An Invoice Reading System Using a Graph Convolutional Network

D. Lohani[1], A. Belaïd[2(✉)], and Y. Belaïd[2]

[1] MOSIG program, GVR, INP, 38000 Grenoble, France
devashishlohani@gmail.com
[2] Université de Lorraine-CNRS-LORIA,
Campus scientifique, 54500 Vandoeuvre-lès-Nancy, France
{abdel.belaid,yolande.belaid}@loria.fr

Abstract. In this paper, we present a model-free system for reading digitized invoice images, which highlights the most useful billing entities and does not require any particular parameterization. The power of the system lies in the fact that it generalizes to both seen and unseen layouts of invoice. The system first breaks down the invoice data into various set of entities to extract and then learns structural and semantic information for each entity to extract via a graph structure, which is later generalized to the whole invoice structure. This local neighborhood exploitation is accomplished via a Graph Convolutional Network (GCN). The system digs deep to extract table information and provide complete invoice reading upto 27 entities of interest without any template information or configuration with an excellent overall F-measure score of 0.93.

1 Introduction

We seek to set up a platform for managing personal data, which complies with the European recommendations on data security [1]. This platform must offer everyone the opportunity to manage his data, to secure, update, consolidate and evolve it. If the feeding of recent data does not pose too many problems, that of the old data requires careful digitization and retro-conversion. The data referred is mainly of the administrative document type and concerns contracts, invoices, pay slips, etc. The scanning is done by the customer and sent to the platform which retrieves the relevant information and provides services to the customers.

The work focused on the processing of invoice images. Invoices broadly contain two types of information: information relating to the issuing company and the receiving customer (generally corresponding to named entities of address type, numbers and billing dates, etc.) and information relating to the products ordered (containing labels, taxes and amounts). Several methods exist for information extration in invoices. Most of them are based on comparing the input document with an already observed template, e.g. rule, keyword or layout based techniques. Many systems first classify the templates, e.g. Intellix [2], ITESOFT [3,4], smartFIX [5] and others [6,7]. Due to their dependence on seeing

G. Carneiro and S. You (Eds.): ACCV 2018 Workshops, LNCS 11367, pp. 144–158, 2019.
https://doi.org/10.1007/978-3-030-21074-8_12

the template beforehand, these systems cannot accurately extract information from unseen layouts of invoices. CloudScan [8] 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. Even though their system performed quite well for simpler entities like date and invoice number, they were not able to extract complex multi-dimensional entities like company or client addresses. It is due to the fact that their system works linearly due to n-grams but invoices also have 2-dimensional relations within entities like in tables and addresses.

The extraction of information inside table is itself a very complicated research topic and has a very related limited work [4,9,10]. Authors in [9,10] have similar approach where an input pattern in table is provided by the client for fields to extract. This pattern is modeled as a graph, which is used to mine similar graphs from a document image in order to produce a model. The biggest problem here is that the client needs to intervene in each invoice and draw a pattern to extract. Furthermore, the results show that it is difficult to adapt to new structures of table and we need to have more or less similar images for proper extraction, which is not the real world case.

Hamza et al. [4] have so far developed the most comprehensive system which deals with both entities inside and outside table. It analyses a document by retrieving and analyzing similar documents or elements of documents (cases) stored in a database. The retrieval step is performed using graph probing. The analysis step is done to the information found in the nearest retrieved cases. The problem with this system is that it is similar to template model as it has to store various cases and solutions but not every time the solution can fit the real world case.

We propose a generic approach to deal with all the information in the invoice in and 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 using GCN which takes into account the features of its neighboring words and their interrelationships to decide the class of a word. The system automatically generalizes to various structures of entities to extract and learns their characteristics. We evaluate our model on a large dataset and provide very detailed and competitive results.

The paper is organized as follows: Sect. 2 presents our approach, Sect. 3 shows the experiments and results. Finally Sect. 4 concludes the work and provides future guidelines.

2 Approach

2.1 System Overview

Our system consists of four major steps as shown in Fig. 1. The system starts by extracting only words from the image. Features are calculated for each one of

them by word embedding. The resulting vectors 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 to form entities.

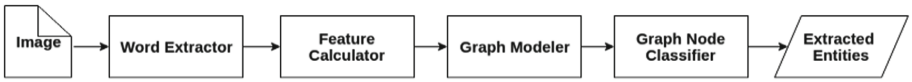

Fig. 1. Schema of our proposed approach

2.2 Word Modeling

Word modeling consists of two steps: word extraction and feature calculation with word representation, which will be detailed in the following.

Word Extraction: The invoice image is run through an OCR engine and the output is retrieved in HOCR format. From this output, we only take word zones (word id, content and bounding box) and ignore all other zones such as graphic lines, photos, blobs, etc., because the higher up zones are composed of word zones and accumulate more OCR segmentation errors. We perform the noise removal on word zones by simply avoiding words with extra big or extra small sizes, words without any content (usually table border and margin lines) and words with erroneous non-alphanumeric contents (prominent when salt and pepper noise is present). The output of this step is a collection of "good" words with their id, content and bounding box information.

Feature Calculation: For each word, we calculate boolean, numeric and text features as follows:

1. Boolean features are calculated as follows:
   (a) isDate: a parser to check whether a word or part of word could be a date.
   (b) isZipCode: checks if a 6 digit zipcode belongs to a small database of zip codes.
   (c) isKnownCity, (d) isKnownDept, (e) isKnownCountry: checks the word in a small database of known cities, departments and countries.
   (f) nature: an 8 dimensional binary vector which denotes the presence of a specific nature of the word. It includes: isAlphabetic, isNumeric, isAlphaNumeric, isNumberwithDecimal, isRealNumber, isCurrency, hasRealandCurrency, mix (except these categories), mixc (mix and currency word).
   We get a 13 dimensional boolean vector as output.

2. Numeric features of a word consists of its relative distance to its nearest neighbors (refer to Sect. 2.4 for more details) in 4 major directions (left, right, top and bottom) (refer to Fig. 2). Relative distances are calculated as follows:

$$RD_L = (Right(Word_{Left}) - Left(Word_{Source}))/Width_{Page} \quad (1a)$$

$$RD_T = (Bottom(Word_{Top}) - Top(Word_{Source}))/Height_{Page} \quad (1b)$$

$$RD_R = (Left(Word_{Right}) - Right(Word_{Source}))/Width_{Page} \quad (1c)$$

$$RD_B = (Top(Word_{Bottom}) - Bottom(Word_{Source}))/Height_{Page} \quad (1d)$$

   Since the values are increasing from left to right and from top to bottom, so RD_L and RD_T are negative while RD_R and RD_B are positive. Each value is normalized with the highest possible value, so absolute value for each of four variables is always less than 1.
3. Text feature calculation is basically converting the word text into a meaningful vector representation. For this task, we use Byte Pair Encoding (BPE) [11] over Glove or Word2Vec because of its ability to deal with out of vocabulary words. This approach breaks the word into subwords to deduce the meaning of the complete word. BPE is an unsupervised subword segmentation method which starts with a sequence of symbols, for example characters, and iteratively merges the most frequent symbol pair into a new symbol. This proved out to be very useful in OCRed invoice images as we were able to deduce the meaning of a word correctly even in the presence of OCR errors due to its subwords.

   Word Representation: We use BPEmb [12], a recent collection of pretrained BPE vectors. We break the input word into maximum of 3 subwords if possible, using the French and English learned vocabulary. Then, we fetch for each subword a 100 dimensional embedding vector. As an output, for each word of the document, we get a 300 dimensional embedding vector.

Finally, we obtain a 317 dimensional feature vector of every word in the document.

2.3 Graph Modeler

In this step, the whole document is modeled as a graph with words as nodes and edges denoting nearest neighbors of a word in 4 major directions.

Algorithm 1. Line Formation

1: Sort words based on Top coordinate
2: Form lines as group of words which obeys the following:
   Two words (W_a and W_b) are in same line if:
   $Top(W_a) \leq Bottom(W_b)$ and $Bottom(W_a) \geq Top(W_b)$
3: Sort words in each line based on $Left$ coordinate

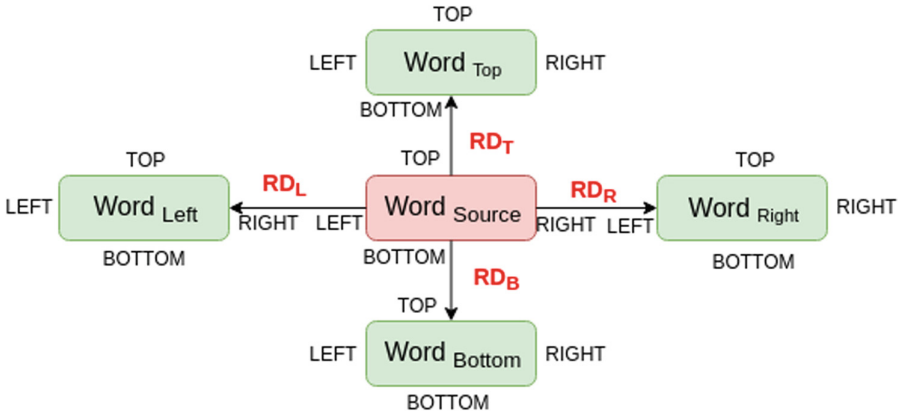

Fig. 2. Nearest neighbors of the source word, $Word_{Source}$ are $Word_{Left}$, $Word_{Top}$, $Word_{Right}$ and $Word_{Bottom}$. Each of the word has four bounding box coordinates: $Left$, Top, $Right$, $Bottom$ depicting its extreme coordinates in 4 directions. Relative distances of $Word_{Source}$ with neighboring 4 words are designated as RD_L, RD_T, RD_R, RD_B for $Word_{Left}$, $Word_{Top}$, $Word_{Right}$ and $Word_{Bottom}$ respectively.

We first run the Line Formation algorithm as defined in Algorithm 1. As a result we get lines as array of words where within each line words are sorted from left to right and lines themselves are read from top to bottom. This ensures that words are read from top left corner of the image first, going line by line from left to right and at last the final bottom right word of the page is read. Note that here notion of line is just a group of words which are well aligned horizontally and we are not forming an actual line rectangle because in our case, lines are used just to read words in right order to build the document graph. Let W_{doc} denote the set of all words in the document. Mathematically, we define the undirected document graph as $G_{doc} = (W_{doc}, E)$, where each $v \in W_{doc}$ corresponds to a word and each edge $e \in E$, follows the Algorithm 2. The graph structure is stored in an unweighted adjacency matrix A which denotes nearest neighbor relationships of all words in a document. We can see in Fig. 3(B) that this approach provides generic graph for complete document without any user intervention and due to its low degree, it is computationally efficient unlike star graph for every entity in ITESOFT system as shown in Fig. 3(A). One can also observe in Fig. 3(B) that each word can have atmost 4° and only one edge in each direction. Words that are read before are given the priority in case of ambiguity. Eg: Word "le" at bottom left of Fig. 3(B) has no top edge connecting it to word "anticipe" because word "anticipe" was read before and it chose word "fois" as bottom edge rather than word "le" even though they had the same distance because left word is preferred in ambiguity as described in Algorithm 2.

Algorithm 2. Graph Modeling Algorithm

1: Read words from each line starting from topmost line going towards bottommost line
2: For each word, perform following:
  2.1 Check words which are in vertical projection with it:
  2.2 Calculate RD_L and RD_R for each of them (refer Sect. 2.3)
  2.3 Select nearest neighbour words in horizontal direction which have least magnititude of RD_L and RD_R, provided that those words do not have an edge in that direction
    2.3.1 In case, two words have same RD_L or RD_R, the word having higher top coordinate is chosen
  2.4 Repeat steps from 2.1 to 2.3 similarly for retrieving nearest neighbour words in vertical direction by taking horizontal projection, calculating RD_T and RD_B and choosing words having higher left coordinate incase of ambiguity
  2.5 Draw edges between word and its 4 nearest neighbours if they are available

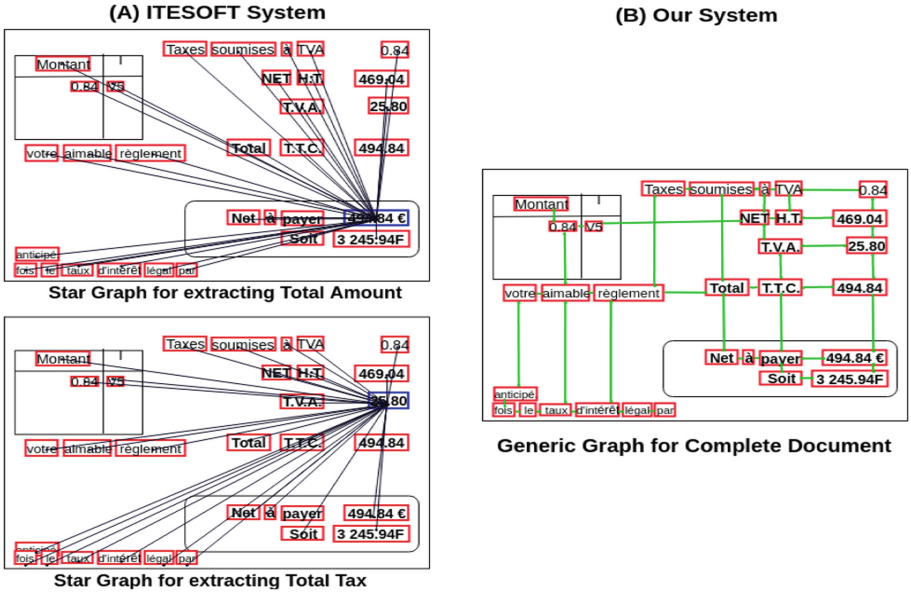

Fig. 3. Graph modeling: (A) ITESOFT system [3] with star graph for each entity to extract (B) Our System with one generic graph for complete document

2.4 Graph Node Classifier

In this step, we consider the problem of classifying nodes (words in our case) in a graph (such as a document graph), where class labels are available for some documents. The problem is basically graph node classification. In our context, it is very important to classify a node by looking into its neighborhood attributes in the graph. To solve this problem, recently [13] used multilayer neural networks operating on graphs called Graph Convolutional Networks (GCN). GCNs are

neural networks operating on graphs and inducing features of nodes (i.e. real-valued vectors/embeddings) based on properties of their neighborhoods. In [13], authors show GCN to be very effective for the node classification task: the classifier was estimated jointly with a GCN, so that the induced node features were informative for the node classification problem. Depending on how many layers of convolution are used, GCNs can capture information only about immediate neighbors (with one layer of convolution) or any nodes at most K hops aways (if K layers are stacked on top of each other).

The basic idea is based on spectral graph theory that the graph convolutions can be dealt as multiplications in the graph spectral domain. The feature maps can be obtained by inverse transform from the graph spectral domain to original graph domain.

In our paper, the word features are learnt by GCN given the graph representation of the document. Given an invoice document, we define its input graph feature vector by F_{in} and we denote the output feature vector after graph convolution by F_{out}. Firstly, F_{in} is transformed to the spectral domain via graph Fourier transform. This transform is based on the normalized graph Laplacian, defined as $L = I_N - D^{-1/2}AD^{-1/2}$, where I_N and D are respectively the identity matrix and the diagonal degree matrix of the graph structure G. Then, L can be eigendecomposed as $L = U\Lambda U^T$, where U is a matrix of eigenvectors and Λ is a diagonal matrix whose diagonal elements are eigenvalues of L. The Fourier transform of F_{in} is a function of U defined as:

$$\hat{F}_{in} = U^T \times F_{in} \tag{2}$$

while the inverse transform is defined as:

$$F_{in} = U \times \hat{F}_{in} \tag{3}$$

The convolution of F_{in} with a spectral filter g_θ is given by:

$$F_{out} = g_\theta * F_{in} = U * g_\theta * U^T * F_{in} \tag{4}$$

where parameter θ is a vector to learn. In order to keep the filter K-localized in space and computationally efficient, [14] proposes an approximated polynomial filter defined as:

$$g_\theta = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) \tag{5}$$

where $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$, $\tilde{L} = \frac{2}{\lambda_{max}}L - I_N$ and λ_{max} denotes the largest eigenvalue of L. $T_k(x)$ is the Chebyshev polynomial of x upto k order. The filtering operation can then be written as $F_{out} = g_\theta F_{in}$. In our model, we use the same filter as in [14] (Fig. 4).

For the graph representation of an invoice document, the i^{th} input graph feature $f_{in,i} \in F_{in}$ of word node v_i is the 317 dimensional feature vector as calculated in Sect. 2.3. Then, the i^{th} output feature $f_{out,i} \in F_{out}$ is:

$$f_{out,i} = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) f_{in,i} \tag{6}$$

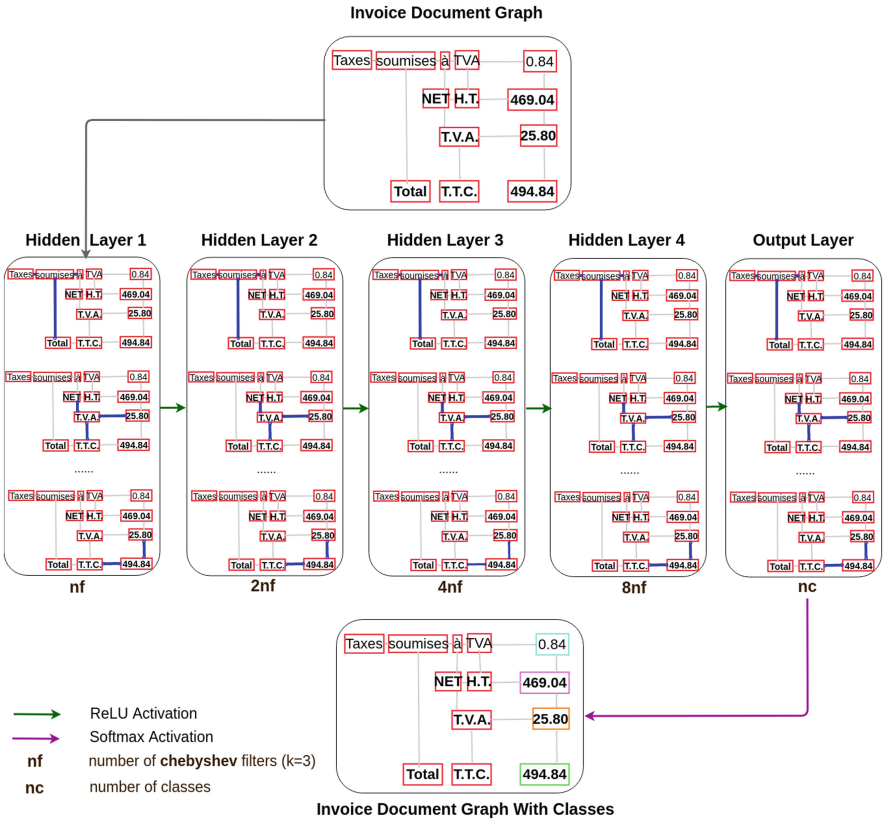

Fig. 4. Proposed 5 layer GCN architecture for graph node classification. Input document graph is passed through 4 hidden layers, each followed by ReLU (shown in green arrows) for non-linearity. We start initially with 16 Chebyshev filters (nf = 16). At last layer, we use the Softmax activation (shown in purple arrow) for classifying each word node into one of 28 classes (nc = 28). Predicted classes are shown in different colors. (Color figure online)

where we set $K = 3$ in the experiments to keep each convolution at most 3-steps away from a center vertex. Our GCN architecture contains 4 layers of graph convolutions with filters increasing by a factor of 2 per layer, starting from 16 filters in 1^{st} layer, each followed by a Rectified Linear Unit (ReLU) activation to increase non-linearity. The output layer is passed through the Softmax activation function which provides a class label to each word of the document. We have 27 classes of interest plus an "undefined" class for not important words. For multi-class classification, we evaluate the cross-entropy error over all the words (as we have labels of all words in supervised training). Dataset has n number of invoice documents and each document has its own graph with labels and no document is linked to other documents in the graph. We perform batch gradient descent using

the full dataset for every training iteration, i.e. we feed n non-linked independent graphs together in each iteration.

Finally, to extract entities, we group words belonging to the same entity class like all seller address class words, all shipping address class words, etc., to form entities while checking through some parsers and conditionals for date, invoice number, etc. to form entities and we follow left to right ordering for it.

3 Experiments

We perform the experiment of entity extraction as an entity classification in an invoice where entity could be in or outside table.

3.1 Dataset and Metric

We have a private dataset of 3100 invoices which is accumulated as a result of collaboration with a company. 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 from one of the 27 entity classes to extract (see Table 2 for detailed class types) plus an undefined class for not important words.

We use Tesseract OCR [15] for text extraction. Each word in the invoice is identified by its word id, location, content and ground truth class. Note that not all invoices had all the entities class present. Also, we have a fixed classes to extract inside a table like *product description*, *unit price*, *quantity*, *total*, etc. and if some other columns are present, then we label them *undefined*. This will be further illustrated with examples in coming sections.

For the experiment, we split the invoice dataset into a training, validation and testing set randomly, using 50%, 20% and 30% respectively. We also ensured that the entity class distribution remains the same in all the three sets. We intentionally kept only 50% for training and big set of 30% for testing because we want to see how well the GCN is able to generalize even with a small training set.

We measure the performance at a very strict way upto the word level. Even after the entities are extracted, the classification errors at the word level help us to point out where exactly the system is lacking and improvement is needed. This way, we can focus on improving that particular class of entity. We compare the predicted entity classes of words with their ground truth classes. We provide the performance per class in terms of precision, recall and F-measure. The overall system performance is the micro average precision, recall and F-measure as it unbiased in multi-class classification.

3.2 Experimental Setup

We train our 5 layer GCN model in a supervised scheme on the architecture described in Sect. 2.4 and evaluate on a test set of 930 randomly chosen invoices. We use the Chebyshev filters of order 3 ($k = 3$) and initial number of filters are 16 ($nf = 16$). The number of classes are 28 ($nc = 28$). We use the L2

regularization factor of 5.10^{-4} for the GCN layer and number of hidden units but we do not use any dropout. We train the model as a single big graph batch for a maximum of 2000 epochs using the Adam optimizer [16] with a learning rate of 0.001 and early stopping with a window size of 50, i.e. we stop training if the validation loss does not decrease for 50 consecutive epochs.

3.3 Results and Discussion

We present in Table 2 the extracted entity results for 27 classes of interest. Before analyzing the system, let us first look at the running time of the system as shown in Table 1. We can see that the OCR (in our case Tesseract OCR) took a big time of 3.5 s. The most time taking step is Feature Calculator because in this step, we have to calculate essential word features like nature, isZipCode, isKnownCity, etc. Also, fetching word embedding from BPE for each word of the image is time taking. We can see that core of the system, i.e., graph modeling and essentially graph node classifying is very quick and takes less than a second. The overall average time for an image to process is 15 s which still needs a big improvement.

Table 1. Running time of our system.

Step	Average running time (in seconds)
Word extractor	3.5
Feature calculator	9.8
Graph modeler	0.8
Graph node classifier	0.9
Total running time	15

The results can be analyzed in 4 broad categories. Words of each invoice falls in one of these 4 categories (refer to Table 2): General invoice entities (rows 1 to 4), company information entities (rows 5 to 13), client information entities (row 14 to 18) and table information entities (rows 19 to 27).

Let us analyze the results category-wise:

1. **General Invoice Entities:** Invoice number and invoice date are very well extracted with F1 0.95 and 0.90, even when sometimes the words are split into 2 or 3 parts (see Fig. 5). Payment mode has good recognition of 0.94 as it is usually preceded by few words and our GCN model easily captures it. Order number however has little low F-measure of 0.80 because it is usually confused with client number and is present very low in the invoices as shown in Fig. 5.
2. **Company Information Entities:** We provide detailed information about company as shown in examples of Fig. 6. Company address is recognized very well with 0.94 F1 score while company name is recognized with a lower 0.84 F1 score because many times the company name is recognized as company

address as they follow together. This infact is not a big error as company name can be part of address. Company identifiers like siret number, vat number, ape code(type of activity a company does) are extracted with more than 0.91 F1 score.

The reason of company siren number score to be 0.86 is that it is usually confused with siret number and they do share similar characteristics in a very condensed context. Company registration city is one of the lowest present entities in the invoices, still a good recognition is obtained. Phone number and fax are recognized very well with almost 0.90 F1 score even though they are quite close in structure.

3. **Client Information Entities:** Like company information, we provide indepth information about the invoice client. Client number is usually confused with order number as discussed above. We provide separate classes for client shipping and billing details as shown in examples of Fig. 7. We assume that when only one client address or client name is present, then it is billing name or billing address. When two client addresses are there, then it is very difficult to distinguish which address is shipping or which address is billing. The billing and shipping address words are mostly similar, i.e., they contain a proper name, a street number and name, a city name, zipcode and sometimes the country name. The only distinguishing feature is the header of addresses. Shipping and billing addresses have some specific headings (see (a) and (c) in Fig. 7). Our model automatically learns these heading representations and propagates this information to other nodes of the graph locally. From top of address like shipping name to bottom of address like country. The result clearly shows that our model extracts both shipping and billing entities very well with over 0.90 F1 score, depicting its power to segregate very closely related multi node entities through neighborhood features learned via convolutions.
4. **Extracted Table Entities:** We can easily observe through Table 2 that our model performs excellent table extraction. System like CloudScan [8] fails to deal with table entities as their approach was heavily dependent on linear neighborhood due to n-grams. Due to excellent feature calculation for price with nature like number with decimal, real values, real values with currency, currency, etc. as discussed in Sect. 2.2, we had a very specific feature representation for various table price related entities. Further, the 4 nearest graph modeling further connected these prices to form a structure. With extraction results of over 0.98 F1 score for table price entities (unit price, price without tax, tax rate, total without tax, total tax amount, net payable amount) (refer to Fig. 8), it is clear that our model is extremely good for table extraction. Further product description which contains different kinds of words like model numbers, guarantee, etc., also has a very good score of 0.95 F1.

The micro averaged overall invoice entity extraction performance of our system is excellent with 0.93 F1 score.

Table 2. Performance of our model on various classes to extract.

Row	Extracted entity	F1	Precision	Recall
1	Invoice number	**0.90**	**0.92**	0.88
2	Invoice date	**0.95**	**0.94**	**0.96**
3	Order number	0.80	0.82	0.78
4	Payment mode	**0.94**	**0.95**	**0.93**
5	Company name	0.85	0.87	0.84
6	Company address	**0.94**	**0.94**	**0.94**
7	Company siren number	0.86	0.87	0.86
8	Company siret number	**0.91**	**0.91**	**0.92**
9	Company vat number	**0.94**	**0.94**	**0.95**
10	Company APE code	**0.95**	**0.95**	**0.95**
11	Company registration city	0.82	0.88	0.77
12	Company phone number	0.89	**0.90**	0.89
13	Company fax number	**0.90**	**0.90**	**0.90**
14	Client number	0.79	0.81	0.79
15	Client billing name	**0.91**	**0.91**	**0.92**
16	Client billing address	**0.90**	**0.90**	**0.91**
17	Client shipping name	**0.93**	**0.94**	**0.92**
18	Client shipping address	**0.90**	**0.91**	0.89
20	Product serial number	0.87	0.87	0.87
21	Product description	**0.95**	**0.95**	**0.96**
22	Product unit price	**0.99**	**0.99**	**0.99**
23	Product Quantity	**0.92**	**0.91**	**0.93**
24	Product price without tax	**0.98**	**0.98**	**0.98**
25	Tax rate	**0.99**	**0.99**	**0.99**
26	Total without tax	**0.98**	**0.99**	**0.98**
27	Total tax amount	**0.99**	**0.99**	**0.99**
28	Net payable amount	**0.99**	**0.99**	**0.99**
Micro average		**0.93**	**0.93**	**0.929**

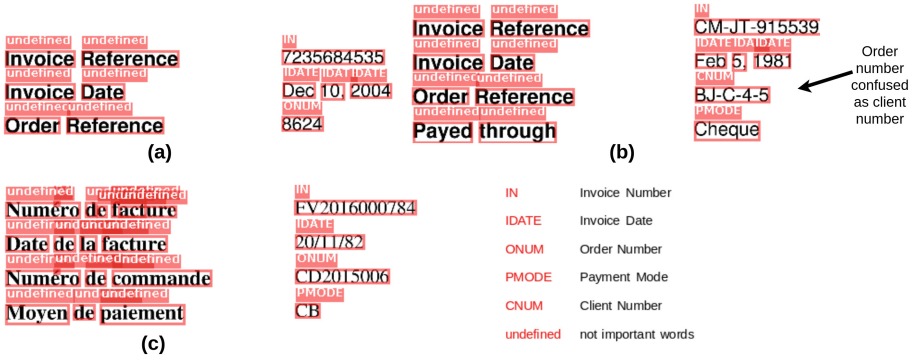

Fig. 5. Example: general invoice entities extraction.

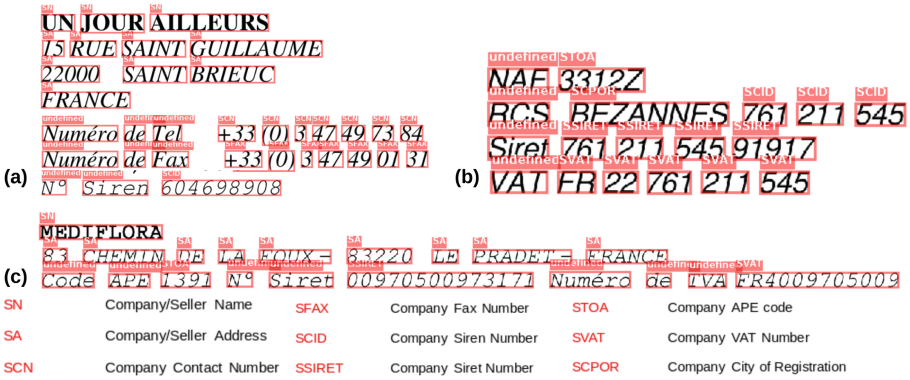

Fig. 6. Example: company entities extraction

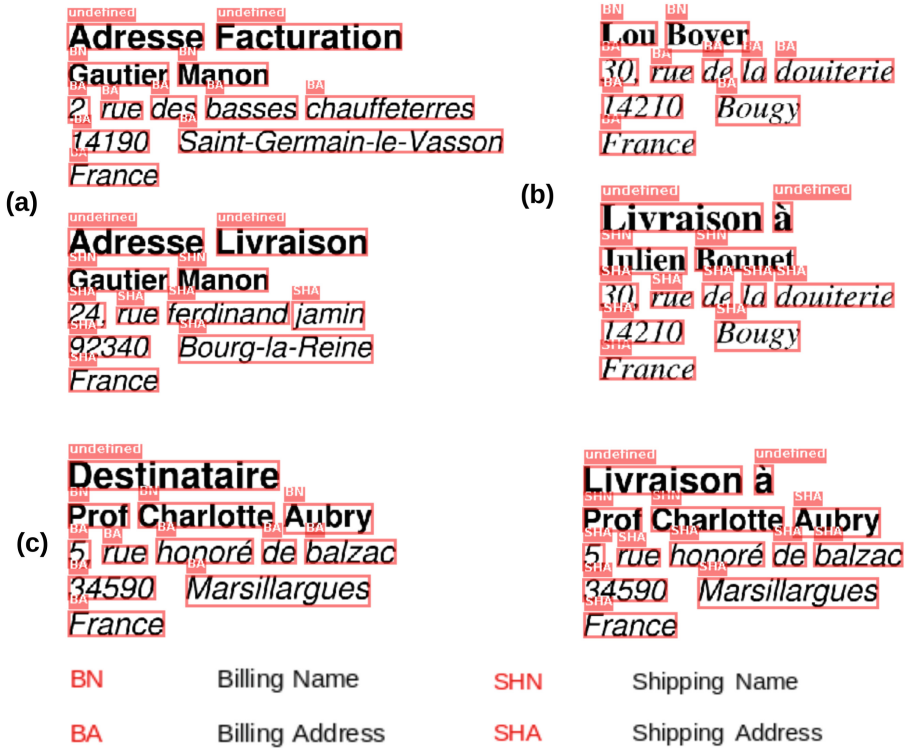

Fig. 7. Example: client entities extraction

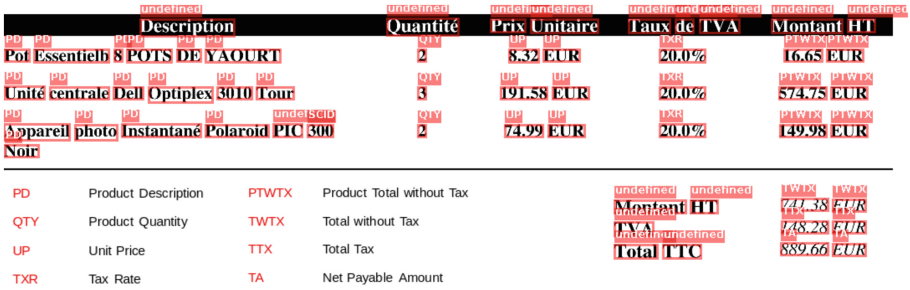

Fig. 8. Example: table extraction

4 Conclusion

In this paper, we proposed a novel and generic approach to extract invoice entities from printed invoice documents. We proved that our approach using localized Graph Convolutional Networks is template independent and effective. It is a complete invoice reading system which extracts entities both inside and outside the table. We extracted 27 very fine entities from the document with an excellent extraction rate of 0.93 F1 score, which is also the best score so far in any IAS available. It is unfortunate that we cannot directly compare our results with the existing systems as there is still no public dataset available in invoices due to their sensitive nature. We sincerely wish such a dataset will be published soon and it would drive the field forward significantly. Unfortunately, we cannot release our own dataset due to privacy restrictions. We showed through various categories of result that our system is able to model entity relations very effectively and is able to generalize the whole invoice document through a graph structure. In our future works, we would like to focus more on the architecture of graph convolutional network, on pre-processing of mobile captured image, on improving the running time of the system, on post processing and on incorporating user feedback in the system.

References

1. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvist. Invest. **30**, 3–26 (2007)
2. Schuster, D., et al.: Intellix-end-user trained information extraction for document archiving. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 101–105. IEEE (2013)
3. Rusinol, M., Benkhelfallah, T., Poulain dAndecy, V.: Field extraction from administrative documents by incremental structural templates. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 1100–1104. IEEE (2013)
4. Hamza, H., Belaïd, Y., Belaïd, A.: A case-based reasoning approach for invoice structure extraction. In: Ninth International Conference on Document Analysis and Recognition, ICDAR 2007, vol. 1, 327–331. IEEE (2007)

5. Dengel, A.R., Klein, B.: smartFIX: a requirements-driven system for document analysis and understanding. In: Lopresti, D., Hu, J., Kashi, R. (eds.) DAS 2002. LNCS, vol. 2423, pp. 433–444. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45869-7_47
6. Cesarini, F., Francesconi, E., Gori, M., Soda, G.: Analysis and understanding of multi-class invoices. Doc. Anal. Recogn. **6**, 102–114 (2003)
7. d'Andecy, V.P., Hartmann, E., Rusiñol, M.: Field extraction by hybrid incremental and a-priori structural templates. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 251–256. IEEE (2018)
8. Palm, R.B., Winther, O., Laws, F.: Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 406–413. IEEE (2017)
9. Kasar, T., Bhowmik, T.K., Belaid, A.: Table information extraction and structure recognition using query patterns. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 1086–1090. IEEE (2015)
10. Santosh, K., Belaïd, A.: Document information extraction and its evaluation based on client's relevance. In: 2013 12th International Conference on Document Analysis and Recognition (ICDAR), pp. 35–39. IEEE (2013)
11. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: 54th Annual Meeting of the Association for Computational Linguistics, pp. 1715–1725 (2016)
12. Heinzerling, B., Strube, M.: BPEmb: tokenization-free pre-trained subword embeddings in 275 languages. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki (2018)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
14. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844–3852 (2016)
15. Smith, R.: An overview of the tesseract OCR engine. In: Ninth International Conference on Document Analysis and Recognition, ICDAR 2007, vol. 2, pp. 629–633. IEEE (2007)
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)