End-to-End Extraction of Structured Information from Business Documents with Pointer-Generator Networks

Clément Sage[1, 2], **Alex Aussem**[1], **Véronique Eglin**[1], **Haytham Elghazel**[1], **Jérémy Espinas**[2]

[1]Univ Lyon, CNRS, LIRIS, France
{csage, aaussem, veglin, helghazel}@liris.cnrs.fr
[2]Esker, France
{clement.sage, jeremy.espinas}@esker.fr

Abstract

The predominant approaches for extracting key information from documents resort to classifiers predicting the information type of each word. However, the word level ground truth used for learning is expensive to obtain since it is not naturally produced by the extraction task. In this paper, we discuss a new method for training extraction models directly from the textual value of information. The extracted information of a document is represented as a sequence of tokens in the XML language. We learn to output this representation with a pointer-generator network that alternately copies the document words carrying information and generates the XML tags delimiting the types of information. The ability of our end-to-end method to retrieve structured information is assessed on a large set of business documents. We show that it performs competitively with a standard word classifier without requiring costly word level supervision.

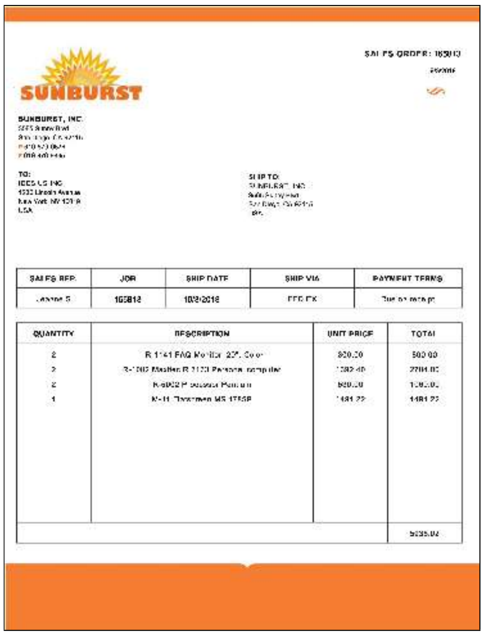
1 Introduction

Companies and public administrations are daily confronted with an amount of incoming documents from which they want to extract key information as efficiently as possible. They often face known types of documents such as invoices or purchase orders, thus knowing what information types to extract. However, layouts are highly variable across document issuers as there are no widely adopted specifications constraining the positioning and textual representation of the information within documents. This makes information extraction a challenging task to automate.

In addition to the incremental approaches based on layout identification (d'Andecy et al., 2018; Dhakal et al., 2019), a number of recent works have proposed deep neural models to extract information in documents with yet unseen layouts. Following Palm et al. (2017), most of these layout-free approaches resort to classifiers that predict the information type of each document word. Yet, the information extraction task does not offer word level ground truth but rather the normalized textual values of each information type (Graliński et al., 2020). The word labels can thus be obtained by matching these textual values with the document words but this process is either time-consuming if manually performed or prone to errors if algorithmically performed. Indeed, extracted information may not appear verbatim in the document as its textual values are normalized. For example, the value "`2020-03-30`" for the document date field may be derived from the group of words "`Mar 30, 2020`". This forces the development of domain specific parsers to retrieve the matching words. Also, multiple document words can share the textual value of a extracted field while being semantically distinct, hence imposing additional heuristics for disambiguation. Otherwise, a street number may be wrongly interpreted as a product quantity, inducing noise in the word labels.

To the best of our knowledge, Palm et al. (2019) is the only related model that directly learns from naturally produced extraction results. However, the authors only tackle the recognition of independent and non-recurring fields such as the document date and omit the extraction of structured entities. Such entities are structures composed of multiple field values. Within documents, structured information is often contained in tables. For example, a product entity is usually described in a table row with its field values, such as price and quantity, being in different columns. Our work is intended to remedy this lack by proposing end-to-end methods for processing structured information. As a first step towards full end-to-end extraction, we focus in this paper on the recognition of fields whose values always appears verbatim in the document, thus eliminating the need for normalization operations.

As illustrated in Figure 1, extracted structured information can be represented in a markup lan-



SUNBURST

SALES ORDER: 165813

[illegible]

SUNBURST, INC.
[illegible]

TO:
IDES US INC
1530 Lincoln Avenue
New York, NY 10118
USA

SHIP TO:
SUNBURST, INC.
[illegible]

SALES REP.	JOB	SHIP DATE	SHIP VIA	PAYMENT TERMS
Leanne S.	165813	10/2/2018	FED EX	Due on receipt

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
2	R-1141 FAQ Monitor 22", Color	300.00	600.00
2	R-1002 Master R 2120 Personal computer	1352.40	2704.80
2	R-5002 Processor Pentium	620.00	1240.00
1	M-11 Tracemen MS 175SP	1481.22	1481.22
			5935.02

(a) Document

```
<Extraction>
    <Product>
        <IDNumber>R-1141</IDNumber>
        <Quantity>2</Quantity>
    </Product>
    <Product>
        <IDNumber>R-1002</IDNumber>
        <Quantity>2</Quantity>
    </Product>
    <Product>
        <IDNumber>R-5002</IDNumber>
        <Quantity>2</Quantity>
    </Product>
    <Product>
        <IDNumber>M-11</IDNumber>
        <Quantity>1</Quantity>
    </Product>
</Extraction>
```

(b) Extracted information

Figure 1: A purchase order (a) and the XML representation of its extracted information (b). In this example, we retrieve the ordered products which are contained in the main table of the document. Two fields are recognized for each product entity: the ID number and the quantity.

guage that describes both its content and its structure. Among many others, we choose the XML language[1] for its simplicity. We define as many XML tag pairs as the number of entity and field types to extract. A pair of opening and closing field tags delimits a list of words constituting a field instance of the corresponding type.

Following successful applications of sequence-to-sequence models in many NLP tasks (Otter et al., 2020), we employ a recurrent encoder-decoder architecture for outputting such XML representations. Conditioned on the sequence of words from the document, the decoder emits one token at each time step: either a XML tag or a word belonging to a field value. Since field values are often specific to a document or a issuer, extracted information cannot be generated from a fixed size vocabulary of words. Rather, we make use of pointing abilities of neural models (Vinyals et al., 2015) to copy words of the document that carry relevant information. Specifically, we adapt the Pointer-Generator Network (PGN) developed by See et al. (2017) for text summarization to our extraction needs. We evaluate the resulting model for extracting ordered products from purchase orders. We demonstrate that this end-to-end model performs competitively with a word classifier based model while avoiding to create supervision at the word level.

2 Related Work

2.1 Information extraction

As mentioned before, most methods for information extraction in documents take the word labels for granted and rather focus on improving the encoding of the document.

Holt and Chisholm (2018) combine heuristic filtering for identifying word candidates and a gradient boosting decision tree for independently scoring them. The strength of their model mainly lies on the wide range of engineered features describing syntactic, semantic, positional and visual content of each word as well as its local context.

When extracting the main fields of invoices and purchase orders, Palm et al. (2017) and Sage et al. (2019) both employ recurrent connections across the document to reinforce correlations between the class predictions of words. They show empirically that Recurrent Neural Networks (RNN) surpass classifiers whose prediction dependence is only due to local context knowledge introduced in the word representations. For this purpose, they arrange the words within a document as a unidimensional sequence and pass the word representations into a bidirectional LTSM (BLSTM) network for field classification. Similar to the state-of-the-art

[1] https://en.wikipedia.org/wiki/XML

in Named Entity Recognition (Yadav and Bethard, 2018), Jiang et al. (2019) also add a Conditional Random Field (CRF) on top of the BLSTM to refine predictions while extracting information from Chinese contracts.

Yet, unlike plain text, word spacing and alignments in both horizontal and vertical directions convey substantial clues for extracting information of documents. By imposing a spurious unidimensional word order, these architectures significantly favor transmission of context in one direction at the expense of the other. Lately, methods that explicitly consider the two dimensional structure of documents have emerged with two different approaches.

Lohani et al. (2018), Liu et al. (2019) and Holeček et al. (2019) represent documents by graphs, with each node corresponding to a word or a group of words and edges either connecting all the nodes or only spatially near neighbors. Convolutional or recurrent mechanisms are then applied to the graph for predicting the field type of each node.

Some authors rather represent a document page as a regular two dimensional grid by downscaling the document image. Each pixel of the grid contains at most one token - either a character or a word - and its associated representation. Then, they employ fully convolutional neural networks to model the document, either with dilated convolutions (Zhao et al., 2019; Palm et al., 2019) or encoder-decoder architectures performing alternately usual and transposed convolutions (Katti et al., 2018; Denk and Reisswig, 2019; Dang and Thanh, 2019). Finally, all these works except Palm et al. (2019) output a segmentation mask representing the probabilities that each token contained in a pixel of the grid belong to the field types to extract. Katti et al. (2018) and Denk and Reisswig (2019) additionally tackle tabular data extraction by predicting the coordinates of the table rows bounding boxes to identify the invoiced products.

Instead of directly classifying each word of the document, Palm et al. (2019) output attention scores to measure the relevance of each word given the field type to extract. The relevant words are then copied and fed to learned neural parsers to generate a normalized string corresponding to the expected value of the field. The predicted string is measured by exact match with the ground truth. Evaluated on 7 fields types of invoices, their end-to-end method outperforms a logistic regression based model whose word labels are derived from end-to-end ground truth using heuristics. However, their approach cannot extract structured information such as the invoiced products.

Although there are publicly released datasets for the task of information extraction in documents (Jiang et al., 2019; Huang et al., 2019; Graliński et al., 2020), as far as we know, none of them are annotated to recognize structured data.

2.2 Structured language generation

A number of works prove that neural encoder-decoder models can produce well-formed and well-typed sequences in a structured language without supplying an explicit grammar of the language.

Extending traditional text recognition, some authors transform images of tables (Zhong et al., 2019; Deng et al., 2019) and mathematical formulas (Deng et al., 2017; Wu et al., 2018) into their LaTeX or HTML representations. After applying a convolutional encoder to the input image, they use a forward RNN based decoder to generate tokens in the target language. The decoder is enhanced with an attention mechanism over the final feature maps to help focusing on the image part that is recognized at the current time step.

Neural encoder-decoder architectures have also been used for semantic parsing which aims at converting natural language utterances to formal meaning representations (Dong and Lapata, 2016; Rabinovich et al., 2017). The representations may be an executable language such as SQL and Prolog or more abstract representations like abstract syntax trees. Text being the modality of both input and output sequences, Jia and Liang (2016), Zhong et al. (2017) and McCann et al. (2018) include attention-based copying abilities in their neural model to efficiently produce the rare or out-of-vocabulary words.

3 Approach

We assume that the text of a document is already transcribed before extracting its information. For scanned documents, we employ a commercial Optical Character Recognition (OCR) engine for retrieving the text.

The method we propose for extracting structured information from a document is depicted in Figure 2. The model is derived from the PGN of See et al. (2017) proposed for summarization of news

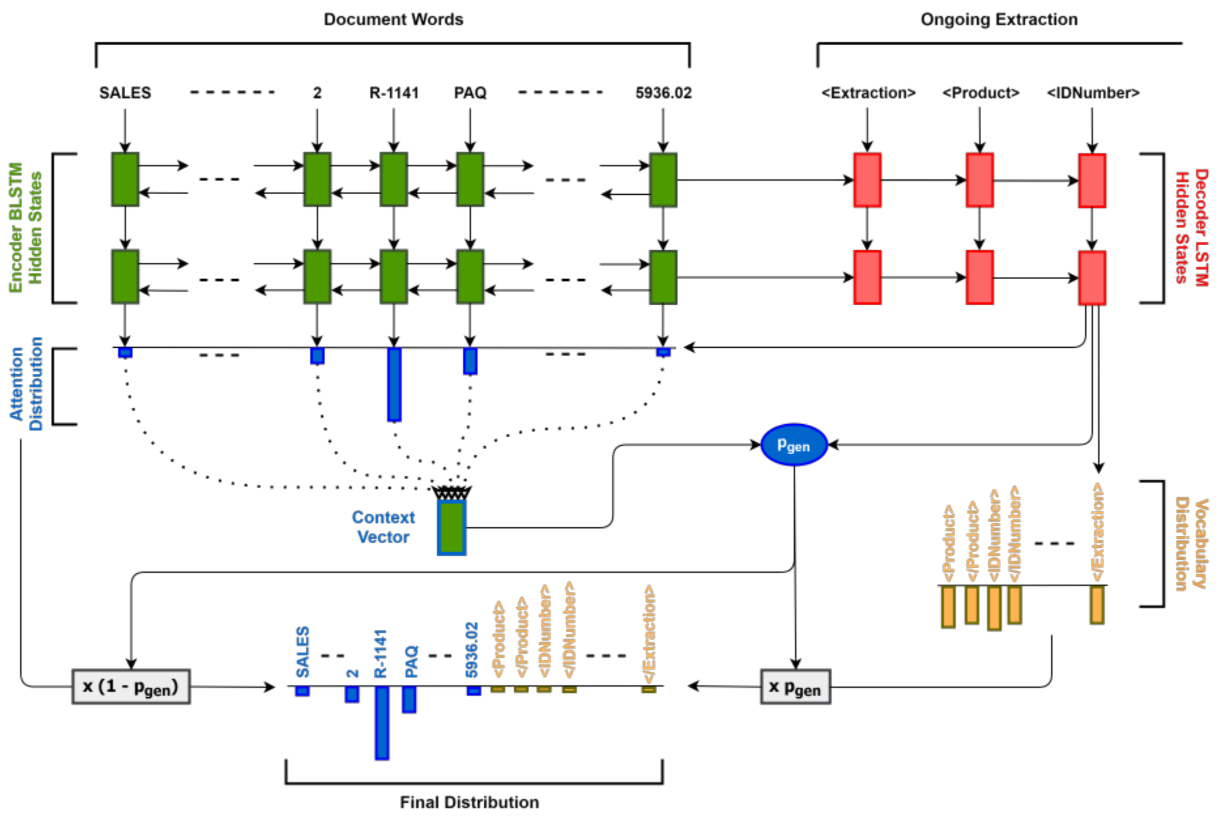


Figure 2: Illustration of the pointer-generator network for extracting structured information of the document in Figure 1. For each decoder time step, a generation probability $p_{gen} \in [0, 1]$ is calculated, which weights the probability of generating XML tags from the vocabulary versus copying words from the document carrying information. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution. For the illustrated time step, the model mainly points to the word `R-1141`, i.e. the ID number of the first product.

articles. The attention-based pointing mechanism allows to accurately reproduce factual information of articles by copying words that are not in the generator's vocabulary, e.g. rare proper nouns. Similarly, we take advantage of its pointing ability to copy the words from the document which carry relevant information while allowing the generator to produce the XML tags which structure the extracted information. In the following subsections, we describe in details our model and highlight key differences with the original PGN.

3.1 Word representation

Each word w_i of the document is represented by a vector denoted r_i. In complement to the word level embeddings used by See et al. (2017), we enrich representations with additional textual features to cope with the open vocabulary observed within the corpus of documents. First, we follow the C2W model of Ling et al. (2015) to form a textual representation q_i^c at the character level. To that end, we apply a BLSTM layer over the dense embeddings associated to the characters of the word and concatenate the last hidden state in both directions. We also add the number n_i of characters in the word and case features, i.e. the percentage α_i of its characters in upper case and a binary factor β_i indicating if it has a title form. We concatenate all these features to form the textual component r_i^t of the word representation:

$$r_i^t = [q_i^w, q_i^c, n_i, \alpha_i, \beta_i] \qquad (1)$$

where q_i^w is its word level embedding.

To take into account the document layout, we also compute spatial features r_i^s of the word. These encompass the coordinates of the top-left and bottom-right edges of the word bounding box, normalized by the height and width of the page. We concatenate the spatial r_i^s and textual r_i^t components to build the word representation r_i.

3.2 Encoder

The words of the document are organized as a unidimensional sequence of length N by reading them in a top-left to bottom-right order. The word representations $\{r_i\}_{i=1..N}$ are then fed to a two-layer BLSTM to obtain contextualized representations through the encoder hidden states $\{h_i\}_{i=1..N}$.

3.3 Decoder

Decoding is performed by a two-layer forward LSTM, producing a hidden state s_t at each time step t. An attention mechanism is added on top of the decoder to compute the attention distribution a^t over the document words and the context vector $h_t^* = \sum_{i=1}^{N} a_i^t h_i$. While See et al. (2017) use the alignment function of Bahdanau et al. (2015), we employ the *general* form of Luong et al. (2015) as this is computationally less expensive while showing similar performances:

$$e_i^t = s_t^\intercal W_a h_i \quad (2)$$
$$a^t = \text{softmax}(e^t) \quad (3)$$

where W_a is a matrix of learnable parameters.

We simplify the computing of the vocabulary distribution P_{vocab} as the generator is only in charge of producing the XML tags and thus has a vocabulary of limited size. We apply a unique dense layer instead of two and do not involve the context vector h_t^* in the expression of P_{vocab}:

$$P_{vocab} = \text{softmax}(V s_t + b) \quad (4)$$

where V and b are learnable parameters

The generation probability $p_{gen} \in [0, 1]$ for choosing between generating XML tags versus copying words from the document is computed as follows:

$$p_{gen} = \sigma(w_h^\intercal h_t^* + w_s^\intercal s_t + w_x^\intercal x_t + b_{ptr}) \quad (5)$$

where x_t is the decoder input, vectors w_h, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Then, p_{gen} weights the sum of the attention and vocabulary distributions to obtain the final distribution $P(w)$ over the extended vocabulary, i.e. the union of all XML tags and unique textual values from the document words:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \quad (6)$$

Note that if a textual value appears multiple times in the document, the attention weights of all the corresponding words are summed for calculating its probability of being copied.

During training, the decoder input x_t is the previous token of the ground truth sequence, while in inference mode, the previous token emitted by the decoder is used. An input token is either represented by a dense embedding if the token is a XML tag or by the textual feature set r_i^t of the corresponding words $\{w_i\}$ if the token is copied from the document.

To help the model keeping track of words already copied, we concatenate the previous context vector h_{t-1}^* with the input representation x_t before applying the first decoder LSTM layer (Luong et al., 2015). We also employ the *coverage mechanism* proposed in See et al. (2017) in order to reduce repetitions in the generated sequences. The idea is to combine the attention distributions of the previous time steps in the coverage vector $c^t = \sum_{t'=1}^{t-1} a^{t'}$ to compute the current attention distribution. We adapt their mechanism to our alignment function, thus changing the equation 2 to:

$$e_i^t = s_t^\intercal (W_a h_i + c_i^t w_c) \quad (7)$$

where w_c is a vector of adjustable parameters.

The training loss is the combination of the negative log-likelihood of the target tokens $\{w_t^*\}_{t=1..T}$ and the coverage loss which penalizes the model for repeatedly paying attention to the same words:

$$loss_t = -\log P(w_t^*) + \lambda \sum_{i=1}^{N} \min(a_i^t, c_i^t) \quad (8)$$

$$loss = \frac{1}{T} \sum_{t=1}^{T} loss_t \quad (9)$$

where λ is a scalar hyperparameter.

When the decoding stage is performed, the resulting string is parsed according to the XML syntax to retrieve all the predicted entities and fields of the document.

4 Dataset

We train and evaluate our extraction model on a dataset of real world business documents which unfortunately cannot be publicly released. It consists of 219,476 purchase orders emanated by 17,664 issuers between April 2017 and May 2018. The dataset is multilingual and multicultural even if the documents mainly originate from the U.S. The number of purchase orders per issuer is at least 3 and at most 31, ensuring diversity of document layouts. Training, validation and test sets have distinct issuers to assess the ability of the model to generalize to unseen layouts. They have been constructed by randomly picking 70 %, 10 % and 20 % of the issuers, respectively. More detailed statistics of the dataset are given in the Table 1.

Table 1: Statistics of our dataset.

Training documents	154,450
Validation documents	22,261
Test documents	42,765
Words per document (Avg.)	411
Pages per document (Avg.)	1.52
Product entities per document (Avg.)	3.52
Tokens in output sequence (Avg.)	32.24
Words per ID number instance (Avg.)	1.36
Words per quantity instance (Avg.)	1.00

This dataset comes from a larger corpus of documents with their information extraction results that have been validated by end users of a commercial document automation software. Among all the types of information, we focus on the extraction of the ordered product entities which have two mandatory fields: ID number and quantity. From this corpus, we select the purchase orders whose location in the document is supplied for all its field instances. The knowledge of location comes from a layout-based incremental extraction system and ensures that we perfectly construct the labels for training a word classifier. Since a field instance can be composed of multiple words, we adopt the IOB (Inside, Outside, Beginning) tagging scheme of Ramshaw and Marcus (1999) for defining the field type of each document word.

5 Experiments

Our end-to-end model is compared on this dataset with a baseline extraction method based on a word classifier. This baseline encodes the document as the end-to-end model does, i.e. with the same operations for constructing the word representations r_i and the encoder outputs h_i. On top of the encoder, a dense layer with softmax activation is added with 5 output units. 4 of these units refer to the beginning and continuation of an instance for ID number and quantity fields. The remaining unit is dedicated to the Outside class, i.e. for the document words carrying information that we do not want to extract. The words with a predicted probability above 0.5 for one of the 4 field units are associated with the corresponding class, otherwise we attribute the Outside class. Field instances are then constructed by merging words with beginning and continuing classes of the same field type. Finally, each quantity instance is paired with an ID number instance to form the product entities. To do so, the Hungarian algorithm (Kuhn, 1955) solves a linear sum assignment problem with the vertical distance on the document as the matching cost between two field instances. For our task, this pairing strategy is flawless if the field instances are perfectly extracted by the word classifier.

The model hyperparameters are chosen according to the micro averaged gain on the validation set. The end-to-end model and baseline share the same hyperparameter values, except the number of BLSTM cells in each encoder layer that is fixed to 128 and 256 respectively, to ensure similar numbers of trainable parameters. The input character and word vocabularies are derived from the training set. We consider all observed characters while we follow the word vocabulary construction of Sage et al. (2019) designed for business documents. This results in vocabularies of respectively 5,592 and 25,677 elements. Their embedding has a size of 16 and 32 and are trained from scratch. The BLSTM layer iterating over characters of document words has 32 cells. For all BLSTM layers, each direction has $n/2$ LSTM cells and their output are concatenated to form n-dimensional vectors. The decoder layers have a size of 128 and are initialized by the last states of the encoding BLSTM layers. At inference time, we decode with a beam search of width 3 and we set the maximum length of the output sequence to the number of words in the document. This results in 1,400,908 and 1,515,733 trainable parameters for the PGN and the word classifier.

To deal with exploding gradients, we apply gradient norm clipping (Pascanu et al., 2013) with a clipping threshold of 5. The loss is minimized with the Adam optimizer, its learning rate is fixed to 0.001 the first 2 epochs and then exponentially decreases by a factor of 0.8. We stop the training when the micro gain on the validation set has not improved in the last 3 epochs. As suggested in See et al. (2017), the coverage loss is added to the minimized loss only at the end of training, for one additional epoch. We weight its contribution by setting $\lambda = 0.1$ as the original value of 1 makes the negative log-likelihood loss increase. The batch size is 8 if the model fits on GPU RAM, 4 otherwise.

The experiments are carried out on a single NVIDIA TITAN X GPU. Model training takes from 3 to 10 days for 10 to 15 epochs. Due to the computational burden, the hyperparameters values have not been optimized thoroughly. Besides, we are not able to train the models on documents with more than 1800 words, which amounts to about 4 % of the training set being put aside. Yet, we evaluate the models on all documents of the validation and test sets. The implementation is based on the seq2seq subpackage of TensorFlow Addons (Luong et al., 2017).

6 Results

6.1 Manual post-processing cost

We evaluate the models by measuring how much work is saved by using them rather than manually doing the extraction. For this purpose, we first assign the predicted products of a document to the ground truth entities, then we count the number of deletions, insertions and modifications to match the ground truth field instances from the predicted instances that have been assigned. The modification counter is incremented by one when a predicted field value and its target do not exactly match. For a given field, we estimate the manual post-processing gain with the following edit distance:

$$1 - \frac{\text{\# deletions} + \text{\# insertions} + \text{\# modifications}}{N} \tag{10}$$

where N is the number of ground truth instances in the document for this field. Micro averaged gain is calculated by summing the error counters of ID number and quantity fields and applying equation 10. We select the assignment between predicted and target entities that maximizes the micro gain of the document. To assess the post-processing gains across a set of documents, we sum the counters of each document before using equation 10.

Our evaluation methodology is closely related to Katti et al. (2018). However, they compute metrics independently for each field while we take into account the structure of entities in our evaluation.

We report in Table 2 the results of both extraction models on the test set. We retain the best epoch of each model according to the validation micro gain. All post-processing gains have positive values, meaning that it is more efficient to correct potential errors of models than manually perform the extraction from scratch (in this case, # insertions $= N$ and # deletions $=$ # modifications $= 0$). We note that the performances of the word classifier and PGN are quite similar. Even if its field level gains are a little behind, the PGN slightly surpasses the word classifier for recognizing whole documents. Both models significantly reduce human efforts as the end users do not have any corrections to make for more than 2 out of 3 documents. Besides, the PGN produces sequences that are well-formed according to the XML syntax for more than 99.5 % of the test documents.

Table 2: Post-processing gains when extracting the products from the test documents. *% Perfect* column indicates the percentage of documents perfectly processed by each model.

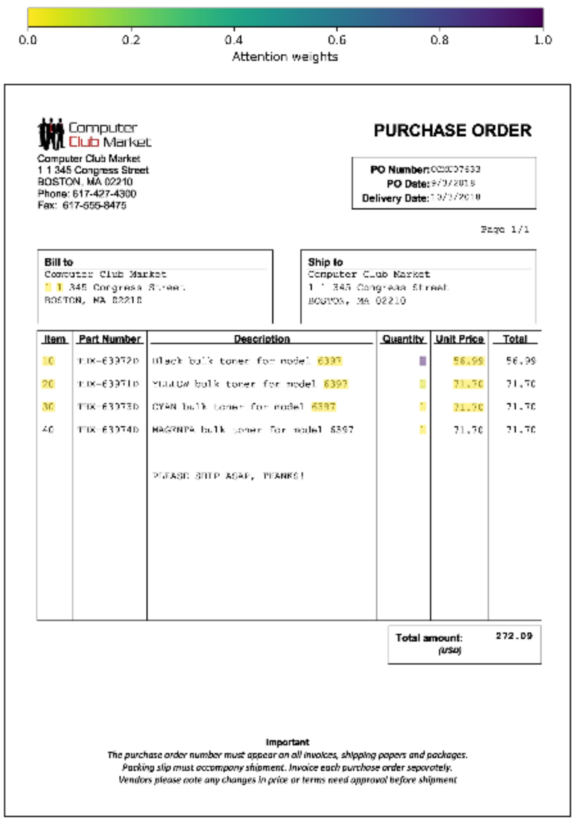
	ID number	Quantity	Micro avg.	% Perfect
Word classifier	0.754	0.855	0.804	67.4
PGN	0.711	0.832	0.771	68.2

6.2 Visual inspection of the attention mechanism

The comparison with the baseline confirms that the PGN has learned to produce relevant attention distributions in order to copy words carrying useful information. In particular, when the expected field value appears multiple times in the document, the PGN is able to localize the occurrence that is semantically correct, as illustrated in the document displayed in Figure 3. As shown, the PGN focuses its attention on the word `1` in the table row of the product that is currently recognized. On the contrary, the model ignores the occurrences of `1` which are contained in the rest of the product table and in the address blocks. This behaviour is noteworthy since the model is not explicitly taught to perform this disambiguation.

7 Discussion

The main difficulty faced by both models is ambiguity in the ground truth as our dataset has been annotated by users from many distinct companies. Some documents contain multiple valid values for a field of a unique product. For example, there may be the references from both recipient and issuer for the ID number. The field value which is retained as ground truth then depends on further processing of the extracted information, e.g. integration into a Enterprise Resource Planning (ERP) system.


0.0
0.2
0.4
0.6
0.8
1.0
Attention weights

Computer Club Market

PURCHASE ORDER

Computer Club Market
1 1 345 Congress Street
BOSTON, MA 02210
Phone: 617-427-4300
Fax: 617-555-8475

PO Number: 000007633
PO Date: 9/7/2018
Delivery Date: 10/7/2018

Page 1/1

Bill to
Computer Club Market
1 1 345 Congress Street
BOSTON, MA 02210

Ship to
Computer Club Market
1 1 345 Congress Street
BOSTON, MA 02210

Item	Part Number	Description	Quantity	Unit Price	Total
10	THX-63972D	Black bulk toner for model 6397	1	56.99	56.99
20	THX-63971D	YELLOW bulk toner for model 6397	1	71.70	71.70
30	THX-63973D	CYAN bulk toner for model 6397	1	71.70	71.70
40	THX-63974D	MAGENTA bulk toner for model 6397	1	71.70	71.70
		PLEASE SHIP ASAP, THANKS!			

Total amount: *(USD)* 272.09

Important
The purchase order number must appear on all invoices, shipping papers and packages.
Packing slip must accompany shipment. Invoice each purchase order separately.
Vendors please note any changes in price or terms need approval before shipment

Figure 3: A sample document with filled bounding boxes around words whose colors depend on their attention weights. For sake of readability, we only highlight the top 15 words. We show attention values for the 6^{th} time step of the pointer-generator network, after having outputted the tokens `<Product>`, `<IDNumber>`, `THX-63972D`, `</IDNumber>` and `<Quantity>`. The model rightly points to the word `1` to extract the quantity value of the first product.

This seriously prevents any extraction model from reaching the upper bound of post-processing gain metrics which is 1.

Besides, the ID number field does not always have a dedicated physical column and rather appears within the description column, without keywords clearly introducing the field instances such as in Figure 1. Also, its instances are constituted on average of more words than the quantity, making less likely the exact match between predicted and target instances. These additional complications explain the gap of model performances between the two fields.

Unlike the word classifier based approach, the PGN tends to repeat itself by duplicating some field instances and skipping others. This is especially observed for documents having a large number of products, therefore large output sequences. To measure the impact of these repetitions on metrics, we split the test set into 3 subsets according to the number of products contained in the document: no more than 3, between 4 and 14 and at least 15 entities. The last subset gathers documents with output sequences of at least 122 tokens. We recompute the metrics for each subset and report the micro averaged gains in Table 3.

Table 3: Micro averaged gains over the test set conditioned on the number N of products in the document.

	$N \leq 3$	$3 < N < 15$	$N \geq 15$
Documents	33,332	7,820	1,613
Product entities	46,893	53,771	44,094
Word classifier	0.804	0.807	0.801
PGN	0.820	0.791	0.696
Without coverage	0.799	0.817	0.671

The performances are stable for the word classifier whatever the number of entities in the document. The PGN is on par with the word classifier for the documents with a small number of products which constitute the vast majority of the dataset. However, its extraction performance greatly declines for large output sequences, indicating that the PGN is more affected by repetitions than the baseline. It is unclear why the coverage mechanism is not as successful on our task as it is for abstractive summarization (See et al., 2017). We also tried to use the temporal attention from Paulus et al. (2018) to avoid copying the same words multiple times but this was unsuccessful too.

8 Conclusion

We discussed a novel method based on pointer-generator networks for extracting structured information from documents. We showed that learning directly from the textual value of information is a viable alternative to the costly word level supervision commonly used in information extraction. In this work, we focused on purchase orders but the approach could be used to extract any structured entity as long as its information type is known at training time.

Future work should aim to: i) reduce repetitions in the output sequences, ii) add parsing abilities into our encoder-decoder in order to transform the values of copied words. This will allow to process fields that need to be normalized when being extracted.

Acknowledgment

The work presented in this paper was supported by Esker. We thank them for providing the dataset on which experiments were performed and for insightful discussions about these researches.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Vincent Poulain d'Andecy, Emmanuel Hartmann, and Marçal Rusiñol. 2018. Field extraction by hybrid incremental and a-priori structural templates. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 251–256. IEEE.

Tuan Anh Nguyen Dang and Dat Nguyen Thanh. 2019. End-to-end information extraction by character-level embedding and multi-stage attentional u-net. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 980–989. JMLR. org.

Yuntian Deng, David Rosenberg, and Gideon Mann. 2019. Challenges in end-to-end neural scientific table recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 894–901. IEEE.

Timo I. Denk and Christian Reisswig. 2019. {BERT}grid: Contextualized embedding for 2d document representation and understanding. In *Workshop on Document Intelligence at NeurIPS 2019*.

Pranjal Dhakal, Manish Munikar, and Bikram Dahal. 2019. One-shot template matching for automatic document data capture. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–6. IEEE.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Filip Graliński, Tomasz Stanisławek, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2020. Kleister: A novel task for information extraction involving long documents with complex layout. *arXiv preprint arXiv:2003.02356*.

Martin Holeček, Antonín Hoskovec, Petr Baudiš, and Pavel Klinger. 2019. Line-items and table understanding in structured documents. *arXiv preprint arXiv:1904.12577*.

Xavier Holt and Andrew Chisholm. 2018. Extracting structured data from invoices. In *Proceedings of the Australasian Language Technology Association Workshop 2018*, pages 53–59.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. 2019. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Zhaohui Jiang, Zheng Huang, Yunrui Lian, Jie Guo, and Weidong Qiu. 2019. Integrating coordinates with context for information extraction in document images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 363–368. IEEE.

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2d documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4459–4469.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 2019. Graph convolution for multimodal information extraction from visually rich documents. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 32–39, Minneapolis, Minnesota. Association for Computational Linguistics.

Devashish Lohani, A Belaïd, and Yolande Belaïd. 2018. An invoice reading system using a graph convolutional network. In *Asian Conference on Computer Vision*, pages 144–158. Springer.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Daniel W Otter, Julian R Medina, and Jugal K Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*.

Rasmus Berg Palm, Florian Laws, and Ole Winther. 2019. Attend, copy, parse end-to-end information extraction from documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 329–336. IEEE.

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

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Clément Sage, Alex Aussem, Haytham Elghazel, Véronique Eglin, and Jérémy Espinas. 2019. Recurrent Neural Network Approach for Table Field Extraction in Business Documents. In *International Conference on Document Analysis and Recognition, ICDAR 2019*, Sydney, Australia.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. 2018. Image-to-markup generation via paired adversarial learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 18–34. Springer.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158.

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

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. 2019. Image-based table recognition: data, model, and evaluation. *arXiv preprint arXiv:1911.10683*.