This is to thwart fraud attempts and eliminate non-authorized spending. This step is done by a manual look-up by the AP staff into the Vendor Master File.



PURCHASE ORDER

Company Name
123 Main Street
Hamilton, OH 44416
(321) 456-7890
Email Address
Point of Contact

DATE
01/18/2021
PURCHASE ORDER NO.
A246
CUSTOMER NO.
114H

BILL TO:
ATTN: Name / Dept
Company Name
123 Main Street
Hamilton, OH 44416
(321) 456-7890
Email Address

SHIP TO:
ATTN: Name / Dept
Company Name
123 Main Street
Hamilton, OH 44416
(321) 456-7890


SHIPPING METHOD	SHIPPING TERMS	SHIP VIA	PAYMENT	DELIVERY DATE

ITEM NO.	DESCRIPTION	QTY	UNIT PRICE	TOTAL
A111	TRESemme Detox and Restore Shampoo, 580ml	1	\$150.00	\$150.00
B222	TRESemme Smooth & Shine Shampoo 340ml	7	\$80.00	\$560.00
			\$0.00	\$0.00
SHIPTOTAL				710.00
enter percentage TAX RATE				3.800%
TOTAL TAX				26.98
TOTAL \$				736.98

Remarks / Instructions:

THANK YOU

AUTHORIZED SIGNATURE _____ DATE 2/15/2016



INVOICE

Company Name
123 Main Street
Hamilton, OH 44416
(321) 456-7890
Email Address

INVOICE NO. 100001
DATE 02/15/16
CUSTOMER ID A246
TERMS Net 30 Days

BILL TO:
ATTN: Name / Dept
Company Name
123 Main Street
Hamilton, OH 44416
(321) 456-7890
Email Address

SHIP TO:
ATTN: Name / Dept
Company Name
123 Main Street
Hamilton, OH 44416
(321) 456-7890

DESCRIPTION	QTY	UNIT PRICE	AMOUNT
TRESemme Detox and Restore Shampoo, 580ml	1	150.00	150.00
TRESemme Smooth & Shine Shampoo 340ml	7	80.00	560.00
			0.00
SUBTOTAL			610.00
TAX (3.8%)			23.18
TOTAL			\$ 633.18

THANK YOU

Figure 3: Sample purchase order and its corresponding invoice

The steps to validate vendor differ for PO vs Non-PO based Invoices. In the proposed system, for PO-based invoices, vendor matching is fairly simple. As the vendor ID is present in the PO database, vendor details can be retrieved by a lookup from the vendor master record, and comparing attributes such as bank account number and vat ID of the vendor from invoice to the master record. However, for Non-PO based invoices, vendor identification is fairly complicated as the vendor information extracted from the invoice may contain some peculiarity in text. This makes it difficult to directly searching the vendor name in the vendor master record difficult. Most of the current ERP systems don't support fuzzy matching for text. So, we implemented a synthesized search by indexing vendor details. In our system we consider three fields, vendor details, vendor bank account number, and vendor's Value Added Tax identification number (VAT id). *Vendor details* is a composite field consisting of vendor name, city, and address. The system performs a fuzzy match on the vendor details and an exact match on bank account number and VAT id. For performing fuzzy matching we used a custom script utilizing Fuzzy-Wuzzy(SeatGeek 2011) which uses the Levenshtein distance (also known as edit distance) to compare two strings.

Line Item Matching

It is one of the most onerous components of processing a PO invoice. Line-item matching involves verifying that the items being billed were ordered (two-way match) and potentially to what was received (three-way match). In two-way matching, the line-items on the invoice are validated against that of PO. Whereas in three-way matching the line items on the invoice are matched against both purchase order and goods receipt. This includes using different parameters such as - description, quantity, price, material/part number, etc. Each vendor has its own set of weights for these parameters.

For some vendors, the description may be of the highest importance whereas others might match line items based on material/part number. Some other vendors may give precedence to quantity, price, etc. Another general practice followed by vendors include having a threshold for price. This is to accommodate various cases ranging from tax values discrepancies, handling charges, or price change. Another challenge here is that data needs to be referenced from multiple sources: invoices, procurement records (PO's), and receiving documents (Bills of Lading, Packing Slips, etc.). For some organizations, this is tiring due to lengthy supplier invoices, involving pages of line-item detail.

So the line item matching algorithm should consider all these parameters while matching between different line items. Another limitation in the development of this algorithm is the lack of training data, which limits us to only unsupervised techniques. The key component in our algorithm is semantic similarity apart from numeric similarity. And the algorithm is devised into two phases as described below.

We applied lexical normalization as given in (Han and Baldwin 2011) to the invoice line item string. Lexical normalization is the process of detecting 'ill-formed Out of Vocabulary (OOV)' words. Examples include but are not limited to misspellings, abbreviations, and common names. For example, people often refer to 'butter' as 'bttr'. Without such lexical normalization, existing corpus and knowledge-based approaches will not be able to detect proper nouns. Our system involves filtering out words that are not there in a standard dictionary. Calculate the fuzzy similarity score between the line-item string with the line-items from PO. To calculate fuzzy matching, each string is represented as a TF-IDF vector of bi-grams and tri-grams and then the cosine similarity between each pair of vectors is calculated.

Using fuzzy matching to compute a similarity score for strings 'Glycerine white distilled 12%' and 'Vinegar white




invoice				purchase_order					
Part No.; Desc	Quantity	Price	Net Amount	PO Material No.; Desc	Quantity	Price	Net Amount	Confidence Score	Feedback
490-CFM-5010-13-22; 7005089219 / CFM-5010-13-22 CU Devices DCFan 16.07CFM 1.68W / DC Fanslu00FE US HTS: 8414596590 ECCN: EAR99 COO: CN	10	7.79	77.90	CUICFM- 5010-13-22; UNKCUICFM- 5010-13-22	10	7.79	77.90	0.66	  

Figure 4: Line Item Matching Example with confidence score

distilled 12%’ returns a high score (> 0.7) because of a high number of matching bi-grams. Hence, we need to remove such string pairs from further comparison. We use python spacy⁴ API for noun phrase chunking. If the noun phrase(s) in the query string and pool of strings do(es) not match, they are not kept in the string pool for comparison.

After identifying the pool of products that match with the current line-item, the next step is to identify the overall score by considering the price, quantity, and other parameter matches. This is done by comparing the price and quantity of invoices with that of the purchase order. There is an allowed deviation threshold which is defined by the accounting expert. If the values are within the range then those products are given a higher score. An example is shown in Figure 4.

Accounting Codes Prediction

Accounting codes like general ledger account (GL account), profit center, and cost center are essential for every purchase of service to accurately account for the expenses in the right category. For a PO-based invoice, the accounting codes and approvals happen at the PO creation stage. Hence, the accounts payable(AP) team is responsible to match the invoice against the purchase order and/or goods receipt. However, for a Non-PO invoice, the AP team has to update the accounting codes along with obtaining review/approvals from the business. The accounting details are updated based on the type of expense mentioned on the invoice. Since this is a judgmental process, many times the team makes a mistake and is highly dependant on business teams for input. Hence automating this process reduces the dependency on business & helps populate correct accounting codes. Accounting codes identification depends on the line-item description along with shipping details. It is a challenge because the description mentioned on the invoice may not be the same as the description of the item ordered. Especially for a Non-PO based invoice, the line-item description may be very generic. For e.g.,: “Broadband bill” might mean “Phone bill”, “internet bill” or any other telecom related service. Hence, we try to identify invoices with similar descriptions and similar shipping details from the recent past.

For both accounting codes prediction, and Coder/Reviewer prediction, the approach is similar to tax code identification described in our previous work(Tater et al. 2021). We experimented with various algorithms including

random-forest, logistic regression, SVM, etc., for these predictions but figured that a hybrid classifier which is a combination of a semantic similarity engine and a rule-based system performed better in terms of accuracy along with high precision and recall. The Semantic Similarity Engine caters to extracting similar item descriptions from the historical data. This is currently done using an Information Retrieval system because of the nature of line-item descriptions but can be replaced by any other semantic similarity engine. These candidate matching descriptions are then searched in historical data to gather all items with these descriptions. Exact matching descriptions are given more score than similar matches. This is followed by the Rule Based Engine that filters the retrieved similar line-items based on shipping addresses (ship to, and ship from) and company code details. The results are then sorted based on time to give more importance to the most recent similar line items.

The accounting code is then predicted by majority voting of the top-N results where N is a configurable parameter based on subject matter expert(SME) knowledge and experimentation. The final confidence score (C_p) is calculated by:

$$C_p = D_s * W_d + M_s * W_m$$

where D_s denotes scores for description similarity and $0 < D_s \leq 1$ for exactly matching description, and by design D_s for fuzzy matches would be a lower value as compared to exactly matching descriptions in a particular configuration. Here, W_d is the configurable weight given for depicting the importance of line-item descriptions. And W_m is the configurable weight given to the majority score.

Whereas the majority voting score (M_s) is calculated as:

$$M_s = \frac{N_m}{N} * \frac{N_m}{N_T}$$

N refers to the number of shortlisted data samples which are being considered for classification. N_m describes the number of items with the majority accounting code out of the shortlisted candidate samples. N_T describes the number of distinct accounting codes in the N data samples.

Coder/Reviewer Prediction

For a PO-based invoice, we have the goods receipt step where the receiver who receives the goods or services confirms receipt of the goods or services. However, for a Non-PO invoice, this is not possible. So, it becomes essential to send the invoice to the person/department for confirmation

⁴<https://spacy.io/>

of receipt of goods or services. More often than not, the name/email ID of the person or department or department code or cost center is provided on the invoice for identification on who placed the order. Based on such information, the AP team needs to identify whom to send the invoice for review and approval. However, some invoices don't have this information, and even when they do, the information provided by vendors may not be consistent and accurate. The team may have to refer to multiple records or databases for identification. We tackle this problem by multiple approaches : (i) If identifying information is available, the system checks for the most similar person/department to the information provided using fuzzy search. (ii) If the identifying information is missing, we use a similar approach as Accounting Codes prediction using an IR-based system on line-item description to identify people/departments who have received the goods/services in the recent past.

Tax Code Prediction

This step involves assigning the correct tax code to each item in the invoice which can be dependent on one or more attributes including (i) Item Description (ii) Where the Item was shipped From (iii) Where the Item is shipped To (iv) Vendor Details (v) Time of Purchase. For tax code identification, a semantic similarity engine searches for similar line-item descriptions with descriptions in historical data. These candidate matching descriptions are then searched in historical data to gather all line-items with these descriptions. They are then filtered based on the matching shipping addresses and company codes and matched for the vendor if present. The results are then sorted based on time to cater to the challenge that the tax code might have changed over time for provided details, thus tackling any concept drift. The tax code is then predicted by majority voting of the top- N results. The approach is very similar to tax code identification described in our previous work (Tater et al. 2021). Similar to accounting code prediction, the predicted confidence (C_p) here is given by:

$$C_p = D_s * W_d + M_s * W_m + V_s * W_v + T_m * W_t$$

where sum of weights for different components = 1 i.e. $W_d + W_v + W_t + W_m = 1$ Here, W_t is the weight (importance) assigned to the tax percentage listed on the invoice, if present. To be noted is that the same tax percentage can have multiple tax codes associated with it. T_m is 1 if predicted tax-code is one of the tax codes associated with the tax percentage and -1 if not. V_s denotes the similarity score for vendor details, and W_v is the configurable weight given for vendor details.

Learning From Feedback

The above described AI modules namely line-item matching, coder prediction, accounting codes prediction, and tax code prediction also involve two thresholds of confidence that decide the flow of each item: (i) minimum confidence (C_{min}) and (ii) maximum confidence (C_{max}). If the confidence exceeds C_{max} , the prediction is considered correct and the step is passed without any manual intervention. However, if the predicted confidence (C_p) is between

C_{max} and C_{min} , that is $C_{min} < C_p < C_{max}$, the line-item for that module is listed for feedback/review by a human agent. The agent can then choose to give an upvote, which would indicate that the prediction is correct, or give a downvote and correct the prediction. This feedback is then used to improve the underlying model. In case the predicted confidence is lower than the minimum defined threshold, i.e. $C_p < C_{min}$, the complete invoice is returned to the workflow for manual processing.

When the system is deployed for a new client, the confidence thresholds (C_{max} and C_{min}) are conservative such that C_{max} is set for a very high value and C_{min} is set to a very low value. This is termed a hyper-care period where for a few weeks the agents validate all the predictions. Then, after some time, analyzing agents' feedback, these thresholds are adjusted to maintain a high accuracy rate while reducing the manual effort. The lowering of C_{max} would result in more invoices being auto-posted.

For the Line-item matching algorithm, we have two hyper-parameters that update the overall confidence score based on user feedback. The hyper-parameters are tuned in a way that, for positive feedback, the learning rate is very slow whereas for a negative vote the learning rate is very high. These hyperparameters ensure our algorithm rightly identify matches with enough positive/negative feedback. Similarly, for tax code or accounting codes identification, each *upvote* (agreement) or *downvote* (disagreement) for a prediction changes the confidence of predicted class. This is also considered as a new data point and added to the learning module. This way, the model also takes into account the inter-agent agreement for the feedback. In essence, the feedback mechanism provides the system with a self-learning capability, combined with knowledge retention and tackling any data drift or concept drift.

Business Impact

The proposed system is deployed for two accounts, a major multi-national electronics distributor and a multinational retail organization to process their PO-based invoices with a combined annual volume of 800k invoices. Currently, the system is deployed for these clients in Europe, and North America and the plans are to roll it out to more markets in Europe Latin America, and Asia. At present, for the electronics distributor clients, it has processed ~70k invoices delivering efficiency of $\geq 76\%$ (transactions which required either no or minimal human intervention). The number of invoices requiring complete manual processing is only 7555 (< 11%) which includes 6631 duplicate invoices. This means 62154 (> 88%) invoices were processed through our system as detailed in Table 1. For the other client, the system has processed 10k invoices but it is still in *hypercare* mode where each invoice is also reviewed by an agent. Building on these successful deployments, the system is being expanded to 37 more markets globally for these 2 clients.

For the markets where the system is already live, the annual expected volume is 88k invoices where each invoice can have multiple items which need to be validated, matched, processed and a tax code needs to be applied. On average, it takes 10 minutes for a human agent to manually process

Total Invoices Inflow	Ready for Posting	Waiting for Feedback	Potential Duplicates	Confirmed Duplicates	Require Manual Intervention
70442	62154	569	164	6631	7555

Table 1: Current Dashboard Statistics of the number of invoices inflow, posting, manual intervention and duplicates

one invoice after indexing to posting it to ERP. Even with a very conservative estimate of saving of 3 minutes per invoice, this has already amounted to savings of 4000 work-hours considering 80k invoices are already processed. Once the system goes live in other markets where it is in the deployment phase, taking into consideration the annual volume of 800k invoices, the projected savings would be 40k work-hours annually. Also, this reduction in time is expected to be a lot larger once clients get more comfortable and trustworthiness of the AI skills.

Another major portion of time consumed for invoice processing is because it involves various touchpoints by various agents and passing of invoices between these agents involves significant wait time. With this system, since the process becomes automated and touchless, we have achieved a $\geq 90\%$ compression of end-to-end cycle time between from invoice receipt to invoice posting. Our system also caters to identifying duplicate invoices. To date, the system has been able to detect > 25 million \$ worth of duplicate invoices.

Another advantage of our system is that the AI modules are expected to learn and improve with the agent feedback as more and more invoices flow through. This would enable the confidence thresholds to be adjusted accordingly and more invoices would auto-post without any manual intervention because of higher confidence predictions. Another major challenge as previously pointed out is the generalizability, reusability, and scalability of our system to other clients, the drag and drop integration of individual modules i.e. the modular and configurable design enables us to quickly deploy the system for another client with minimal changes and effort. Its also in early deployment for three more clients now that it has proven its edge over manual methodology.

Deployment and Maintenance

We have a development team of around 30 people who manage this product following an agile development methodology. There are monthly releases consisting of new enhancements and bug fixes, following our own DevOps pipeline to build, test and deploy the solutions to production. The system is currently deployed as a single-tenant service with 3 separate docker images, each handling different stages and functionalities. The first docker image caters to flow authoring via node-red, which helps in orchestrating the invoice processing pipeline written in NodeJS. An example workflow is depicted in Figure 2. The second docker image caters to the front-end interactions for agents and admins. This was developed using AngularJS and NodeJS. The third docker entails all the micro-services for each step in the invoice processing pipeline. This is developed using python and APIs were created using python flask. Also, we have

a separate docker image for each micro-service if a client wants to deploy only specific micro-services. This decoupling helps to ship different parts of the system separately. Various machine learning models were trained using open source scikit-learn⁵ library (Pedregosa et al. 2011). The IR system works on top of open source pylucene⁶ library. The database used for storing historical and processing data is mongodb⁷. There is also a governance dashboard which keeps track of the status of invoices similar to Table 1.

The maintenance of the system includes adding new classes (new tax codes, GL codes, coder names, etc.) when necessary; adjusting confidence thresholds as the system improves with feedback, and building new rules if needed with changing business requirements. Maintenance also entails including any corrections for predictions reported in any audits. With respect to deployment for a new client, very few changes are required because of the modular and configurable design: (i) The individual model needs to be trained on client-specific data. (ii) A drag-and-drop configuration of modules as per the client requirements for their invoice processing pipeline. (iii) A one time configuration of different weights and confidence thresholds (25 parameters) for different microservices.

Related Work

The work by (Hedberg 2020) analyses the use of ML and decision support system for invoice processing and highlights the benefits and challenges associated with it. It also shows that there is a high variety in how cost centers and accounts are perceived in different organizations, and the complexity of invoices varies with different parameters. (Doshi, Kotak, and Sahitya 2020) and (Desai et al. 2021) highlight the importance of automation and Robotic Process Automation (RPA) in invoice processing. Smirnov et. al. (Smirnov et al. 2016) , Hu et. al. (Hu 2015) and Tater et. al. (Tater et al. 2018) also discuss various ways of identifying invoice payment status using machine learning algorithms. Some work is available on individual invoice processing steps like similarity between line items in purchase order and invoices as discussed in (Maurya et al. 2020). Also, our work (Tater et al. 2021) discusses tax code determination following an approach where we consider historical data for tax code classification. Data drift and concept drift have also been considered in other works like (Quionero-Candela et al. 2009) (Widmer and Kubat 1996; Gama et al. 2004).

Other companies are using a variety of solutions such as: robotics process automation (RPA), rule-based solutions,

⁵<https://scikit-learn.org/stable/>

⁶<https://lucene.apache.org/pylucene/>

⁷<https://www.mongodb.com/>

and electronic invoicing platforms. The major challenge with these solutions is they primarily work with structured data or for repetitive tasks. RPA is a good solution for repetitive tasks with low or zero cognitive load. Similarly, rule-based solutions would again be applicable for a fixed set of rules. Also, not every organisation is large enough in terms of volume of invoices to have its own electronic invoicing platforms which entail building web applications for vendors to input their invoice in a specified format and then processing it. On the other hand, our system is catered to handle unstructured data such as line item description and uses ML algorithms to tackle complex tasks as well.

Conclusion and Future Work

We have developed a configurable system for automatic accounts payable invoice processing where each independent module can be used or dropped depending on clients' requirements. Some modules like line-item matching, accounting codes prediction, coder name prediction, and tax code prediction use supervised and unsupervised algorithms, while others are rule-based. Some of these modules also use feedback from agents to improve their performance. This helps in tackling any changes or drifts in the data along with addressing the problem of unseen data. These agent feedback also help in fine-tuning the confidence thresholds, thus pushing more invoices directly for posting over time without any manual intervention. Also, owing to the modular design of the system, each module can be separately changed/improved based on data-characteristics or client requirements or in lieu of a better performing algorithm. We have successfully deployed the system for 2 clients in multiple markets and are in the process of deploying it for more markets and clients with minimal changes. This highlights the generalizability and scalability of the proposed system.

Aided by extensive logging and the data collected with the system, clients can better analyze vendor relationships. As part of future work, we need to come up with ways in which new rules or modules can be directly added by clients themselves if a particular need arises. Also, the industry as a whole needs to look at more explainable use of AI modules which makes it comfortable for new clients to adopt such automation systems since some new clients are wary of using such systems because of audits and regulations.

References

- Bohn, T. 2010. Cost-cutting with accounts payable automation. *Financial executive*, 26(6): 65–67.
- Cosgrove, C. 2013. Invoice Exceptions Should Be Like Caution Signs For Your AP Process. <https://www.cloudxdpo.com/blog/bid/220099/Accounts-Payable-Process-Automating-Manual-Validation-Steps>. Accessed: 2021-12-27.
- Desai, D.; Jain, A.; Naik, D.; Panchal, N.; and Sawant, D. 2021. Invoice Processing using RPA & AI. *Available at SSRN 3852575*.
- Doshi, P.; Kotak, Y.; and Sahitya, A. 2020. Automated Invoice Processing System Along with Chatbot,“. *International Journal of Research in Engineering, Science and Management*, 3(5): 29–31.
- Furth, D. 2005. Accounts payable automation pays dividends. *The CPA Journal*, 75(7): 16.
- Gama, J.; Medas, P.; Castillo, G.; and Rodrigues, P. 2004. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, 286–295. Springer.
- Han, B.; and Baldwin, T. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 368–378. Association for Computational Linguistics.
- Hedberg, N. 2020. *Automated invoice processing with machine learning: Benefits, risks and technical feasibility*. Ph.D. thesis, KTH, School of Industrial Engineering and Management.
- Hu, P. 2015. *Predicting and improving invoice-to-cash collection through machine learning*. Ph.D. thesis, Massachusetts Institute of Technology.
- Maurya, C. K.; Gantayat, N.; Dechu, S.; and Horvath, T. 2020. Online similarity learning with feedback for invoice line item matching. *arXiv preprint arXiv:2001.00288*.
- Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12: 2825–2830.
- Quionero-Candela, J.; Sugiyama, M.; Schwaighofer, A.; and Lawrence, N. D. 2009. *Dataset shift in machine learning*. The MIT Press.
- Schaeffer, M. S. 2002. *Essentials of accounts payable*. John Wiley & Sons.
- SeatGeek. 2011. Fuzzy String Matching in Python. <https://github.com/seatgeek/fuzzywuzzy>. Accessed: 2021-12-27.
- Smirnov, J.; et al. 2016. *Modelling late invoice payment times using survival analysis and random forests techniques*. Ph.D. thesis, Universitas Tartuensis.
- Tater, T.; Dechu, S.; Gantayat, N.; Guptha, M.; and Narayanan, S. 2021. Tool for Automated Tax Coding of Invoices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17): 15185–15194.
- Tater, T.; Dechu, S.; Mani, S.; and Maurya, C. 2018. Prediction of invoice payment status in account payable business process. In *International Conference on Service-Oriented Computing*, 165–180. Springer.
- Wagner, J. 2018. Duplicate Invoice Payments: How to Avoid Cash Leakage in Accounts Payable. <https://www.oversight.com/blog/duplicate-invoice-payments-avoid-cash-leakage-accounts-payable>. Accessed: 2021-12-27.
- Widmer, G.; and Kubat, M. 1996. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1): 69–101.
- Yujian, L.; and Bo, L. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6): 1091–1095.