

Automatic generation of a custom corpora for invoice analysis and recognition

Jérôme Blanchard, Yolande Belaïd, Abdel Belaïd

▶ **To cite this version:**

Jérôme Blanchard, Yolande Belaïd, Abdel Belaïd. Automatic generation of a custom corpora for invoice analysis and recognition. ICDAR-WIADAR, Sep 2019, Sydney, Australia. hal-02460869

HAL Id: hal-02460869
https://inria.hal.science/hal-02460869

Submitted on 30 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Automatic generation of a custom corpora for invoice analysis and recognition

Jérôme Blanchard
FAIR[&]SMART
11 Rempart St Thiébault,
57000 Metz, France
jerome.blanchard@fairandsmart.com

Yolande Belaïd
Université de Lorraine - LORIA
Campus scientifique
54500 Vandoeuvre, France
yolande.belaid@loria.fr

Abdel Belaïd
Université de Lorraine - LORIA
Campus scientifique
54500 Vandoeuvre, France
abdel.belaid@loria.fr

***Abstract*—In this paper, we present a bill-type document generator capable of supplying on demand all the mass of documents that a learning system needs. The lack of administrative documents has long been a handicap because of the confidentiality of this type of document. In addition, this generator allowed us to solve the problem of annotations since they are done automatically during the generation and put directly in XML-GEDI form. Then, to show the interest of the generator, we proposed a system of invoice recognition based on graph convolutional neural network. The experiments took place in excellent conditions since we had all the possibilities to vary the classes, the samples in the classes, and their parameters.**

***Keywords*-Invoice generator, GEDI format, Graph Convolutional Neural Network**

I. Introduction

In order to extract named entities from scanned invoice's documents, we are using an Artificial Intelligence (AI) system that has been trained on annotated invoices taken from real life. We faced a common problem in AI training due to the fact that our invoices corpora was not big enough to produce relevant results using deep learning neural network. We realized that building a well annotated corpora is very costly, generates intellectual property problems and is not easy to customize when you want to add or modify some annotations. We decided to built a software to automate the generation of a custom corpora, based on an invoice data model made of real data. Our methodology could be used for any other data set that need an automated annotation based ground truth. The generated data (images and annotations) can be used solely or mixed with real life data in order to improve an existing dataset, especially for AI training.

This generation of on-demand documents goes with the idea of wanting to process streaming data without class limits. However, most existing systems like Intellix [4], ITESOFT [5], [6], smartFIX [7] and others [8], [9] create specific models limited to the data they have. Due to their dependence on seeing the template beforehand, these systems cannot accurately extract information from unseen layouts of invoices. CloudScan [10] is perhaps the only model so far which can handle unseen layout invoices quite well. It is based on classification of word n-grams into entities of interest instead of mapping of words to fields.

We propose a generic approach to deal with all the entities in the invoice in or outside table. We model the whole invoice document as a document graph of words, then we classify each word in the document into classes of interest to extract through a graph convolutional network (GCN) and finally we group the words of same classes together to obtain the final entities. The power of our system lies in the graph modeling and GCN, which takes into account the features of its neighboring words and their inter-relationships to decide the class of a word [1].

The paper is organized as follows: Section II presents the generator and its different possibilities. Section III shows the invoice analysis approach. Finally, we give some conclusion remarks in IV.

II. Invoice generation

By analyzing real invoices, determined that it was nearly always composed of the same elements, some mandatory or always present like: a Head zone containing Company Informations, Client Informations, Invoice Numbers and Dates; a central zone containing Products, Payment Infos and some ID Numbers and a Footer zone containing various informations but at least the Totals. Others elements are optional or their position can vary a lot like: Company Logo, some Company Informations like VAT Number, SIRET, Expedition Type and Shipping Cost, etc.

In order to be able to produce any kind of invoice, we first defined a model containing all possible invoice data (see Figure 1). Model structure is critical for the accuracy of the training and for the final usage. It must be fine grain enough to ensure good annotation accuracy. We defined classes that we want to extract or classify using the AI process. We decided to use an object based model to define all the components of an invoice. All those objects are organized in a tree where root node is the invoice itself.

All objects that compose the data tree have to provide a generator for its own type. This generator can use several kinds of data source: random generation, data source based generation, set of values, set of regular expressions. Thus, generation of an invoice data results in crawling the tree to generate each node using its own generation implementation. Container nodes are responsible for propagating the

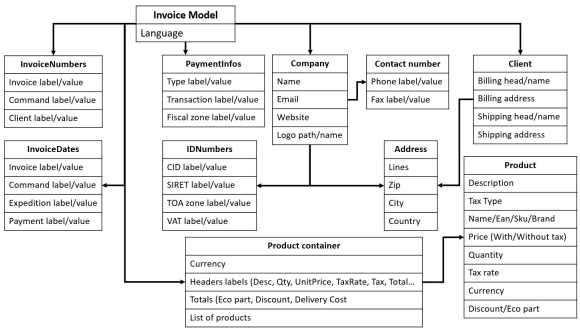


Figure 1: Invoice Model Schema.

generation to their children.

A. Context generation

For each invoice generation, some definitive choices are made during tree generation and those choices have impact on the rest of the generation: for example, the generation language, currency or date format. As soon as the currency is chosen it has to be taken into consideration for many elements of the tree. To enforce this coherency across all nodes of the invoice, we are using a dedicated structure, the Generation Context, which is propagated to all nodes during the tree generation. Some context choices can have impact where it seems it should not: the choice of the company logo can be directly linked with the country previously set in the context avoiding inconsistency on logo slogan that are not in the same language than the rest of the invoice. Thus, the generators of some model nodes can be increased with newly discovered context dependencies by adding heuristics in this node generator based on context values.

B. Data generation strategies

The first strategy for generation is to use a set of regular expressions to produce data. This works well for invoice numbers or invoice dates. Those nodes are using predefined regular expression based on real life invoice data observation. Then, we use a library (https://github.com/mifmif/Generex) that is able to randomly generate any value in the whole regular expression domain possibilities. The regular expressions can also be conditioned with some generation context's choices, some are dedicated to specific language or country for example.

A second strategy for generation is to use real life open databases. For example, addresses are randomly generated using public base for all countries that we want to support. This can create more realistic invoice using, for a Belgium invoice, real Belgium addresses. We also use, for labels or some textual parts of an invoice, predefined set of all values that we already observed on real invoices. Those samples are ordered by language and/or by country.

Another generation strategy is to include some pseudo random in some heuristic choices, sometimes with a statistical distribution of results. For example, the client's billing address is often the same than the client's shipping address so we used a generation proportionality of 80% the same and 20% another adress. In the same way, command date, shipping date and billing date are generated using some heuristics that avoid a shipping date prior or too far of the command date. We also use some heuristics for the generation of the products included in the invoice: depending on the product category, the product random quantity is not the same (it is reasonably possible to buy 10 ink cartridge but not to buy 10 microwaves).

C. Global generation process

The global generation process consists of fixing or randomizing some context elements (country, language) and of starting an iterative process to be able to generate as much as invoice data we need.

The generation strategies as much as the data sources (products, addresses, companies logos, etc.) and the regular expressions for elements generation can be easily expended and included in a new generation cycle by improving node generators heuristics or data sources.

D. Graphical representation of the invoice model

Once the data model is generated, we can build a graphical representation of that invoice. We decided to generate invoices in PDF that we can export in an image format (TIFF or JPG). The graphical generation consists in drawing all pieces of invoice data model onto a single page using a dedicated Layout. The layout is responsible of placing model data on the sheet but also to produce the annotations corresponding to the graphical position and size of all elements; image is generated in a PDF flow and annotations in an XML flow (in the GEDI format (https://sourceforge.net/projects/gedigroundtruth/)). As for the invoice model, layout elements are built in a tree structure.

E. Graphical elements generation

We created a set of graphical components with basic behavior of typical graphical toolkits but also with the annotation process built-in. We started by implementing the smallest object of our kit: the Text Box. This basic component consist of a part of text which can be drawn into the picture. More complexity is added to this simple behavior like, the desired font, the padding, the box size (allowing to split text if end of line is reached), color, background color, line height. More than only graphical consideration, we give also annotation consideration like the entity name that corresponds to this text.

F. Annotation generation

The annotation generation consists of creating an XML element based on GEDI schema. The annotation is included

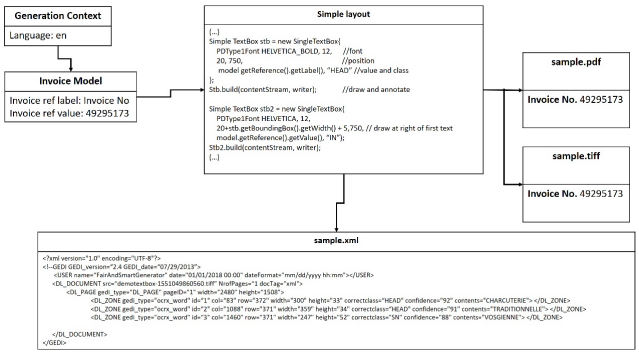


Figure 2: Annotation generation schema.

in an XML stream and include position and box size of the text zone but also the text itself and the annotation class that we want to learn. Layout is also responsible, by using invoice model data, to fix annotation classes.

G. Layout for invoices

In order to produce invoices with complex layouts, we have developed graphical objects that acts as containers for Text Box. Thus, we have Horizontal Container, Vertical Container, Table Row, etc. We also defined some specific elements for graphical purpose only like Graphic Lines, Images, Borders, etc.

H. Fixed based layout

Fixed based layout are a simple static based combination of existing elements to produce a "clone" of a real invoice and to use it for the generation of many invoices with various data. This allows to have a ground truth very close to the reality but with a bigger number of invoices. Nevertheless, we need to have many existing samples in order to have a varied ground truth and layout integration can be costly in this case.

I. Generic layout

We decided also to work on generic layout by defining parts of an invoice that could produce random behavior. We defined Global Zones (Header, Product Table, Product Footer, Global Footer). Header Zone will be divided in four regions allowing to place randomly global components: Company Information, Order information, Billing Address, Shipping Address. The rendering of all components and regions will be randomized using specific graphical representation (background color, font size, border, etc.). This will allow to have mixed rendering layout based on a rich graphical library. The Table 3 shows the different element location and neighbor relationships, while the Table I highlights the elements appearing in the different blocks, indicating whether they are mandatory or optional relationships

The Figure 4 shows an example of generated invoices, one from actual invoice (a) and one generated randomly (b).

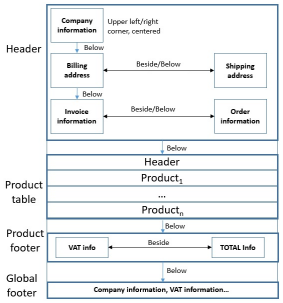
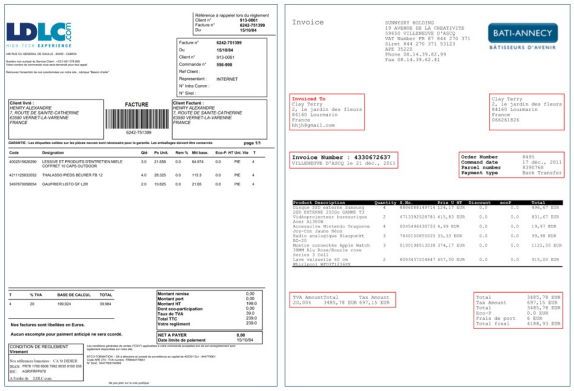


Figure 3: Main table elements and their location and inter-relationships.



(a) Actual (b) Random

Figure 4: Examples of generated invoices: (a) from actual invoice by varying the information inside the blocks, and (b) randomly generated.

III. Invoice Analysis

The global system has been published in [1]. We are going to just summarize here the main idea of the system. The image is run through an OCR engine (Tesseract). From this output, we only take word zones and ignore all other zones such as graphic line, area, image, etc., because the higher up zones are composed of word zones and accumulate more ocr segmentation errors. Then, features are calculated for each word. These words are used to model the complete document as a graph with words as nodes and edges depicting neighborhood relationships. This document graph is fed to a graph node classifier which classifies each word into classes of interest. Finally words belonging to same classes are grouped together following left to right ordering to form entities.

Text feature calculation is basically converting the word text into a meaningful vector representation. For this task, we use BPEmb [13]. We obtain a 317 dimensional feature vector of every word in the document. The last 17 elements are reserved for the existence of keywords like date, zipcode, city, etc.

The whole document is modeled as a graph with words

Table I: Intra-element constraints

Type	Delivery / Billing	Invoice information	Order Information	Product table	Summary table / totals
Mandatory	Last name / First Name No. and street ZipCode, City name	Company Name No. and street ZipCode, City name	Invoice number Date	Order number	Quantity Name Unit Price excl tax Total Price excl tax
Optional	Country Tel number @mail	Country Tel number Fax N° @mail @site	Location Client number Company Name	Order date Client number Payment mode Payment Total Sending date Sending mode Parcel number	Product reference Unit price incl tax Discount Final unit price incl tax Eco participation VAT

as nodes and edges denoting nearest neighbors of a word in 4 major directions. This is different from, for example, the system of ITESOFT [5] using a star graph for each important word. For modeling, we were inspired by the system of [14] who used a Graph Convolutional Networks (GCN). We used more than 3000 images generated by our generator, by selecting those which seemed to us quite representative of the sought variations. In current system, we scan each image in 300 dpi. The invoices are annotated at word level by providing each word a ground truth class.

IV. Conclusion

In this paper, we presented two ways to automatically generate invoices, inspired by actual invoices to which we applied random variations of a few elements. The other is completely free, by simply considering the elements that must appear in the invoice, but by introducing inter and intra-element relations. The organization scheme of the elements in the page was complex because the constraints are numerous. We opted for reasonable constraints leading to the production of fairly realistic bills. This work has allowed us to generate enough well annotated invoice samples to allow learning and recognition in good conditions.

References

[1] D. Lohani, A. Belaïd and Y. Belaïd. An Invoice Reading System using a Graph Convolutional Network. IWRR, Dec 2018, PERTH, Australia. 2018.

[2] B. Klein, S. Agne and A. Dengel: Results of a study on invoice-reading systems in Germany. DAS, pp. 451-462, 2004.

[3] D. Nadeau and S. Sekine: A survey of named entity recognition and classification. Lingvisticae Investigationes, 30, pp. 3-26, 2007.

[4] D. Schuster and K. Muthmann and D. Esser and A. Schill and M. Berger and C. Weidling and K. Aliyev and A. Hofmeier: Intellix-end-user trained information extraction for document archiving. ICDAR, 2013, pp. 101-105.

[5] M. Rusinol, T. Benkhelfallah and V. Poulain d'Andecy: Field extraction from administrative documents by incremental structural templates. ICDAR, 2013, pp. 1100-1104.

[6] H. Hamza, Y. Belaïd and A. Belaïd: A case-based reasoning approach for invoice structure extraction. ICDAR, 2007, pp. 327-331.

[7] A. Dengel and B. Klein: smartfix: A requirements-driven system for document analysis and understanding. DAS, 2002, pp. 433-444.

[8] F. Cesarini, E. Francesconi, M. Gori and G. Soda: Analysis and understanding of multi-class invoices. ICDAR, 2003, pp. 102-114.

[9] V. P. d'Andecy, E. Hartman and M. Rusiñol: Field extraction by hybrid incremental and a priori structural templates. DAS, 2018, pp. 251-256.

[10] R. B. Palm, O. Winther and F. Laws: Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. ICDAR 2017, pp. 406-413.

[11] T. Kasar, T. K. Bhowmik and A. Belaïd: Table information extraction and structure recognition using query patterns. ICDAR, 2015, pp. 1086-1090.

[12] R. Sennrich, B. Haddow and A. Birch: Neural machine translation of rare words with subword units. 54th Annual Meeting of the Association for Computational Linguistics, 2016, pp. 1715–1725

[13] B. Heinzerling and M. Strube: BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In: LREC 2018, pp. 2989-2993.

[14] T. N. Kipf and M. Welling: Semi-supervised classification with graph convolutional networks. ICLR, 2017, pp. 1-14.

[15] M. Defferrard, X. Bresson and P. Vandergheynst: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, 2016, pp. 3844-3852

[16] R. Smith: An overview of the tesseract ocr engine. In ICDAR 2007, pp. 629-633.

[17] D. P. Kingma and J. Ba: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.