Unstructured Document Recognition on Business Invoice

CS229: Machine Learning

Wenshun Liu, Billy Wan, Yaqi Zhang
Email: wl88@stanford.edu, xwan@stanford.edu, yaqiz@stanford.edu

***Abstract*—This project describes a bag-of-words approach for business invoice recognition. Bags of potential features are generated to capture layout and textual properties for each field of interest, and weighted to reveal key factors that identify a field. Feature selection, threshold tuning, and model comparison are evaluated. Overall, we achieved 8.81% for training error and 13.99% for testing error.**

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 vendor, 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 this project proposes intends to yield additional insights to this problem. 8 fields of interests (including a negative class) are identified and recognized under the process outlined in Fig. 1.

The raw inputs are scanned invoice images. After image processing, OCR, and pattern matching steps, a list of word groups (tokens) and coordinates are extracted from the original images, and are used as the actual input for the model. Bags of potential features are generated from the actual inputs under a set of feature selection rules to capture various layout properties and word patterns for each field, and then weighted using 3 classification models (Naive Bayes, Logistic Regression, and SVM) to output a predicted field for every word group.

II. Related Work

Numerous image processing and machine learning attempts have been made to tackle the invoice recognition problem from different angles.

Image processing approaches rely on column detection and word sequence recognition within each logically segmented region [2], with the occasional aid of machine learning techniques for more precise region classification results. While image processing techniques [3] can be of great aid to many other models, a recognition algorithm centered around it overly simplifies the complex nature of invoice layout, and assumes homogeneous properties in region segmentation based on linear combination of rules, which is almost never the case in practice given the unpredictable nature of invoice layout.

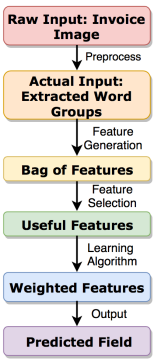

Fig. 1. Flow of Invoice Recognition

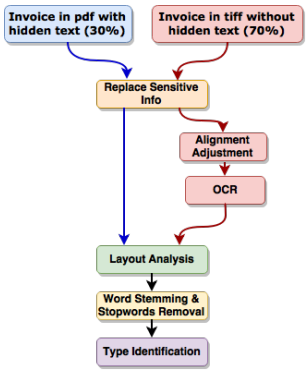

Fig. 2. Flow of Data Preprocessing

In need of more complex models leveraging machine learning techniques, template based classification algorithms are proposed, where the template of an invoice (calculated and represented by a set of layout attributes) is either matched against a template library [1] [4], or is assigned to a cluster of templates sharing similar properties [5]. In either approach, the template library or cluster constantly expands when no obvious match exists.

Template based models have the obvious benefit of being able to recognize the entire invoice all at once through pre-established template-specific rules, and perform the best with high quality images and highly distinctive templates. Unfortunately, neither is guaranteed in reality as invoices are often poorly scanned, and the huge vendor (and thus invoice template) pool implies frequent occurrence of invoices with similar structure but minor (yet crucial) variance in layout and field arrangements. In the case of template library, this could cause critical fields to be mis-recognized following incorrect rules. In the case of clusters, defining template "distance" and distinguishing minor variance within a cluster are themselves tricky and error-prone. Such models are also memory-intensive as the library or cluster size constantly expands.

Also available are rule-based models, where sets of hand-crafted rules are weighted to capture micro-level details for each field [6]. Such approaches avoid the inflexibility when an invoice is treated as a whole. While hand-crafted rules work

well with invoices containing industry standard components and layout, they imply presumptions on field properties that are either incorrectly arbitrary (e.g., amounts are not always right aligned) or simply not achievable (e.g., match price and quantity with amount to identify invoice lines while not all three fields are present in many real-world invoices).

The model proposed in this project also performs field-by-field recognition, though instead of merely determining the weights of hand-crafted rules, a bag of potential features is supplied to generate the best set of rules that should be used.

III. Dataset and Features

A. Dataset Generation

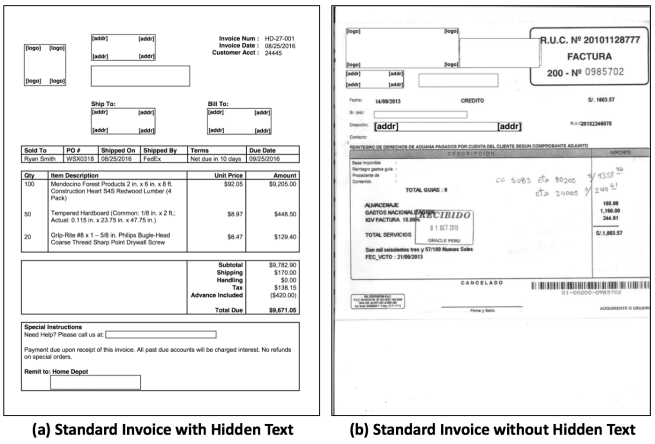

Fig. 3. Sample Invoices

97 raw invoice images are obtained from the internal testing library of Oracle Corporation, examples of which are displayed in Fig. 3. Sensitive information has been manually replaced with tokens (LOGO, ADDR, etc) labeled on the corners of the bounding boxes, to preserve position without revealing the actual text. While some are well-formatted PDF files with hidden text, most are TIFF images that require additional steps before PDF Layout Analysis [7] can take place to extract word groups. The process of generating word groups and coordinates as actual training input is outlined in Fig. 2

TIFF images are first rotated to best alignment with rotation angle calculated from Hough Line Transform [8], and then performed Optical Character Recognition (OCR) [9] to obtain any textual groups exist in the image. All sample invoices are then sent to PDF Layout Analysis, where coordinates of tokenized textual groups are retrieved and stored. Then, additional word processing is implemented on each token, including Porter2 word stemming [10], stopwords removal [11], and type identification using regular expressions. 5 special types - DATE, MONEY, NUMBER, TELE, and EMAIL - are defined for the last process, where the exact textual values are substituted for more generic representation.

B. Feature Generation

For each token, the following set of feature selection rules is applied: horizontally aligned tokens, vertically aligned tokens, nearby tokens within a distance threshold, overall vertical position, and its own type. The resulting features from each token are then accumulated to form the bag of potential features, where binary values are used to track whether any given feature exists for a token. For instance, if token A is horizontally aligned with token B, feature B_halign will be generated and be of value 1 for A and 0 otherwise. The feature selection rules are picked to capture basic layout information, without presumption of any standard template.

Around 8000 features were generated for the 2095 tokens ($m = 2095$) in 97 invoices, and further reduced through preliminary feature pruning (Section III-A) to exclude the ones that only appear once (and are thus not indicative). The remaining features ($n \approx 2000$) are then put into careful feature selection process detailed in Section V.

IV. Methods

Using the scikit-learn library [12], three learning models were trained and used to make predictions: Multinomial Naive Bayes, Logistic Regression, and Support Vector Machines (SVM). In all three models, the class labels $y \in [0, 7]$, where 0 is the negative class, and 1-7 corresponds to *invoice number*, *invoice date*, *total amount*, *PO #*, *payment terms*, *due date*, and *tax*, respectively.

A. Multinomial Naive Bayes

In the Multinomial Naive Bayes model, we make the assumption that the features x_i are conditionally independent given the class labels y. We used this model with Laplace smoothing to fit parameters $\phi_{y=k} = p(y = k)$, and $\phi_{j,y=k} = p(x_j = 1|y = k)$, where $k \in [0, 7]$, in order to maximize the joint likelihood of the data, given by

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

The maximum likelihood estimation of these parameters are given as follows:

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

After fitting these parameters, to make a prediction on a new example with feature x, we calculate the posterior probability for each class k using

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

and substituting the probabilities with the corresponding parameters. The class with the highest posterior probability will then be our prediction.

B. Logistic Regression

In the Logistic Regression model, we first transform our multiclass classification problem into a binary classification problem using the one-vs-rest approach, i.e. when calculating the probability for class k, all other classes have the same label. In binary Logistic Regression, we use the following hypothesis to make predictions:

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

This hypothesis indicates that a logistic/sigmoid curve, which smoothly increases from 0 to 1, is fit to the data such that we predict 1 when $h_\theta(x) > 0.5$ and 0 otherwise. Then, we choose the logistic loss

$$\begin{aligned}\mathrm{L}(z,y) &= \log(1+\exp(-yz)) \\ &= \log(1+\exp(-y\theta^T x)),\end{aligned}$$

where $z = \theta^T x$. The loss is minimized when we have a large margin yz, and maximized otherwise. ℓ_2-regularization is used to combat overfitting. We used a slight variation of empirical regularized risk function than the one from class note to minimize:

$$J_C(\theta) = \frac{C}{m}\sum_{i=1}^{m}\mathrm{L}(\theta^T x^{(i)}, y^{(i)}) + \frac{1}{2}\|\theta\|_2^2 \quad (1)$$

$$= \frac{C}{m}\sum_{i=1}^{m}\log(1+\exp(-y^{(i)}\theta^T x^{(i)})) + \frac{1}{2}\|\theta\|_2^2. \quad (2)$$

The scikit-learn implementation fits the parameter θ by solving the dual optimization problem of L2-Regularized Logistic Regression using coordinate descent [13].

C. SVM

As in Logistic Regression, we first use the one-vs-rest approach to transform our problem into a binary classification problem with labels $\{-1, 1\}$ for all classes. Then, we choose the margin-based loss function

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

where $z = \theta^T x$. This loss function is zero as long as the margin yz is greater than 1 (the model makes the correct prediction and is reasonably confident). In order to fit the model, we try to minimize the empirical ℓ_2-regularized risk, given by

$$J_C(\theta) = \frac{C}{m}\sum_{i=1}^{m}\mathrm{L}(\theta^T x^{(i)}, y^{(i)}) + \frac{1}{2}\|\theta\|_2^2 \quad (3)$$

$$= \frac{C}{m}\sum_{i=1}^{m}[1-y^{(i)}\theta^T x^{(i)}]_+ + \frac{1}{2}\|\theta\|_2^2 \quad (4)$$

The scikit-learn LinearSVC's implementation of SVM uses a linear kernel K. Based on the represented theorem, we can implicitly represent $z = \theta^T x$ as $\sum_{i=1}^{m}\alpha_i x^{(i)T} x^{(i)}$. This allows us to use the kernel trick, and rewrite the empirical regularized risk in terms of α:

$$J_C(\alpha) = \frac{C}{m}\sum_{i=1}^{m}[1-y^{(i)}K^{(i)T}\alpha]_+ + \frac{1}{2}\alpha^T K\alpha, \quad (5)$$

where $K^{(i)}$ is the i^{th} column of the Gram matrix of kernel K. The LinearSVC implementation fits the parameter α of the model by solving the dual optimization problem of the C-Support Vector Classification formulation of SVM [14]. After fitting the parameters, the model then makes predictions based on the value of $z = \theta^T x$.

V. Results and Discussion

A. Metrics

Given the fact that in any invoice, there are far fewer fields that are actually classes 1-7 than fields that belong to Other, our data is heavily skewed towards class 0. Thus, the accuracy (percentage error) of the overall model is not the most informative metric, as a good classification on the dominant class would result in a high accuracy irrespective to performance on the minority classes. Therefore, we computed the precision, recall, specificity, and the corresponding F_1 scores for all classes. F_1 score is the harmonic mean of precision and recall [15], which allows us to combine precision and recall, which are trade-offs of each other, into one metric.

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}, F_1 = 2\cdot\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$$

Sometimes, however, it is useful to have a single metric reflecting how well we perform on the minority classes (all classes except "Other") in the trade-off space. Therefore, we used another metric - the average of F_1 scores of all classes weighted by the frequency of each class label. To exclude the contribution of the majority class, we excluded the "Other" class when calculating the weighted average F_1 score.

B. Regularization Parameter Scaling

Regularization term is added to Logistic Regression and SVM in Eq.(1-5) to reduce overfitting, where C is the regularization parameter that controls the cost of misclassification on the training data [16]. Fig.4 shows the scaling of C with respect to F_1 score for both ℓ_1- and ℓ_2-regularization.

We can clearly see that the results are less desirable when C is either too small or too large. When C is too small, the cost of misclassification is low on the training data, resulting a overly "soft" hyperplane margin that risks underfitting. When C is too large, the algorithm is forced to explain the input data stricter as the penalty for non-separable points is high, resulting in a model that overfits.

Although ℓ_1-regularization is computationally more efficient for sparse data, from Fig.4 we see that ℓ_2-regularization in general works better in optimizing the average F_1 score we are interested in. Thus, for our model, ℓ_2-regularization is applied to both Logistic Regionssion and SVM, with $C = 10.72$ for Logistic Regression and $C = 0.58$ for SVM.

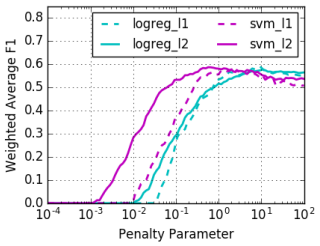

Fig. 4. Regularization

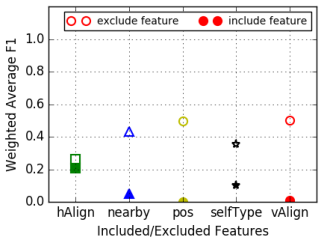

Fig. 5. Feature Rules Effectiveness

C. Feature Selection

Fig.5 evaluates the effectiveness of each feature selection rule, where solid markers represent weighted average F_1 when only that selection rule is included (inclusion set), while hollow markers represent the case when only that rule is excluded (exclusion set). We can easily see that horizontal alignment (hAlign) is the most effective, with best performance in the inclusion set and worst performance in the exclusion set, and is followed by self type (selfTpe), nearby, vertical align (vAlign), and position (pos). It is not a surprise that hAlign is the most effective, as many fields of interests, regardless of the absolute position, are horizontally aligned in a decent number of invoices. On the contrary, while the absolute vertical position was moderately indicative when data size was small, The variety in invoice templates increase dramatically as data set expands, making it harder to derive common properties in layout structure and causing this feature selection rule to be less instructive.

Following rule-level analysis, filter feature selection is performed on the individual features to exclude the less indicative ones generated by effective selection rules. First, for every feature, we use the empirical distribution of each feature, $p(x_i)$, and that of each class, $p(y)$, as well as their joint distribution, $p(x_i, y)$, to compute the mutual information $\mathrm{MI}(x_i, y)$ between x_i and y, given by the Kullback-Leibler (KL) divergence of the distributions $p(x_i, y)$ and $p(x_i)p(y)$, as

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

$\mathrm{MI}(x_i, y)$ thus serves as a score, a large value of which indicates the feature x_i is strongly correlated with the class labels and small otherwise. Therefore, we pick top scored features in our model. To decide how many features to include, we sweep the number of top scored features included and use the weighted average F_1 score with k-fold validation to threshold desired performance.

Fig. 6 shows our experimental results for both training F_1 score and testing F_1 score computed with k-fold cross-validation for three algorithms. For training F_1 score, both SVM and Logistic Regression monotonically increase as more features are included because the model is overfit. The performance of Naive Bayes, however, first dramatically improves as number of features increases when feature size is small and then starts to degrade when too many features are included. This is because Naive Bayes weights probabilities of all features equally and simply multiply them together. Therefore, the contribution of indicative features to the overall probability decreases as more features are included, resulting in a downgrade in performance. Similar but more exaggerated behavior can be observed in testing F_1 score for Naive Bayes. Additionally, Naive Bayes assumes features are independent to each other, which in many cases is not accurate. For instance, if token 'invoice number' is in nearby region, it is very likely that 'invoice date' and other header tokens are also in the nearby region. Finally, performance of SVM and Logistic Regression increases first before reaching constant. The turning point of $F1$ score at 434 number of features is where we threshold amount of features to include, as including more features will not improve the performance.

TABLE I
AVERAGE TRAINING AND TESTING ACCURACY

	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 I shows the best k-fold cross validation accuracy achieved and training accuracy for three algorithms with corresponding number of top scored features. There is only slight overfitting in Logistic Regression and SVM due to application of regularization and overall accuracy is promising.

Fig. 7 shows the precision, recall, and specificity of all classes and three algorithms on training and test data computed with k-fold cross validation. The trade-off between precision and recall can be found in most classes. Fig. 7(a) shows that our models performs well across all classes on the training data. However, comparison between 7(a) and (b) shows that our algorithms have a high variance for Invoice Number and PO #, which suggests overfitting. These fields are especially prone to overfitting because their bag of features vary hugely and we have a especially small quantity of PO # tokens.

It can be seen from Fig. 7(a) & (b) that for almost all classes, Naive Bayes has a worse performance than Logistic Regression and SVM in terms of both precision and recall. We believe that this is largely due to the aforementioned fact that the Naive Bayes model makes the assumption that all features are conditionally independent given the class labels, which likely does not hold in this classification context. Logistic Regression and SVM have similar precision in all fields of interest but *PO #*, *Due Date*, and *Tax*, with SVM performing slightly better. SVM outperforms Logistic Regression for *PO #*, whereas Logistic Regression has better performance for *Due Date* and *Tax*. One major difference between these fields is that

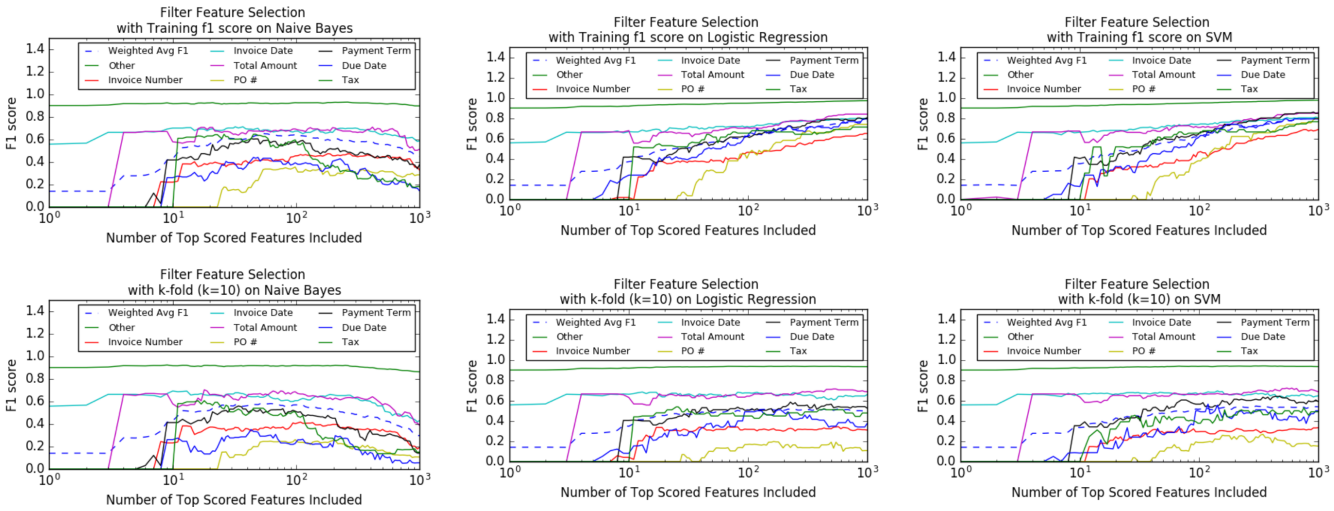

Fig. 6. Filter Feature Selection. The solid lines are individual F_1 scores for all classes and the dashed line is the weighted average F_1 score

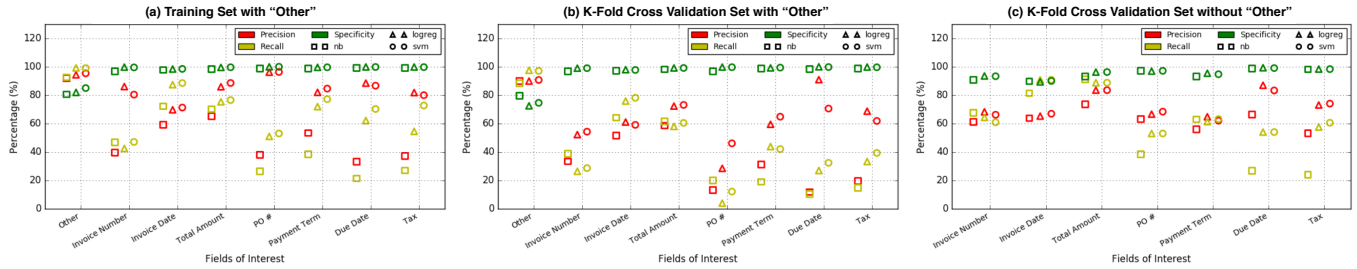

Fig. 7. Classifier Overall Performance, measured by Precision, Recall, and Specificity

fields that are *PO #*'s can vary largely in terms of its type and its position in the invoice file, whereas *Due Date* and *Tax* are generally of the same type (date and money, respectively), and can be identified by self type more easily. This suggests that SVM is better than Logistic Regression at extracting the most useful features among all when making a prediction.

Another important observation is that except *Other* (class label 0), all fields of interest suffer from low recall regardless of which algorithm is used, but have very high specificity, whereas the *Other* field has very high recall but low specificity. This indicates that it is very easy for our learning algorithms to mis-classify fields like *Invoice Number*, *Invoice Date*, etc. (class labels 1-7) as *Other*, while there are very few instances where *Other* is classified as the rest of the fields. The reason is that the *Other* field actually contains all different type of fields that are unlabeled in our data, which results in a highly skewed class distribution and leads to a very high tendency for our algorithm to predict a field to be *Other*. This argument can be further corroborated by Fig. 7(c), which is the same plot but without the *Other* field. It can easily be seen that the performance across all fields of interest dramatically increases, especially for precision and recall. In other words, our algorithm can reliably tell whether a field belongs to one class or another among class labels 1-7, but because of the presence of large amount of tokens belonging to *Other* and *Other* exhibits large range of feature characteristics, our performance suffers from the high rate of misclassifying 1-7 as 0. We believe the issue can be alleviated when more training data are available and more type of fields of interest are labeled in our training data.

VI. Conclusion and Future Work

Overall, SVM produced the best results because it does not make unnecessary assumptions like Naive Bayes does, and can intelligently find the best margin separating two classes.

If we had more time, we would explore more ways to handle overfitting, which includes expanding training data size, and using wrapper model feature selection for more precise results. Further analysis might be worthwhile on the effectiveness of the current set of feature selection rules in capturing the inherent nature of invoice layout, and perform rule-level feature selection on a larger pool of potential feature generation rules. Additionally, current results are largely affected by poor OCR performance, which suggests investigation on more optimized OCR tools or collection of higher quality invoice images.

REFERENCES

[1] D. Ming, J. Liu, and J. Tian, "Research on chinese financial invoice recognition technology," *Pattern recognition letters*, vol. 24, no. 1, pp. 489–497, 2003.

[2] S. Marinai, "Introduction to document analysis and recognition," in *Machine learning in document analysis and recognition*, pp. 1–20, Springer, 2008.

[3] 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, pp. 605–608, IEEE, 2000.

[4] E. Sorio, "Machine learning techniques for document processing and web security," 2013.

[5] H. Hamza, Y. Belaïd, A. Belaïd, and B. B. Chaudhuri, "Incremental classification of invoice documents," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[6] Y. Belaïd and A. Belaïd, "Morphological tagging approach in document analysis of invoices," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 1, pp. 469–472, IEEE, 2004.

[7] Y. Shinyama, "Pdfminer: Python pdf parser and analyzer," 2010.

[8] H. P. VC, "Method and means for recognizing complex patterns," Dec. 18 1962. US Patent 3,069,654.

[9] R. Smith, "An overview of the tesseract ocr engine," 2007.

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

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

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

[13] 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, no. Aug, pp. 1871–1874, 2008.

[14] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[15] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[16] 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, no. Sep, pp. 1143–1175, 2004.