Distinctive Bag-of-Words Modeling for Business Data Extraction

Mouhssine Rifaki
mouhssine.rifaki@psl.eu

***Abstract*—I developed a bag-of-words approach for recognizing business data. To identify each field of interest, I constructed sets of potential features that capture both layout and textual attributes. These features were weighted to emphasize the key factors that differentiate each field. The process involved feature selection, fine-tuning thresholds, and comparing different models. As a result, I achieved a training error rate of 8.81% and a testing error rate of 13.99%.**

I. Introduction

Invoice processing is one of the most critical tasks for the financial department of any organization. In many of such departments, invoices are still examined and entered manually, a process that is slow, costly, prone to human errors, and has become a bottleneck of high-speed data processes especially when the number of invoices grows dramatically with the development of the social economy [1]. While a standard list of critical fields is usually visible in almost all invoices, the choice of keywords and layout can vary largely from vendor to another, creating the challenge of extracting structured information from unstructured documents in an attempt to automate such invoice recognition and entry process.

The invoice recognition model proposed in this project aims to provide deeper insights into addressing this challenge. Eight fields of interest, including a negative class, are identified and processed according to the workflow illustrated in Fig. 1.

The raw data for this model consists of scanned invoice images. Following image processing, OCR, and pattern-matching steps, word groups (tokens) and their coordinates are extracted from the original images. These extracted elements serve as the model's actual inputs. From these inputs, feature sets are created based on predefined selection rules to capture a range of layout characteristics and word patterns specific to each field. The features are then evaluated and weighted using three classification models (Naive Bayes, Logistic Regression, and Support Vector Machines) to predict the corresponding field for each word group.

II. Related Work

Various approaches in image processing and machine learning have been explored to address the challenges of invoice recognition from multiple perspectives.

Image processing methods often focus on detecting columns and recognizing word sequences within logically segmented regions [2], sometimes incorporating machine learning techniques to enhance the accuracy of region classification. While these techniques [3] can significantly support other models, relying solely on an image processing-based recognition algorithm oversimplifies the complexity of invoice layouts. Such methods often assume consistent properties in region segmentation using linear rule combinations, which rarely hold true in practice due to the highly variable and unpredictable nature of invoice designs.

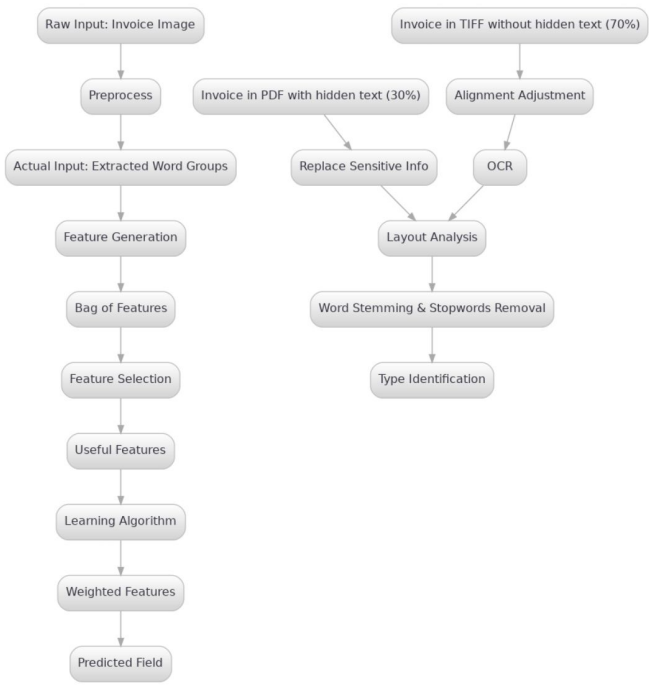

Fig. 1: Flow of Invoice Recognition

To address the need for more sophisticated models, template-based classification algorithms have been introduced. These methods rely on the template of an invoice, represented by a set of layout attributes, which is either matched to a predefined template library [1] [4] or grouped into a cluster of templates with similar characteristics [5]. In both approaches, the template library or clusters are dynamically expanded whenever a clear match cannot be found.

Template-based models offer the clear advantage of recognizing an entire invoice in one step using pre-established, template-specific rules. They tend to perform exceptionally well with high-quality images and highly distinctive templates. However, these ideal conditions are rarely guaranteed in practice. Invoices are often poorly scanned, and the vast number of vendors—and thus invoice templates—frequently results in documents with similar structures but slight (and often critical) variations in layout and field arrangement. For template libraries, this can lead to misrecognition of key fields due to the application of incorrect rules. For clusters, challenges arise in defining template "distance" and identifying subtle but significant differences within a cluster, making the process complex and error-prone. Additionally, these models are memory-intensive, as the library or cluster size continues to grow over time.

Rule-based models are another available approach, relying on sets of hand-crafted rules that are weighted to capture fine-grained details for each field [6]. These models avoid the rigidity of treating the entire invoice as a single unit. While hand-crafted rules can be effective for invoices with industry-standard components and layouts, they often rely on assumptions about field properties that may be arbitrarily incorrect (e.g., amounts are not always right-aligned) or impractical (e.g., matching price and quantity with the total amount to identify invoice lines, even though many real-world invoices lack one or more of these fields). The model developed in this project also recognizes fields individually, but instead of only assigning weights to predefined rules, it utilizes a bag of potential features to determine the optimal set of rules for each field.

III. DATASET AND FEATURES

A. Dataset Generation

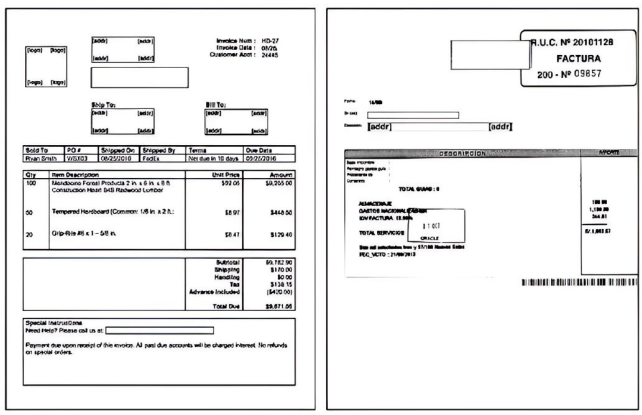

Fig. 2: (Left): Invoice with Hidden Text, and (Right): Invoice without Hidden Text

A total of 97 raw invoice images were sourced from Oracle Corporation's internal testing library, with sample images shown in Fig. 3. To protect sensitive information, tokens such as LOGO and ADDR were manually inserted in place of actual text, with their positions preserved by labeling the corners of bounding boxes. While some of the invoices are well-structured PDF files containing hidden text, the majority are TIFF images that require preprocessing before applying PDF Layout Analysis [6] to extract word groups. The workflow for generating word groups and their corresponding coordinates, which serve as the training input, is illustrated in Fig. 3.

The TIFF images are first aligned by calculating the optimal rotation angle using the Hough Line Transform [7]. Optical Character Recognition (OCR) [8] is then applied to extract any textual groups present in the images. All sample invoices are subsequently processed using PDF Layout Analysis to retrieve and store the coordinates of the tokenized textual groups. Additional word processing is performed on each token, including Porter2 word stemming [9], removal of stopwords [10], and type identification using regular expressions. Five specific types—DATE, MONEY, NUMBER, TELE, and EMAIL—are defined for this final step, replacing exact textual values with more generalized representations.

B. Feature Generation

For each token, the following set of feature selection rules is applied: horizontal alignment with other tokens, vertical alignment with other tokens, proximity to nearby tokens within a defined distance threshold, overall vertical position, and the token's type. The features generated for each token are then aggregated to create a bag of potential features, where binary values indicate the presence or absence of a particular feature for a given token. For example, if token A is horizontally aligned with token B, the feature *B_halign* will be assigned a value of 1 for token A and 0 otherwise. These feature selection rules are designed to capture fundamental layout information without making assumptions about any standardized template.

Approximately 8000 features were generated for the 2095 tokens ($m = 2095$) extracted from 97 invoices. These features were then reduced through an initial pruning process (Section III-A) to remove those that appeared only once, as they are not considered indicative. The remaining features ($n \approx 2000$) were subsequently subjected to a detailed feature selection process, as outlined in Section V.

IV. METHODS

Using the scikit-learn library [11], three machine learning models were trained and employed for predictions: Multinomial Naive Bayes, Logistic Regression, and Support Vector Machines. In all three models, the class labels were

defined as $y \in [0, 7]$, where 0 represents the negative class, and 1 through 7 correspond to the following fields: invoice number, invoice date, total amount, PO #, payment terms, due date, and tax, respectively.

A. Multinomial Naive Bayes

In the Multinomial Naive Bayes model, it is assumed that the features x_i are conditionally independent given the class labels y. This model was used with Laplace smoothing to estimate the parameters $\phi_{y=k} = p(y = k)$ and $\phi_{j,y=k} = p(x_j = 1|y = k)$, where $k \in [0, 7]$, to maximize the joint likelihood of the data, expressed as:

$$L(\phi_{y=k}, \phi_{j,y=k}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

The maximum likelihood estimates for these parameters are given by:

$$\phi_{y=k} = \frac{\sum_{i=1}^{m} 1(y^{(i)} = k)}{m},$$

$$\phi_{j,y=k} = \frac{\sum_{i=1}^{m} 1(x_j^{(i)} = 1, y^{(i)} = k) + 1}{m + K}.$$

After fitting these parameters, predictions for a new example with features x are made by calculating the posterior probability for each class k as follows:

$$\begin{aligned} p(y = k|x) &= \frac{p(x|y = k)p(y = k)}{p(x)} \\ &= \frac{\prod_{i=1}^{n} p(x_i|y = k)p(y = k)}{\sum_{l=1}^{K} \prod_{i=1}^{n} p(x_i|y = l)p(y = l)}, \end{aligned}$$

where the probabilities are replaced with their corresponding parameter estimates. The class with the highest posterior probability is selected as the prediction.

B. Logistic Regression

In the Logistic Regression model, the multiclass classification problem is transformed into a binary classification problem using the one-vs-rest approach. This means that when calculating the probability for a specific class k, all other classes are treated as a single label. In binary Logistic Regression, the following hypothesis is used to make predictions:

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}.$$

This hypothesis represents a logistic (sigmoid) curve that smoothly transitions from 0 to 1. Predictions are made by assigning a label of 1 if $h_\theta(x) > 0.5$, and 0 otherwise. The logistic loss function is then defined as:

$$L(z, y) = \log(1 + \exp(-yz)) = \log(1 + \exp(-y\theta^T x)),$$

where $z = \theta^T x$. The loss is minimized when the margin yz is large and maximized when the margin is small. To address overfitting, l_2-regularization is applied. A slightly modified version of the empirical regularized risk function is minimized:

$$\begin{aligned} J_C(\theta) &= \frac{C}{m} \sum_{i=1}^{m} L(\theta^T x^{(i)}, y^{(i)}) + \frac{1}{2}\|\theta\|_2^2 \\ &= \frac{C}{m} \sum_{i=1}^{m} \log(1 + \exp(-y^{(i)}\theta^T x^{(i)})) + \frac{1}{2}\|\theta\|_2^2. \end{aligned}$$

The parameters θ are fitted using the scikit-learn implementation, which solves the dual optimization problem of l_2-Regularized Logistic Regression through coordinate descent [12].

C. Support Vector Machines

Similar to Logistic Regression, the one-vs-rest approach is applied to transform the problem into a binary classification task with labels $\{-1, 1\}$ for all classes. The margin-based loss function used is defined as:

$$L(z, y) = [1 - yz]^+ = \max\{0, 1 - yz\},$$

where $z = \theta^T x$. This loss function becomes zero as long as the margin yz is greater than 1, indicating that the model has made a correct prediction with sufficient confidence.

To fit the model, the empirical l_2 regularized risk is minimized, expressed as:

$$\begin{aligned} J_C(\theta) &= \frac{C}{m} \sum_{i=1}^{m} L(\theta^T x^{(i)}, y^{(i)}) + \frac{1}{2}\|\theta\|_2^2 \\ &= \frac{C}{m} \sum_{i=1}^{m} [1 - y^{(i)}\theta^T x^{(i)}]^+ + \frac{1}{2}\|\theta\|_2^2. \end{aligned}$$

The scikit-learn implementation of LinearSVC uses a linear kernel K. According to the representer theorem, $z = \theta^T x$ can be implicitly expressed as $\sum_{i=1}^{m} \alpha_i {x^{(i)}}^T x^{(i)}$. This enables the use of the kernel trick, rewriting the empirical regularized risk in terms of α:

$$J_C(\alpha) = \frac{C}{m} \sum_{i=1}^{m} [1 - y^{(i)} K(i)^T \alpha]^+ + \frac{1}{2}\alpha^T K \alpha,$$

where $K(i)$ represents the ith column of the Gram matrix of kernel K. The LinearSVC implementation determines the parameter α by solving the dual optimization problem in the C-Support Vector Classification formulation of SVM [13].

After fitting the parameters, the model makes predictions based on the value of $z = \theta^T x$.

V. RESULTS AND DISCUSSION

A. Metrics

Given that, in any invoice, there are significantly fewer fields belonging to classes 1-7 compared to those classified as "Other," the data is heavily skewed toward class 0. As a result, the overall model accuracy (percentage error) is not the most informative metric, since strong performance on

the dominant class can yield high accuracy regardless of the model's performance on the minority classes. To provide a more balanced evaluation, I computed the precision, recall, specificity, and the corresponding F1 scores for all classes. The F1 score, defined as the harmonic mean of precision and recall [14], combines these two metrics, which often trade off against each other, into a single measure:

$$\text{precision} = \frac{TP}{TP+FP}, \quad \text{recall} = \frac{TP}{TP+FN},$$
$$\text{specificity} = \frac{TN}{TN+FP}, \quad F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

In some cases, however, it is useful to have a single metric that reflects performance specifically on the minority classes (all classes except "Other") in the trade-off space. For this purpose, I calculated the average F1 score for all classes, weighted by the frequency of each class label. To ensure that the contribution of the majority class does not dominate this metric, the "Other" class was excluded when computing the weighted average F1 score.

B. Regularization Parameter Scaling

A regularization term is added to both Logistic Regression and SVM, as shown in Eq.(1-5), to reduce overfitting. The parameter C serves as the regularization control, balancing the cost of misclassification on the training data [15]. Fig. 4 illustrates the relationship between C and the F1 score for both l_1- and l_2-regularization.

From the figure, it is evident that performance deteriorates when C is either too small or too large. A small C reduces the penalty for misclassification on the training data, leading to an overly "soft" hyperplane margin that risks underfitting. Conversely, a large C imposes a high penalty for non-separable points, forcing the algorithm to fit the data more strictly, which increases the risk of overfitting.

Although l_1-regularization is computationally more efficient for sparse data, Fig. 4 shows that l_2-regularization generally provides better optimization for the average F1 score, which is the primary metric of interest. Therefore, l_2-regularization is applied in both Logistic Regression and SVM for this model. The optimal regularization parameters are $C = 10.72$ for Logistic Regression and $C = 0.58$ for SVM.

Table I presents the final training and testing errors for each classifier after parameter optimization. The results confirm the effectiveness of my regularization approach, particularly for SVM which achieves the lowest error rates despite using fewer features than Logistic Regression. Notably, while Naive Bayes exhibits minimal overfitting with only a 0.26% difference between training and testing errors, its overall performance lags behind the other methods. This suggests that the added complexity and careful regularization of SVM and Logistic Regression are justified by their superior classification accuracy, even though they require more computational resources during training.

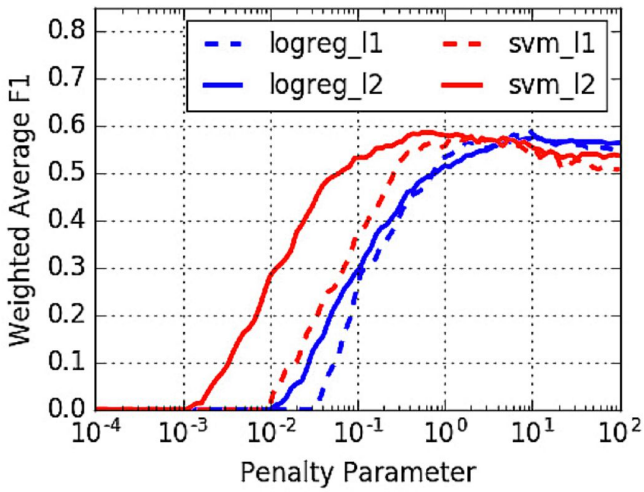

Fig. 3: Regularization

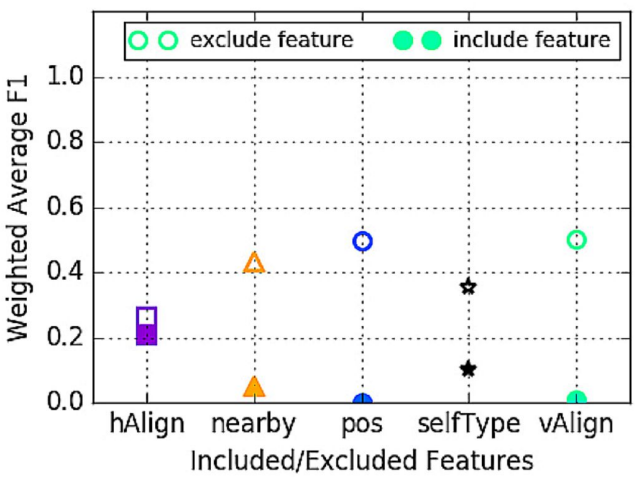

Fig. 4: Feature Rules Effectiveness

C. Feature Selection

Fig. 4 evaluates the effectiveness of each feature selection rule. Solid markers represent the weighted average F1 score when only that rule is included (inclusion set), while hollow markers represent the case when only that rule is excluded (exclusion set).

It is evident that horizontal alignment (hAlign) is the most effective rule, showing the best performance in the inclusion set and the worst in the exclusion set. This is followed by self type (selfType), nearby, vertical alignment (vAlign),

and position (pos). The superior performance of hAlign is expected, as many fields of interest, regardless of their absolute position, are horizontally aligned in a significant number of invoices. Conversely, while the absolute vertical position was moderately indicative with smaller datasets, the variety in invoice templates increases substantially with larger datasets, making it harder to derive common structural properties and reducing the effectiveness of this rule.

After the rule-level analysis, filter-based feature selection is applied to individual features to exclude less indicative ones generated by effective selection rules. For each feature, the empirical distributions $p(x_i)$, $p(y)$, and their joint distribution $p(x_i, y)$ are used to compute the mutual information $MI(x_i, y)$ between x_i and y. This is calculated using the Kullback-Leibler (KL) divergence between the distributions $p(x_i, y)$ and $p(x_i)p(y)$:

$$\begin{aligned} MI(x_i, y) &= KL(p(x_i, y) \parallel p(x_i)p(y)) \\ &= \sum_{x_i \in \{0,1\}} \sum_{y=0}^{7} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}. \end{aligned}$$

The value of $MI(x_i, y)$ serves as a score, with larger values indicating that a feature x_i is strongly correlated with the class labels. Based on these scores, the top features are selected for the model. To determine the number of features to include, a sweep is conducted over the number of top-scored features, and the weighted average F1 score is computed using k-fold validation to threshold the desired performance.

Fig. 4 presents the experimental results for both training and testing F1 scores computed with k-fold cross-validation for three algorithms. For the training F1 score, both SVM and Logistic Regression show a monotonic increase as more features are included, reflecting overfitting. Naive Bayes, on the other hand, initially shows a sharp improvement as the number of features increases when the feature size is small but begins to degrade when too many features are included.

This is because Naive Bayes assigns equal weights to the probabilities of all features and multiplies them, causing the influence of indicative features to diminish as more features are added, ultimately reducing performance. A similar but more pronounced behavior is observed for the testing F1 score of Naive Bayes. Furthermore, Naive Bayes assumes that features are independent, which is often inaccurate. For example, if the token "invoice number" is nearby, it is highly likely that "invoice date" and other header tokens are also in the nearby region.

For SVM and Logistic Regression, performance initially improves before plateauing. The turning point of the F1 score, observed at 434 features, is chosen as the threshold for the number of features to include, as adding more features beyond this point does not improve performance.

TABLE I: Average Training and Testing Accuracy

Model	Training Error (%)	Testing Error (%)
Naive Bayes (9 features)	16.17	16.43
Logistic Regression (249 features)	10.95	14.53
SVM (157 features)	8.81	13.99

D. Overall Performance Evaluation

Table 1 summarizes the best k-fold cross-validation accuracy achieved and training accuracy for the three algorithms, along with the corresponding number of top-scored features. Slight overfitting is observed in Logistic Regression and SVM due to regularization, but overall accuracy is promising. Fig. 6 illustrates the precision, recall, and specificity for all classes and three algorithms on both training and test data, computed using k-fold cross-validation. A trade-off between precision and recall is evident for most classes.

Fig. 6(a) indicates that the models perform well across all classes on the training data. However, a comparison between Fig. 6(a) and Fig. 6(b) reveals high variance for the "Invoice Number" and "PO #" fields, suggesting overfitting. These fields are particularly prone to overfitting due to their highly variable feature sets and the small quantity of PO # tokens in the dataset.

From Fig. 6(a) and (b), it can be observed that Naive Bayes performs worse than Logistic Regression and SVM in terms of both precision and recall for almost all classes. This is likely because Naive Bayes assumes that all features are conditionally independent given the class labels, an assumption that often does not hold in this classification context. Logistic Regression and SVM exhibit similar precision across most fields of interest, except for PO #, Due Date, and Tax, where SVM generally performs slightly better. SVM outperforms Logistic Regression for PO #, whereas Logistic Regression achieves better performance for Due Date and Tax.

E. Overall Performance Evaluation

Table 1 presents the best k-fold cross-validation accuracy achieved and training accuracy for the three algorithms, along with the corresponding number of top-scored features. Slight overfitting is observed in Logistic Regression and SVM due to the use of regularization, but the overall accuracy remains promising. Fig. 6 shows the precision, recall, and specificity for all classes and the three algorithms on training and test data, computed using k-fold cross-validation. A trade-off between precision and recall is evident in most classes. Fig. 6(a) demonstrates that the models perform well across all classes on the training data. However, a comparison between Fig. 6(a) and Fig. 6(b) reveals high variance for the "Invoice Number" and "PO #" fields, suggesting overfitting. These fields are particularly prone to overfitting due to their highly variable feature sets and the limited number of PO # tokens in the dataset.

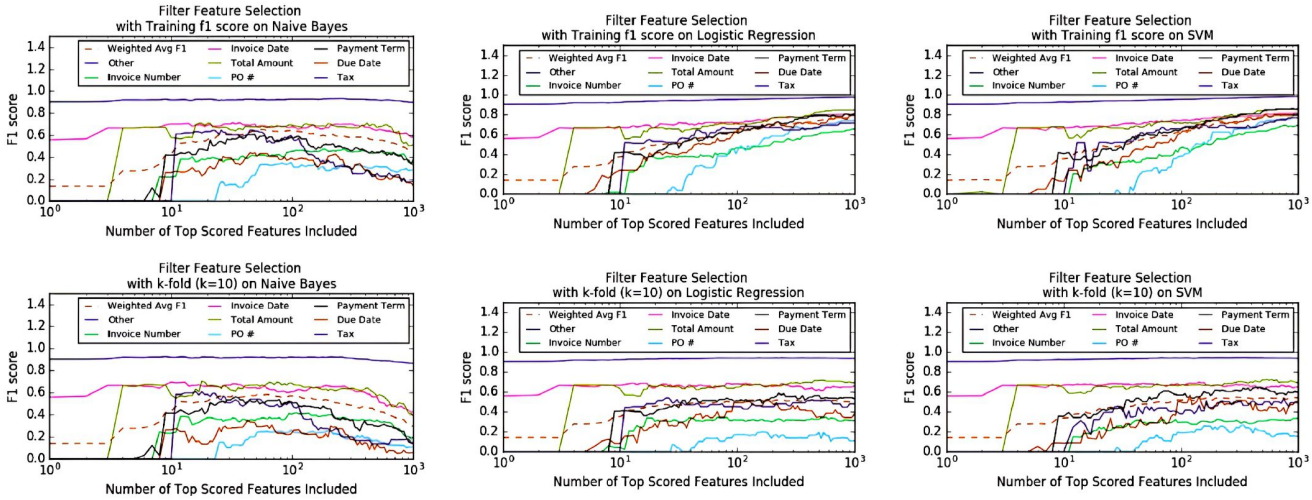

Fig. 5: Filter Feature Selection. Solid lines represent individual F1 scores for all classes, while the dashed line indicates the weighted average F1 score.

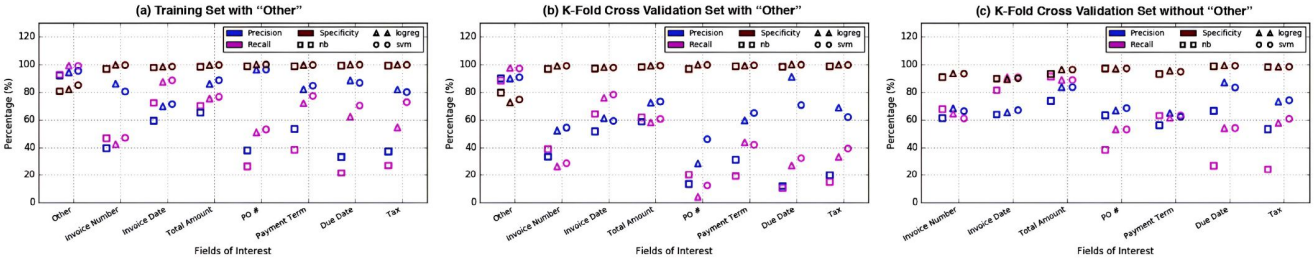

Fig. 6: Classifier Overall Performance, measured by Precision, Recall, and Specificity.

From Fig. 6(a) and Fig. 6(b), it is evident that Naive Bayes generally performs worse than Logistic Regression and SVM in terms of both precision and recall across almost all classes. This underperformance is likely because Naive Bayes assumes that all features are conditionally independent given the class labels, which is an assumption that does not hold in this classification context. Logistic Regression and SVM exhibit similar precision across most fields of interest, except for PO #, Due Date, and Tax, where SVM performs slightly better. Specifically, SVM outperforms Logistic Regression for PO #, while Logistic Regression achieves better results for Due Date and Tax.

The key distinction between these fields is that PO # fields vary significantly in terms of type and position within invoice files, whereas Due Date and Tax are generally consistent in type (date and money, respectively) and can be more easily identified using their self type. This suggests that SVM is more effective than Logistic Regression at extracting the most relevant features for making predictions.

Another notable observation is that, with the exception of the Other field (class label 0), all fields of interest suffer from low recall, regardless of the algorithm used, but exhibit very high specificity. Conversely, the Other field shows very high recall but low specificity. This indicates that the learning algorithms frequently misclassify fields such as Invoice Number or Invoice Date (class labels 1-7) as Other, while rarely misclassifying the Other field as one of the specific fields.

This behavior can be attributed to the fact that the Other field encompasses a diverse range of unlabeled fields in the data, leading to a highly skewed class distribution and a strong tendency for the algorithm to predict a field as Other. This argument is further supported by Fig. 6(c), which shows the same plot without the Other field. Without the Other class, the performance across all fields of interest improves dramatically, especially in terms of precision and recall.

In summary, the algorithms can reliably distinguish between the fields belonging to class labels 1-7 but struggle due to the large volume of tokens labeled as Other. The wide range of feature characteristics exhibited by the Other field leads to a high rate of misclassification for fields 1-7 as 0. I believe that this issue could be mitigated with the availability of more

training data and the inclusion of additional labeled fields of interest in the training dataset.

VI. CONCLUSION AND FUTURE WORK

Overall, SVM produced the best results, as it does not rely on unnecessary assumptions like Naive Bayes and can effectively determine the optimal margin to separate two classes. Given more time, further exploration could be conducted to address overfitting, including expanding the size of the training dataset and employing wrapper model feature selection for more precise results.

Additional analysis could focus on evaluating the effectiveness of the current set of feature selection rules in capturing the inherent structure of invoice layouts. This could include performing rule-level feature selection on a larger pool of potential feature generation rules. Furthermore, the current results are significantly influenced by poor OCR performance, highlighting the need for investigating more optimized OCR tools or obtaining higher-quality invoice images for analysis.

REFERENCES

[1] Y. Zhou and C. L. Tan, "Hough technique for bar charts detection and recognition in document images." vol. 2, 01 2000.

[2] D. Ming, J. Liu, and J. Tian, "Research on chinese financial invoice recognition technology," *Pattern Recognition Letters*, vol. 24, pp. 489–497, 01 2003.

[3] S. Marinai, "Introduction to document analysis and recognition," in *Machine Learning in Document Analysis and Recognition*. Springer, 2008, pp. 1–20.

[4] Y. P. Zhou and C. L. Tan, "Hough technique for bar charts detection and recognition in document images," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2. IEEE, 2000, pp. 605–608.

[5] H. Hamza, Y. Belaid, A. Belaid, and B. Chaudhuri, "Incremental classification of invoice documents," 01 2009, pp. 1 – 4.

[6] Y. Shinyama, "Pdfminer - python pdf parser," 12 2007.

[7] P. Hough, "Method and means for recognizing complex patterns," Patent, 12, 1962.

[8] R. Smith, "An overview of the tesseract ocr engine," vol. 2, 10 2007, pp. 629 – 633.

[9] M. Chapput, "Python implementation of porter2 stemming algorithm," 2010.

[10] E. Loper and S. Bird, "Nltk: The natural language toolkit," 2002, uRL: http://arxiv.org/abs/cs/0205028.

[11] G. Louppe, "Scikit-learn: Machine learning in the python ecosystem," 12 2013.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: a library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 08 2008.

[13] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 07 2007.

[14] D. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 10 2020.

[15] D.-R. Chen, Q. Wu, Y. Ying, and D.-X. Zhou, "Support vector machine soft margin classifiers: Error analysis." *Journal of Machine Learning Research*, vol. 5, pp. 1143–1175, 09 2004.

[16] Y. Bela and A. Bela, "Morphological tagging approach in document analysis of invoices," *Pattern Recognition, International Conference on*, vol. 1, pp. 469–472, 08 2004.