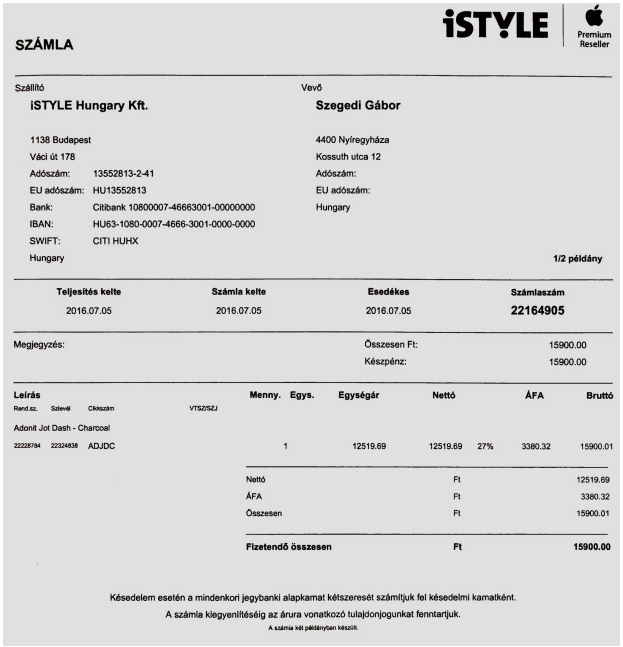
Context-based Information Classification on Hungarian Invoices

Gábor Szegedi, Diána Bajdikné Veres, Imre Lendák, and Tomáš Horváth

Department of Data Science and Engineering
ELTE – Eötvös Loránd University, Faculty of Informatics
Budapest, H-1117 Budapest, Pázmány Péter sétány 1/C., Hungary
vszm@inf.elte.hu, veresdia91@gmail.com, {lendak, tomas.horvath}@inf.elte.hu

Abstract: The field of Artificial Intelligence has always strove towards solving problems with computers that were previously only solvable by humans. An interesting challenge we have these years is extracting information from printed documents. In this publication we focus on the sub-domain of classifying pieces of information on printed invoices. Our goal here was to create a solution capable of finding information on scanned invoices without knowing the template of the invoice. The template-less design is important as invoices can have many different structure based on the issuer. First we feed the invoice image to a commercially available Optical Character Recognition (OCR) engine which returns the extracted texts with their bounding boxes. This information itself wouldn't be enough so we enrich it with feature engineering. The engineered features give information about the content of the text and the context of the surrounding of the bounding box as well as meta information about the entire document. The main novelty of our work comes from how we store contextual information. We then classify the text fragments into 8 categories: Invoice Number, Issue Date, Transaction Date, Due Date, Seller tax number, Customer tax number, Total gross and Other. In this paper we measure the performance of several classification algorithms for our data. We achieved 0.93 macro average F1 measure with our best model.



SZÁMLA

iSTYLE Premium Reseller

Szállító
iSTYLE Hungary Kft.
1138 Budapest
Váci út 178
Adószám: 13552813-2-41
EU adószám: HU13552813
Bank: Citibank 10800007-46663001-00000000
IBAN: HU63-1080-0007-4666-3001-0000-0000
SWIFT: CITI HUHX
Hungary

Vevő
Szegedi Gábor
4400 Nyíregyháza
Kossuth utca 12
Adószám:
EU adószám:
Hungary

1/2 példány

Teljesítés kelte	Számla kelte	Esedékes	Számlaszám
2016.07.05	2016.07.05	2016.07.05	22164905

Megjegyzés:
Összesen Ft: 15900.00
Készpénz: 15900.00

Leírás (Rend.sz. Szlevél Cikkszám VTSZ/SZJ)	Menny.	Egys.	Egységár	Nettó	ÁFA	Bruttó
Adonit Jot Dash - Charcoal						
22228784 22324838 ADJDC	1		12519.69	12519.69 27%	3380.32	15900.01

Nettó	Ft	12519.69
ÁFA	Ft	3380.32
Összesen	Ft	15900.01
Fizetendő összesen	**Ft**	**15900.00**

Késedelem esetén a mindenkori jegybanki alapkamat kétszeresét számítjuk fel késedelmi kamatként.
A számla kiegyenlítéséig az árura vonatkozó tulajdonjogunkat fenntartjuk.
A számla két példányban készült.

Figure 1: A sample printed invoice that was photographed using a mobile phone.

1 Introduction

Information extraction can be a very easy or an impossibly hard problem depending on the kind of data we have. In this paper we are dealing with invoices of various sources. We have received thousands of files from our business partner OTP[1] bank to create a solution for their problem of parsing the incoming invoices. The format of the files fall in 3 main categories:

1. Photographed image of a printed invoice
2. Scanned image of a printed invoice
3. Text based PDF invoice

The problem with the dataset is that we can't use a finite set of templates. The law does not constrain the format of invoices and thus each issuer has its own template for generating invoices. Even if we could manage to create a template for each of the issuers, the number of issuers is growing.

Because of these given conditions we have developed a solution that is generic and free of any premises about the structure of the input document.

1.1 Related Work

The main approaches to invoice recognition are Template based, Graph Convolutional Neural Network (CNN) based and direction based.

Information Extraction (IE) from structured documents has been a studied problem for decades now. One of the first works was *The generic information extraction system* by Jerry R. Hobbs [1]. In his work he laid out an architecture for creating systems capable of extracting information from text using a rule based layer-by-layer approach.

Parsing invoice data is a sub-domain of IE. In the case when the input data consists of a finite set of known

[1] https://www.otpbank.hu/portal/en/home

layouts we can simply extract information based on the known positions of the key pieces. Such a problem does not require sophisticated Machine Learning (ML) solutions, but rather a simple algorithmic approach.

The problem gets significantly harder if we do not know the document templates at training time. In such a case we need to create a generic solution that is smart enough to extract information from unknown document layouts. This field has been studied as well in recent years.

A very recent study entitled *An Invoice Reading System Using a Graph Convolutional Network* was published in 2019, in which the authors turned the output of an Optical Character Recognition (OCR) system into a Graph [2]. The graph's nodes contained the text outputs from the OCR and edges were pointing to nearby nodes based on the location information given by the OCR output. The nodes were then fed to a Graph based CNN to classify for entities of interest.

CloudScan – A configuration-free invoice analysis system using recurrent neural networks is another article from 2017 in which the authors have created a fully fledged end-to-end pipeline for parsing invoices [3]. They get the text from OCR and create N-grams of words in the same line up to a length of 4. They enrich their entities with features based on the properties of the piece of text in the N-gram. They add contextual features based on the closest 4 entities above, below, left and right of the current text. The classifier they use is a Long Short-Term Memory (LSTM) Recurrent Neural Network [4].

2 Scope

Our goal was to find the most relevant information needed by our industry partner. The scope of our work was to extract the following information from the given invoices:

- Invoice Number: The unique identifier of the invoice;
- Issue Date: The date the invoice was issued;
- Transaction Date: The date of reconciliation;
- Due Date: The date until the invoice has to be reconciled;
- Seller tax number: The tax number of the seller party;
- Customer tax number: The tax number of the customer party (This is only included if the customer is a company, and not an individual);
- Total gross: The total amount to be paid by the customer party.

The invoices considered are all printed invoices, thus no handwritten invoices were used as those invoices are getting ousted. The invoices can be image based or "text" based such that, e.g. a portable document format (PDF).

3 Our approach

In any ML problem the biggest dividing factor is how the data is represented. In our case we have a very high level representation of the data given images of invoices. We didn't find it feasible to train a complex Neural Network that operates directly on the image data for the following reasons:

1. The dimensionality of the images is large and we can't shrink them without loosing the small but essential textual information.
2. Training times without dimensionality reduction would be too big even on a special hardware.
3. There are invoices that come in textual PDF format.

For these 3 reasons we chose to simplify the high level representation of the invoice data. We began by breaking down the invoice into a list of *Fragments*. A Fragment is a piece of text with its location *(X, Y, page)* and dimensionality *(width, height)* information.

Creating the list of Fragments is a rather simple task. For text PDF invoices the representation is basically the same as we desire. We just need to use a Python API [2] to get the text with the bounding boxes from each page of the PDF. Converting image based invoices to Fragments is also trivial as we can use an OCR API which returns a list of texts with their bounding boxes. We have integrated our pipeline with Tesseract[3], Azure OCR[4] and OCR Space[5]. We found Tesseract to be quite inaccurate compared to the other two, maybe because our invoices are in Hungarian and not in English. In the end we used OCR Space.

Once we have the list of Fragments the task is finding the relevant ones and extracting the information out of the text. This can be formulated as a classification task, where each Fragment is a separate input to the classifier and the model assigns a label denoting what kind of information is in the Fragment. For each label type we select the Fragment of the highest probability as the Fragment containing the given piece of information.

3.1 Feature Engineering

It should be possible to build a model that we can feed the entire fragment list as input and ask the model to return what is the most probable fragment for each label is. We could use Recurrent Neural Networks and consider the list of fragments a sequence of input. However this would make the problem very hard.

[2] We used pdfminer.six API https://pypi.org/project/pdfminer.six/

[3] https://github.com/tesseract-ocr/tesseract

[4] https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-recognizing-text

[5] https://ocr.space/

Instead of this we have chosen to do extensive feature engineering to add relevant information to each fragment and do classification on each fragment individually. In total we have added 104 features to the existing 6, such that *(text, X, Y, width, height, page)*, what is the main added value of our work. We can group these engineered features into 4 groups which we go into detail below.

Positional features We have added a lot of features that are describing the position of the Fragment relative to the document. We intuitively found this to be very important. For example the invoice number is usually on the top of the first page of a document. To describe this first we needed to add a feature for storing which page the Fragment is part of. Then we need the features to describe the X and Y coordinates of the Fragment on the given page. Lastly we have added features to describe the dimensionality of the Fragment.

Document features Another set of features added were to describe the entire document. Knowing the positional attributes of a Fragment is not meaningful without knowing the dimensionality attributes of the document. Thus we have added features for document width and height, their ratios, the number of pages. We also added features describing where are the outermost Fragments for the current page and for the entire document.

Text features In ML, when dealing with text, the problem arises from the fact that our models are only capable of working with numbers. We need to encode the text into a finite set of features all the time, so that is what we did as well for our Fragments.

One obvious feature that we can always use to quantify text is the length of the text we have. We added this of course to our features.

Another set of features we used were counters of special characters in the Fragment. For example the count of whitespaces is an important feature telling us how many words are there in the fragment. Counters of digits and alphabetical characters are also good indicators of what kind of text are we dealing with. The count of slashes and dashes could imply that the current Fragment contains a bank account number or an invoice number. We could add many more special characters depending on the data we have.

Lastly we have added a few Boolean features derived from the text based on whether the text matches a given regular expression or not. We created regular expression features for matching date, tax number, currency and decimal point.

Contextual features The feature categories we discussed so far are standard features we have seen in other related works as well [3] [5] [2]. These features in themselves wouldn't be enough to do labeling effectively, we need the context of the Fragments as well. We have also seen the idea of adding context information to the Fragments in other works but there is no consensus on what is the best way to do this.

In [3], the authors give context information by adding the features of the nearest neighboring Fragments in the 4 general directions (Up, Down, Left, Right).In [2], a very similar method is used as the graph nodes connect to the 4 general directions as well. In [6], the authors took a different approach by storing the polar coordinates of each fragment relative to the current fragment.

We take a new approach in capturing the context of the Fragment. We have identified 38 keywords in invoices that are indicative of important information in the vicinity. For each Fragment we search for the nearest occurrence of all the keywords. We calculate the distance from the Fragment to the keyword and store the normalized x and y values as features for each of the keywords. After doing this for each keyword we get a complex web over our invoice document. We can see this mechanism in Figure 2 for a single Fragment of a sample invoice.

Using such keywords comes naturally to us humans when we interpret an invoice. For example if we take a look at an invoice and we see 2 tax numbers we can decide which belongs to the seller by deciding which one is closer to the keyword *Seller* or *Provider* [6].

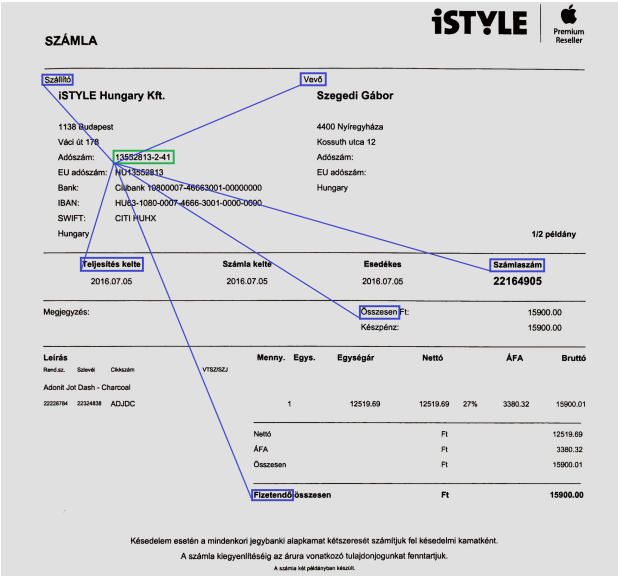


Figure 2: A sample invoice where the Fragment in the green bounding box is the tax number of the seller and the closest keywords are highlighted with blue bounding boxes. Note that some of the keywords occur multiple times on the invoice but we chose the closest one for the current Fragment.

[6] We use Hungarian keywords only as the dataset is Hungarian

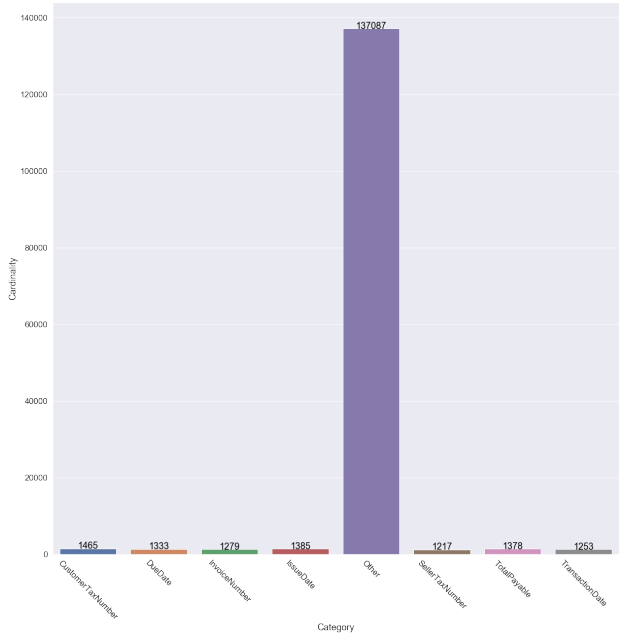


Figure 3: Class distribution of Fragments on one of our test sets.

There are 2 special cases to note when pointing the to the nearest keyword. The first one is when the current fragment contains one of the keywords. In that case we chose to set the value of the x and y features to be zero. The second special case is when a certain keyword is not to be found on the invoice. In this case we assign -1 to the x and y features of the keyword. Our approach is different from the related works because we neither restrict the context to the 4 nearest Fragments in the general directions, nor do we include all the features. The contextual information is relevant because of the keywords these vectors are pointing towards.

4 Classification

After preparing the data we had to train a classifier for labeling the Fragments. During the training and evaluation we had to be cautious as the dataset is very unbalanced because on each document there are hundreds of text Fragments but we are only interested in 7 of them. See Figure 3 for the distribution of the 8 class labels.

As 93% of this data was labeled as *Other*, a classifier that assigns the *Other* label blindly to any input would achieve an overall accuracy of 93%. This is of course not good as our focus is on the classes with low cardinality. To overcome this we are using the macro average of the F1 Score of the classes to compare the classifiers.

We have chosen 3 classifiers for comparison: Decision Tree Classifier, Random Forest Classifier [7] and Extreme Gradient Boosting Classifier [7].

[7]For the Decision Tree and Random Forest models we used the implementations from scikit-learn

5 Experiments

The data we received consisted of thousands of Invoices from different issuers in various file formats. About half of them were electronic invoices in textual PDFs, while other half was images of either photographs or scans of printed invoices.

First we had to manually annotate the ground truth about the invoices. For each invoice we stored the relevant information pieces that were in scope.

After that we had run the OCR engine on all of the image based invoices. We then filtered out those image based invoices where the OCR output did not contain at least the invoice number and the seller tax number. We did this because the OCR engine optimization is out of our scope and this way we eliminated any errors due to image quality or bugs in the OCR engine used. We had 2000 invoices that met this criteria.

Lastly we ran the algorithm to convert the filtered invoices into Fragments. We manually labeled the Fragments to eliminate any OCR issues, or data format differences there may be. This was the data that we could build our models upon.

We have split the Fragments into train, test and validation sets as usual in the field. The ratios used were 60% for training, 20% for validation (tuning the hyper-parameters) and 20% for comparing the trained and optimized models. The results are visible in Table 5. Without surprise the XGBoost Classifier performs the best out of these 3, but even the Decision Tree and Random Forest models are very close to the Gradient Boosting model. XGBoost outperforms the other 2 on all of the labels and in the overall macro average as well. The tree based models fall behind in the *Total Gross* label.

6 Conclusion and future work

In this work we presented a method of identifying key information on Invoices. We have built upon external dependencies via Optical Character Recognition, which is used to break down the image based invoice into text pieces with positional information called Fragments. We did an extensive feature engineering so that the Fragments contain features regarding the properties document, the contained text and the context of the text as well.

The main added value of our work comes from how we add contextual information to the Fragments. It works by pointing vectors from the Fragment to each of the nearest keywords. This gives the power of our method to generalize so that unlike some of the previous works it can perform well on previously unseen Invoices.

However our work does not stop here. We plan on creating an end-to-end solution for extracting the information from invoices. We have merely identified the Fragments

	Decision Tree	Random Forest	XGBoost
Other	0.99	0.99	1.00
Invoice number	0.81	0.86	0.89
Issue date	0.88	0.89	0.93
Transaction date	0.87	0.89	0.94
Due date	0.88	0.93	0.96
Seller tax number	0.89	0.91	0.92
Customer tax number	0.92	0.93	0.94
Total gross	0.71	0.75	0.87
Macro Average	0.8685	0.8952	0.9297

Table 1: Comparison of models after hyperparameter optimization. The numbers are F1 scores.

correctly, but extracting the information from the selected Fragments still holds challenges.

As our main focus was on feature engineering there can still be room for improvement in the modeling side. We are planning on experimenting with Neural Network models in the future to see if we can improve our accuracy.

We also plan on embedding the system in a service, where the user sends in the invoice and our system either returns the extracted information, or a new version of the invoice image where the key information points are highlighted like in Figure 4.

Acknowledgements The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

Supported by Telekom Innovation Laboratories (T-Labs), the Research and Development unit of Deutsche Telekom and EIT Digital.

References

[1] J. R. Hobbs, "The generic information extraction system," in *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993.

[2] D. Lohani, A. Belaïd, and Y. Belaïd, "An invoice reading system using a graph convolutional network," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 144–158.

[3] R. B. Palm, O. Winther, and F. Laws, "Cloudscan-a configuration-free invoice analysis system using recurrent neural networks," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 406–413.

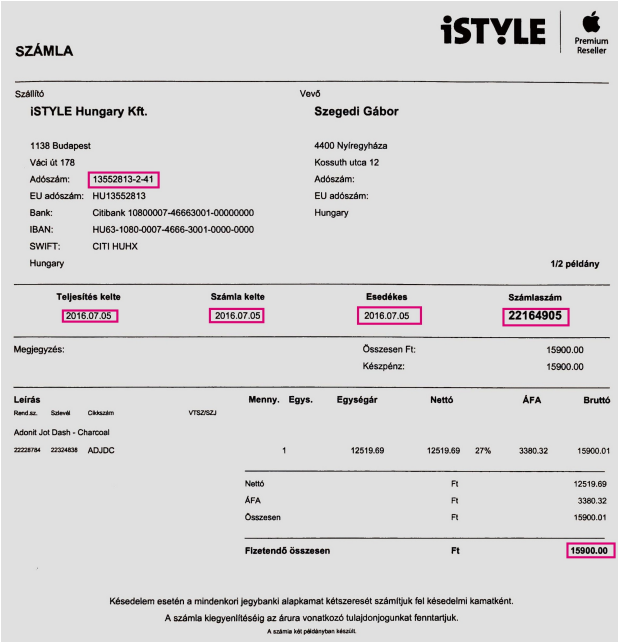


Figure 4: The key information pieces our system can identify from the Fragments.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] H. T. Ha, Z. Nevěřilová, A. Horák *et al.*, "Recognition of ocr invoice metadata block types," in *International Conference on Text, Speech, and Dialogue*. Springer, 2018, pp. 304–312.

[6] M. Rusinol, T. Benkhelfallah, and V. Poulain dAndecy, "Field extraction from administrative documents by incremental structural templates," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1100–1104.

[7] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.