Learning Structured Information from Small Datasets of Heterogeneous Unstructured Multipage Invoices

David Emmanuel Katz[1], Christophe Guyeux[2],
Ariel Haimovici[1], Bastian Silva[1],
Lionel Chamorro[1], Raul Barriga Rubio [1] and
Mahuna Akplogan[1]
[1] smartlayers.io, [2] Université de Bourgogne Franche-Comté, France

Abstract

We propose an end to end approach using graph construction and semantic representation learning to solve the problem of structured information extraction from heterogeneous, semi-structured, and high noise human readable documents. Our system first converts PDF documents into single connected graphs where we represent each token on the page as a node, with vertices consisting of the inverse euclidean distances between tokens. Token, lines, and individual character nodes are augmented with dense text model vectors. We then proceed to represent each node as a vector using a tailored GraphSAGE algorithm that is then used downstream by a simple feedforward network. Using our approach, we achieve state-of-the-art methods when benchmarked against our dataset of 205 PDF invoices. Along with generally published metrics, we introduce a highly punitive yet application specific informative metric that we use to further measure the performance of our model.

1 Introduction

The problem of structured information extraction from human readable formats to machine readable formats has been an active area of research for both academia and industry due to economies of scale that automated technologies of data extraction provide corporations with. This information extraction is of interest to many disciplines, which see it as a decision support tool and a way to save money. For example, the medical information services of hospitals in most countries in the world must, from the patient record, extract the medical procedures performed and encode them according to the ICD10 nomenclature, so that the social security can reimburse the hospital on a fee-for-service basis.

While automatic information retrieval from scanned documents is mature to a certain extent, and there have been a number of success stories, there is substantial room for further application specific innovations. Some documents are more difficult to process, and some applications have specific needs that are difficult to satisfy. For example, the sensitive nature of corporate financial information makes solutions without high levels of accuracy at a target level entity unfeasible.

First of all, each company has its own template, and therefore, in the production phase, we have new templates not contained in the learning set. Moreover, the vast majority of companies interested in automatic information recognition from invoices are small, and can only manually annotate a few dozen to a few hundred invoices. In other words, the learning set is very small, and therefore unsuitable for traditional large scale deep learning techniques. Finally, some fields of these invoices are sensitive and require perfect recognition, such as the invoice amount or the tax percentage. The simple addition or deletion of a digit in these numbers has a large impact in these final results.

This is why we wanted to place ourselves in the following specific context, which has not been looked at much in the literature: small knowledge base, not representative of the variety of templates, and high accuracy required. Considering this specificity, the main contribution of this article is twofold. On the one hand, a system is proposed that achieves competitive results using a small amount of data compared to the state-of-the-art systems that need to be trained on large datasets, that are costly and impractical to produce in real-world applications. On the other hand, we propose to report the Levenshtein ratio [13] of predicted entities and annotated entities, as a good $f1-$score is not a guarantee of a correct recovered target entity such as a VAT or Invoice Number.

The remainder of this article is organized as follows. State of the art methods for invoice extraction are presented in the next section. Our proposal is explained in details in Section 3. An experimental evaluation is provided in Section 4, and obtained results are discussed too. This research work ends by a conclusion section, in which the contribution is summarized and intended future work is outlined.

2 State of the art

The extraction of information from documents like invoices has long been seen as a task of sequence labeling: after extraction with a OCR tool of the whole text from documents, each obtained token receives a label that follows a rule-based approach for old methods. These rules were defined by humans and based on trigger words, regular expressions, or even linguistic descriptors (e.g., amounts are in the neighborhood of “total” or “VAT”). Such rule-based approaches are used, for instance, in [10] and [6]. Then, as in a lot of fields of research, rule-based methods have been supplanted by machine learning ones [9], like random forests [19] or support vector machines [21]. During the most recent years (2020's) deep learning methodologies have surpassed other traditional machine learning techniques, as the experimental evidence of [14] provides.

Among these machine learning based approaches, a collection of tools are available that share the same hypothesis, which is: all possible templates are available in the training set. This is the case for instance in [8], in which a database of field positions is required for each template. This is the case too for [15], where all (field, pattern, parser) triplets must be manually provided for each template, while other approaches of this kind can be found in, e.g., [6] and [4] for a specific invoice framework. Such an hypothesis is however problematic in our context, as in practice, every new customer comes with its own particular form of invoice, which most of the time presents a new template, different from anything already known.

In recent years, deep learning methods have outperformed the more classical ones, both in terms of precision and recall. The most known deep learning based approaches are listed hereafter. Note that these methods frequently need a huge training dataset, which can only be produced by big companies in invoices context.

CloudScan is a commercial software proposed by Tradeshift [17]. It is based on a recurrent neural network that has been trained to recognize 8 fields on a corpus of 300k invoices. A good point is that this platform requires no configuration and does not rely on statistical models. However, half of the 8 fields presented an $f1-score$ lower than 0.87 (e.g., 0.76 for order ID), while to be useful in the invoice information extraction context, a larger score is required. And while invoices are often written in both vertical and horizontal directions, they only choose to apply a left-to-right order. This problem of not considering the spatial information into the key information extraction process is circumvented by CUTIE [23], which stands for Convolutional Universal Text Information Extractor. This is a model based on spatial and contextual information provided to a convolutional neural network coupled with a word embedding layer. Obtained results are quite good, but it presuppose to be able to extract a gridded version of the text, which is problematic in the invoice case. Furthermore, as is a shared theme across invoice extraction models such as [14], its learning stage needs thousands of labeled documents.

Finally, and to the best of our knowledge, the only article that faces the same constraints than us about the dataset size is [11]. They propose two methods which are based on neural networks, and focus on the trade-off between data requirements and performance in the extraction of information. The first method consist to adapt Named Entity Recognition systems for fields extraction of invoices, by fine-tuning BERT [7] to this specific task. The second "class-based" method adapted the features of CloudScan and proposed some extra features to automatically extracts the features, with no preprocessing step nor dictionary lookup. However, the $f1-$scores they obtain are always close to 0.80, which are far from being useful in practice. Furthermore, the smallest set they consider for learning is still huge (20k), which is very far from the concrete use context we aim at. Finally, they only use 1D information as their method is

Natural Language Processing oriented, while invoices structure is 2D.

3 Methodology

Our task consists of taking a set of raw PDF documents with every word on every page being labeled, in order to train a statistical model that can infer the label of all of the words on every page of a new unobserved document. Every document can contain one or more tokens, totaling n tokens in a dataset: the label can be UNDEFINED ℓ_u, or a given target entity ℓ^i, where $\forall i \in [1, n]$, ℓ^i is one of the text listed in Table 1. The layout and structure of every document is obviously not unique.

'Invoice Date',	'Invoice Number',	'Client Name',
'Company Tax Number',	'Total Invoice Money',	'Client Address',
'Logo',	'Total Invoice Money w/o Tax',	'Company Address',
'Company Phone Number',	'Payment Conditions',	'Total Tax',
'Due Date',	'Client Account Number',	'Client Tax Number',
'Tax Percentage',	'Company Name',	'Additional Bussines Information',
'Company Bank',	'IBAN/Account Number',	'Company Email',
'Client Contract Number',	'Company Website',	'Route Number',
'Client Delivery Address',	'Order Number',	'Penalites',
'Additional Financial Information',	'Delivery Details',	'AdditionalTaxInformation',
'AdditionalLegalInformation',	'Date Ordered',	'Client Phone Number',
'Company Order Number',	'Client Email',	'Incoterm',
'LineItemsTable',	'Delivery Date',	'Company Contract Number'

Table 1: List of target features for the information extraction process

Our process can be separated into 5 key steps preceded by a preprocessing, namely: Training of graph embedding, Feature Extraction from words, Feature Merger, Feedforward Network, and Word Token Merger, see Figure 1. These steps are described hereafter.

Initialization We first concatenate pages into a single image file for each document. Then, big connected components are deleted, which removes specific lines, mainly in tables. This helps the OCR tool to extract boxes within table cells. Concretely, this deletion is operated as follows. The image is converted in grayscale, and a thresholding is applied to it (by using THRES_BINARY_INV+THRES_OTSU from OpenCV [2]). We extract the objects with OpenCV's findContours, and we take the rectangle circumscribed to the object. If its length or width is less than 0.1* that of the image, we hide the rectangle. This preprocessing allows to remove various elements such as signatures.

Finally, text boxes and their content are recognized thanks to Tesseract v.5.0 [20], leading to a set of $\{(w^i, x_1^i, y_1^i, x_2^i, y_2^i, \ell^i) | i = 1..n\}$, where for all i, w^i corresponds to the string of characters inside the bounding box i, x_1^i, x_2^i, y_1^i, and y_2^i correspond to the Cartesian coordinates of the corners of this box, and ℓ^i corresponds to the unique label associated manually to the string character in box i. An example of such tuples: ("9,270.00 EUR", 100, 100, 200, 200, "INVOICE AMOUNT").

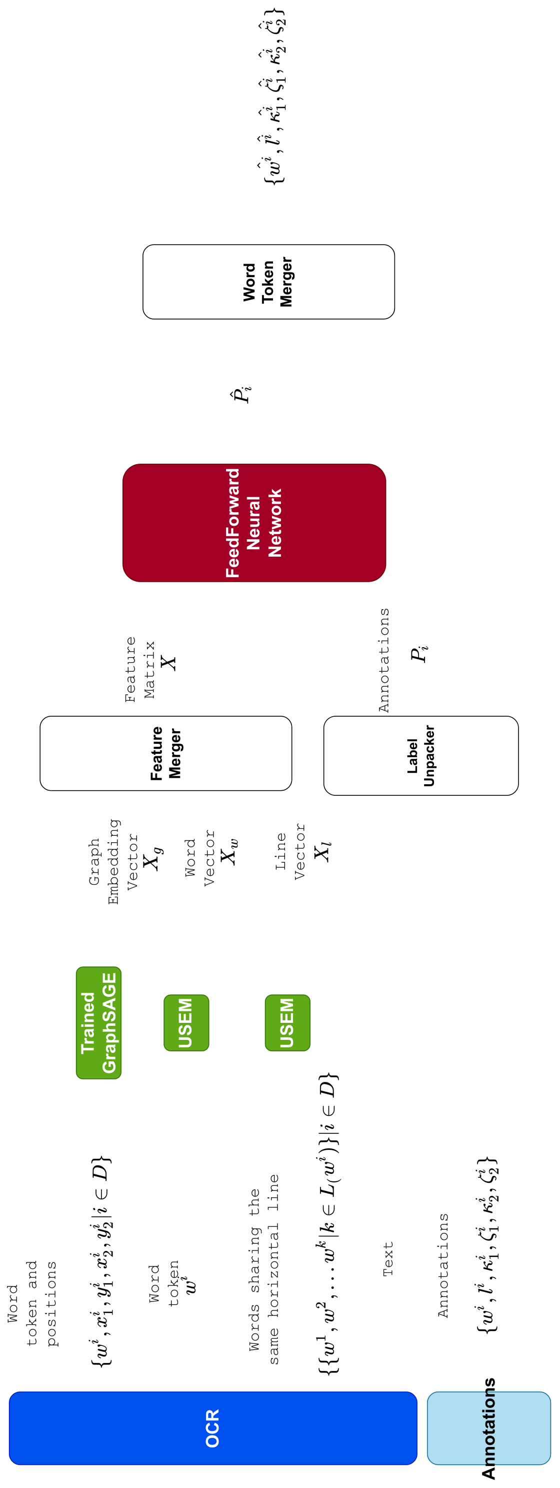
OCR
Annotations
Word token and positions
$\{w^i, x_1^i, y_1^i, x_2^i, y_2^i \mid i \in D\}$
Word token w^i
Words sharing the same horizontal line
$\{\{w^1, w^2, \dots w^k \mid k \in L_(w^i)\} \mid i \in D\}$
Text
Annotations
$\{w^i, l^i, \kappa_1^i, \zeta_1^i, \kappa_2^i, \zeta_2^i\}$
Trained GraphSAGE
USEM
USEM
Graph Embedding Vector X_g
Word Vector X_w
Line Vector X_l
Feature Merger
Label Unpacker
Feature Matrix X
Annotations P_i
FeedForward Neural Network
$\hat{P}_i$
Word Token Merger
$\{\hat{w^i}, \hat{l^i}, \hat{\kappa_1^i}, \hat{\zeta_1^i}, \hat{\kappa_2^i}, \hat{\zeta_2^i}\}$

Figure 1: Main pipeline

Training of graph embedding Using the OCR output of Tesseract, we construct a weighted undirected graph for every document in our dataset. The graph nodes are the ocr tokens and the edges are such that two boxes are horizontally connected if they are contiguous in the same line, and they are vertically connected if they have any overlap in the horizontal axis and have no other box vertically between them. The edges are weighted proportional to the inverse of the Euclidian distance between box centers, as depicted in Figure 2. An additional heuristic to decrease memory footprint is that we do not draw edges between tokens that are separated by 5 horizontal lines.

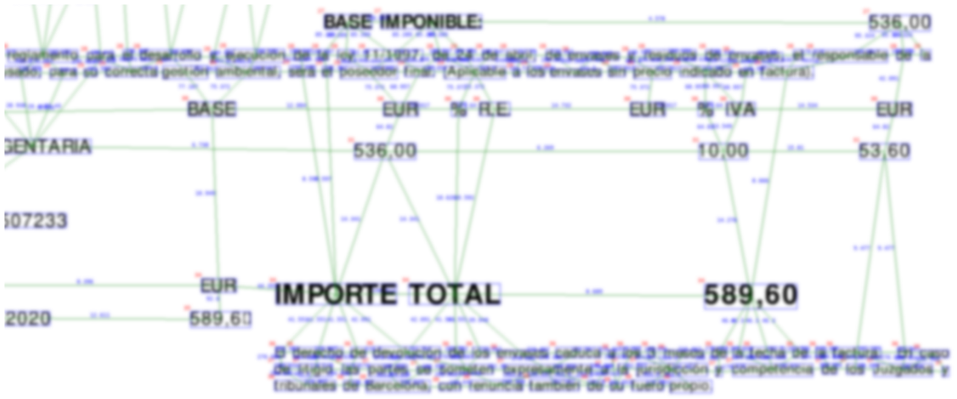

Figure 2: Weighted undirected graph from a given piece of invoice

The individual word cannot be included as an attribute of each vertex in the GraphSAGE [12] step to come, so instead a list of 11 Boolean statistics are produced for each word vertex, cf. Table 2. And words within each node are replaced by these Boolean attributes. The graph construction and graph training process actually does not only take in the 11 Boolean features, but they take in also 3 additional numerical features (to parameterize the nodes in the graph): One multishot vector with the characters of the token (this what we call "Simple Character Encoding"), a one hot vector of the token within the all of the tokens in the corpus (any unseen token is labelled as Undefined), and finally a single TF-IDF score of the token with respect to all the tokens in the document and the dataset.

Finally, an untrained GraphSAGE model is used by training it on the graph structures described above, this model being selected for its ability to encode vertices with properties (here, the Boolean statistics). A fully

is_alpha	is_digit	like_email
like_num	like_url	is_lower
is_punct	is_quote	is_space
is_title	is_upper	

Table 2: The 11 Boolean statistics used in token encoding

trained GraphSAGE model is produced at this stage, fitting an embedding for each node in the graph, the task being the classification of the 39 annotated labels of Table 1, see Figure 3. This step converts each node of the previous graph into a vector X_g of size 1×1024, as illustrated in Figure 4. This 1024 dimension vector systematically represents the bounding box in question, and also captures via the embedding processes samples and aggregations of neighboring nodes.

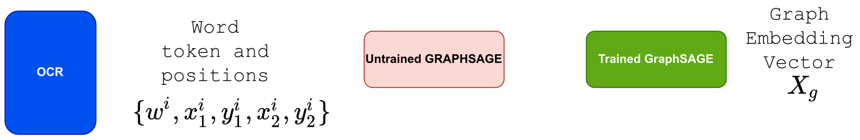

Figure 3: GraphSAGE training pipeline

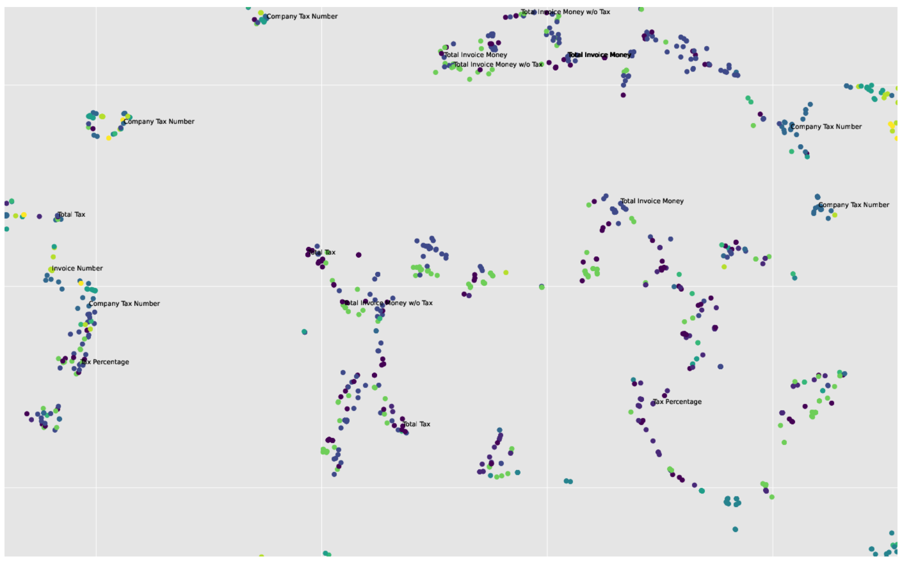

Figure 4: t-SNE [22] reduction of a GraphSAGE embedding, in which inherent structures of the documents are unveiled.

Feature Extraction from words Needed for further downstream processing, we proceed to augment our GraphSAGE nodes by including an embedding of each line and character using USEM [3]. This stage takes the output of the OCR model and aggregates every token with a concatenation of every token that exists on the same horizontal line. An one hot representation at the character level for each token has been applied, while a multihot representation of the tokens in the document has been considered as text vectorizer at the line level.

Both sequences of characters, further referred to as tokens and line tokens (the individual word, and the line) are passed onto a standard Google's Universal Sentence Encoder USEM from TensorFlow [1], to produce two embedding vectors X_w for words, and X_l for lines, both of size 512.

Label unpacking OCR software is not entity aware and will, in an apparently random fashion, split up words or merge them together to create OCR tokens. And due to the potentially poor quality of the documents in question, the OCR model can potentially omit or even add characters that create additional noise. However, these tokens are the only piece of observation we will obtain from unseen data. This is why our annotation process focuses on capturing entire entities that appear on a document. Entities can occupy several words, or even several lines, therefore we need a specific phase that conducts collision detection on the OCR output, to match it with our annotated observations to assign every OCR token i a label ℓ^i.

If a OCR output token is within the area of an annotated token, then the OCR token is assigned the label ℓ^i; if there is no annotated token that surrounds the OCR token, then the OCR token is assigned a label ℓ_u for undefined. After running the label unpacking, we are left with out target vector P with an integer representing each target entity.

Feature Merger With our graph and word embedding vectors created for every word in our document, we proceed to merge our features into a single matrix that can be used for downstream processes: X_g, X_w, and X_l are concatenated to create an individual 1×2048 vector. All tokens are then grouped together to form a matrix X.

Feedforward Network We elected to use a simple feedforward network for the classification task. We train the network using only a subset of 10 target features of particular interest (see Table 3). The network consists of three dense layers of 1024 neurons with activation ReLU plus a dense softmax layer with 10 neurons, while the GCAdam optimizer has been chosen. Due to extreme unbalance of target entities with undefined in our dataset we use sparse categorical cross entropy cost function and $f1-$score for evaluation.

Word Token Merger Due to the nature of our use case, our individual OCR tokens are not suitable for direct consumption by downstream processes.

'Company Name',	'Invoice Number',
'Invoice Date',	'Company Tax Number',
'Due Date',	'Tax Percentage',
'Total Invoice Money',	'Total Invoice Money w/o Tax',
'Payment Conditions',	'Total Tax'

Table 3: List of 10 target features used in the classification task

Our pipeline must take care of assuring end users that the exact sequence of characters (including newlines and spaces) are extracted from a document and labeled with a specific target entity e, hence our training – and most importantly evaluation – pipeline will score target entities at an annotated level, requiring that individual OCR tokens be merged together following a heuristic that depends on a distance metric. Tokens are merged horizontally and/or vertically if their bounding boxes are within ϵ distance from each other and whose labels ℓ match. The word token merger finally produces merged word tokens with next bounding boxes.

4 Experiments

4.1 Experimental protocol

Our invoice dataset consists of individual images for each page of every invoice of our 205 total private invoices. Invoices where assigned a human labeller to draw bounding boxes on every occurrence of a target entity on the page. A second human labeller was used to review the work of the first one. The target entities where one of the labels listed in Table 1. The dataset contained one third of documents in spanish, english and french respectively. It has been split into 70% for training and 30% for testing.

In order to understand the performance gain in introducing graph embedding, we present two baseline methods that are trained directly on OCR tokens and labels, and we compare them to the proposed method. These baseline models consist of a ridge classifier on the one hand, and a neural network on the other hand. They have been trained on the TF-IDF [18] vector of the tokens and the bounding boxes. At each time, the $f1-$score is computed, and it is compared to the results obtained by CloudScan [16] and by Hamdi et al. [11].

These two tools have been chosen, because CloudScan is a very recent and well-known professional deep learning based software that achieves state-of-the-art results, and because [11] is the study the closest of our particular context. Furthermore, they both provide $f1-$scores, or precision and recall. We do not provide further comparison, because these two tools have the highest scores in the literature. And as it has been evoked before, existing work are not directly applicable to our specific context. Finally, document datasets are always private in this area of research, which makes reproducibility and comparison at least questionable in practice.

Name	precision	recall	f1-score
Total Tax	0.00	0.00	0.00
Invoice Number	1.00	0.05	0.10
Invoice Date	0.00	0.00	0.00
Total Invoice Money	0.00	0.00	0.00
Tax Percentage	1.00	0.05	0.10
Company Name	0.28	0.33	0.30
Total Invoice Money w/o Tax	0.00	0.00	0.00
Payment Conditions	0.90	0.15	0.26
Due Date	0.00	0.00	0.00
Company Tax Number	0.68	0.13	0.22
UNDEFINED	0.79	0.98	0.87

Table 4: Baseline 1: Linear Ridge Classifier

Name	precision	recall	f1-score
Total Tax	0.23	0.16	0.19
Invoice Number	0.88	0.35	0.50
Invoice Date	0.67	0.49	0.57
Total Invoice Money	0.40	0.31	0.35
Tax Percentage	0.71	0.77	0.74
Company Name	0.48	0.49	0.48
Total Invoice Money w/o Tax	0.30	0.14	0.19
Payment Conditions	0.87	0.23	0.36
Due Date	0.43	0.55	0.48
Company Tax Number	0.63	0.25	0.36
UNDEFINED	0.87	0.91	0.89

Table 5: Baseline 2: Neural Model

Note finally that our implementation uses Tesseract V, StellarGraph's GraphSAGE implementation [5], and Tensorflow 2.4.3. And we have trained our model in Google cloud platform using a TESLA V100 GPU.

4.2 Obtained results

In what follows, we present the classification statistics and sequence scoring statistics, and some observations and conclusions from them.

Obtained results with the two baselines are provided in Table 4 for the linear ridge classifier, and in Table 5 for the neural network. Results from CloudScan are reproduced in Table 6, while scores obtained by Hamdi et al. can be found in Table 7. Finally, our own scores are provided in Table 8.

Field	LSTM
Number	0.760
Date	0.774
Currency	0.905
Order ID	0.523
Total	0.896
Line Total	0.880
Tax Total	0.878
Tax Percent	0.869

Table 6: $f1-$scores of CloudScan for unseen templates

Fields	NER-based	class-based
docType	0.87	0.90
docNbr	0.69	0.86
docDate	0.85	0.82
dueDate	0.82	0.84
netAmt	0.51	0.74
taxAmt	0.54	0.77
totAmt	0.51	0.86
currency	0.80	0.89

Table 7: $f1-$scores of the two methods proposed in Hamdi et al. [11]

Name	precision	recall	f1-score
Total Tax	0.93	0.86	0.89
Invoice Number	0.97	1.00	0.98
Invoice Date	0.98	1.00	0.99
Total Invoice Money	0.98	0.97	0.97
Tax Percentage	0.76	1.00	0.86
Company Name	0.98	1.00	0.99
Total Invoice Money w/o Tax	0.97	1.00	0.98
Payment Conditions	0.96	1.00	0.98
Due Date	1.00	0.98	0.99
Company Tax Number	0.87	1.00	0.93
UNDEFINED	1.00	0.72	0.84

Table 8: Our Model: Graph Embeddings + Word/Character Embeddings

From the results it is clear that a linear ridge classifier has the worst statistical results, the two state-of-the-art tools do much better than our two baselines. We can also see that adding the embedding phase to the neural network, which is exactly our proposed method, allows to greatly improve the scores. We can also see that the two state of the art tools produce, in the end, a rather bad classification, with many $f1-$scores either mediocre, or even very low. Finally, our method obtains much better $f1-$scores, proving that our classification can be used in practice. Let us however remark that it is very hard to compare these types of models across different datasets, and it is not totally fair to compare such $f1-$scores, as the quality of the annotations, data, nature of labels, and many other minor details can have a major impact in predictive power.

To further illustrate this results, we propose the use of a finer metric that exactly computes at which edit distance our results are to the ground truth. Obtained results are summarized in Table 9 for further investigations. The average of document averages Levenshtein scores by field is equal to 0.77, while the full average over the whole corpus is of 0.78. These results start to look good, if we take into consideration that we have customized the Levenshtein ratio as follows: if a token is misclassified, the entire Levenshtein score is set to 0, given that the information is not correctly extracted and has no tangible use to end users. This additional penalization allows us to optimize for industry usability. Concretely, these scores more refined than a simple $f1$ mean that our method is useful in practice, in the specific context of this study, even though further improvements are welcome. Note that due to the very low classification score of the baselines, the final Levenshtein ratio is not reported as it is close to 0. Similarly, we cannot compute this ratio for the two state of the art tools, as we do not have access to their database.

	Levenshtein Ratio
Name	
Total Tax	0.70
Invoice Number	0.95
Invoice Date	0.97
Total Invoice Money	0.82
Tax Percentage	0.68
Company Name	0.81
Total Invoice Money w/o Tax	0.73
Payment Conditions	0.72
Due Date	0.94
Company Tax Number	0.74

Table 9: Levenshtein scores of the proposed method

4.3 Discussion

The main question at the origin of our experiments can be summarized as follows: can the node embedding we propose add predictive power? Intuitively, we can think that it can, because it is always useful to look at what there is, in the document, in the neighborhood of a given token (this is precisely what humans do). The results obtained confirm this intuition, since without this embedding, we fall back on our baselines, which contrary to our method are unable to make a classification. And when we add embedding to a simple neural network, we get better results than the state of the art, even though they use the best deep learning tools. We can summarize this by saying that the embedding tool is more important than the classification tool, at least in the particular context of our study: embedding neighbor token significantly leads to greater predictive and empirical results.

Another point to emphasize is that the process of unpacking and regrouping OCR produced tokens into full target entities is not a trivial task. End users expect high levels of quality from their invoice information extraction model, and the only way to empirically guarantee the performance of the model is by scoring it on its ability to produce the entire sequence of characters. Most research today in the field only reports classification metrics of individual OCR tokens. Using Levenshtein scores allows us to look at entity level extraction capability and not just accuracy of classification.

Note finally that our model can scale from small to large scale due to online learning and parallelization, while there is room for improvement creating lighter and more powerful versions of this pipeline. We have thoroughly tested our embedding method before arriving at our proposal, but we have only detailed here the best embedding, due to lack of space, and because it is not useful to show techniques that did not have adequate performance. Among these techniques investigated were object detection approaches, that failed to produce satisfactory results with our limited amount of data.

5 Conclusion

In this article we have studied information retrieval from multi language invoices restricted by the amount of data that is available at training with an emphasis of generating highly accurate results applicable to real world corporate automation projects. To stress its real-world applicability, we proposed to consider finer metrics based on edit-distances rather than f1 scores on token classes only. We have been able to show that the graph node embedding step is a key driver of increased accuracy in 2d document information extraction.

Proceeding our implementation and experimentation phases we are able to achieve competitive state of the art results in addition to industry specific applicability. Our end-to-end approach has been detailed and we have performed various evaluations confirming the practical interest of our approach.

In our future work, we would first like to measure the impact of the final classifier on the quality of the results, and see if an improvement can be obtained either by tuning the architecture or its hyper parameters. Increased efficiency and work can be conducted in the token merging staging, by running studies on different graph construction methods. We would also like to understand how human-corrected predictions can be fed back into our pipeline to increase accuracy, and what is the average marginal effect of a newly corrected prediction our evaluation metrics. Finally, we will look at other types of documents, to see if what has been done on invoices can easily be extended to other contexts of heterogeneous documents.

Acknowledgement

The authors would like to thank Lila Benhammou founder of Humans4Help and Co-founder of Smart Layers. This study was funded by BPIfrance in the frame of the French Deeptech company.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, 2018.

[4] Francesca Cesarini, Enrico Francesconi, Marco Gori, and Giovanni Soda. Analysis and understanding of multi-class invoices. *Document Analysis and Recognition*, 6(2):102–114, 2003.

[5] CSIRO's Data61. Stellargraph machine learning library. `https://github.com/stellargraph/stellargraph`, 2018.

[6] Andreas R Dengel and Bertin Klein. smartfix: A requirements-driven system for document analysis and understanding. In *International Workshop on Document Analysis Systems*, pages 433–444. Springer, 2002.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[8] Daniel Esser, Daniel Schuster, Klemens Muthmann, Michael Berger, and Alexander Schill. Automatic indexing of scanned documents: a layout-based approach. In Christian Viard-Gaudin and Richard Zanibbi, editors, *Document Recognition and Retrieval XIX*, volume 8297, pages 118 – 125. International Society for Optics and Photonics, SPIE, 2012.

[9] Ralph Grishman. Information extraction: Techniques and challenges. In *International summer school on information extraction*, pages 10–27. Springer, 1997.

[10] Ralph Grishman and Beth M Sundheim. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[11] Ahmed Hamdi, Elodie Carel, Aurélie Joseph, Mickael Coustaty, and Antoine Doucet. Information extraction from invoices. In *International Conference on Document Analysis and Recognition*, pages 699–714. Springer, 2021.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[13] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[14] Bodhisattwa Majumder, Navneet Potti, Sandeep Tata, James B. Wendt, Qi Zhao, and Marc Najork. Representation learning for information extraction from form-like documents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 6495–6504, 2020.

[15] Eric Medvet, Alberto Bartoli, and Giorgio Davanzo. A probabilistic approach to printed document understanding. *International Journal on Document Analysis and Recognition (IJDAR)*, 14(4):335–347, 2011.

[16] Rasmus Berg Palm, Ole Winther, and Florian Laws. Cloudscan - a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 406–413, 2017.

[17] Rasmus Berg Palm, Ole Winther, and Florian Laws. Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 406–413. IEEE, 2017.

[18] Anand Rajaraman and Jeffrey David Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011.

[19] Pappu S Rao and Vasumathi Devara. To improve the web personalization using the boosted random forest for web information extraction. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, 13(6):1264–1268, 2020.

[20] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[21] Aixin Sun, Myo-Myo Naing, Ee-Peng Lim, and Wai Lam. Using support vector machines for terrorism information extraction. In *International Conference on Intelligence and Security Informatics*, pages 1–12. Springer, 2003.

[22] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[23] Xiaohui Zhao, Endi Niu, Zhuo Wu, and Xiaoguang Wang. Cutie: Learning to understand documents with convolutional universal text information extractor. *arXiv preprint arXiv:1903.12363*, 2019.