Deep learning for information extraction from business documents

Clément Sage

N°d'ordre NNT :
2021LYSE1172

THESE de DOCTORAT DE L'UNIVERSITE DE LYON

opérée au sein de

l'Université Claude Bernard Lyon 1

Ecole Doctorale N°512
Informatique et Mathématiques de Lyon

Spécialité de doctorat :
Informatique

Soutenue publiquement le 01/10/2021, par :

Clément SAGE

Deep learning for information extraction from business documents

Apprentissage profond pour l'extraction de l'information des documents commerciaux

---

Devant le jury composé de :

Paquet Thierry	Professeur, Université de Rouen et Normandie Université	Président
Doucet Antoine	Professeur, La Rochelle Université	Rapporteur
Lemaitre Aurélie	Maître de conférences, Université Rennes 2	Rapporteure
Belaïd Yolande	Maître de conférences, Université de Lorraine	Examinatrice
Faci Noura	Maître de conférences, Université Claude Bernard Lyon 1	Examinatrice
Aussem Alexandre	Professeur, Université Claude Bernard Lyon 1	Directeur de thèse
Eglin Véronique	Professeure, INSA Lyon	Co-directrice de thèse
Elghazel Haytham	Maître de conférences, Université Claude Bernard Lyon 1	Co-directeur de thèse
Bérard Jean-Jacques	Directeur R&D, Esker	Invité
Espinas Jérémy	Responsable industriel, Esker	Invité

Acknowledgments

First, I am deeply grateful to Esker for funding this PhD thesis and to all its collaborators that helped me during these three intense years. Particularly, I would like to thank Jean-Jacques Bérard and Cédric Viste for having proposed this thesis and let me conduct my research with complete freedom.

I also thank Thibault Douzon for the scientific exchanges we had and for the help to conduct the experiments of the last chapter.

Last but not least, I would like to offer my special thanks to my industrial supervisor, Jérémy Espinas, for having been my travel companion to present our work at Sydney, for his continuous support and for the proofreading of my scientific production.

I would like to extend my sincere thanks to Alexandre Aussem, Véronique Eglin, Haytham Elghazel for being my academic supervisors. Their invaluable advice and plentiful experience have allowed me to greatly improve my research.

I would also like to thank Antoine Doucet and Aurélie Lemaitre for accepting to be the rapporteurs of my PhD manuscript.

Additionally, I would like to express gratitude to Thierry Paquet, Noura Faci and Yolande Belaïd for accepting to belong to my defense jury and also to my monitoring committee for the two former.

I also thank Olivier Commowick for publicly providing this beautiful thesis template[1].

My appreciation finally goes out to my family, girlfriend and friends for their encouragement and support all through my studies.

[1] https://olivier.commowick.org/thesis_template.php

Abstract

Due to the massive and increasing amount of documents received each day and the number of steps to process them, the largest companies have turned to document automation software for reaching low processing costs. One crucial step of such software is the automatic extraction of information from the documents, particularly retrieving fields that repeatedly appear in the incoming documents. To deal with the variability of structure of the information contained in such documents, the industrial and academic practitioners have progressively moved from rule-based methods to machine and deep learning models for performing the extraction task. The goal of this thesis is to provide methods for learning to extract information from business documents.

In the first part of this manuscript, we embrace the sequence labeling approach by training deep neural networks to classify the information type carried by each token in the documents. When provided perfect token labels for learning, we show that these token classifiers can extract complex tabular information from document issuers and layouts that were unknown at the model training time. However, when the token level supervision must be deduced from the high-level ground truth naturally produced by the extraction task, we demonstrate that the token classifiers extract information from real-world documents with a significantly lower accuracy due to the noise introduced in the labels.

In the second part of this thesis, we explore methods that learn to extract information directly from the high-level ground truth at our disposal, thus bypassing the need for costly token level supervision. We adapt an attention-based sequence-to-sequence model in order to alternately copy the document tokens carrying relevant information and generate the XML tags structuring the output extraction schema. Unlike the prior works in end-to-end information extraction, our approach allows to retrieve any arbitrarily structured information schemas. By comparing its extraction performance with the previous token classifiers, we show that end-to-end methods are competitive with sequence labeling approaches and can greatly outperform them when their token labels are not immediately accessible.

Finally, in a third part, we confirm that using pre-trained models to extract information greatly reduces the needs for annotated documents. We leverage an existing Transformer based language model which has been pre-trained on a large collection of business documents. When adapted for an information extraction task through sequence labeling, the language model requires very few training documents for attaining close to maximal extraction performance. This underlines that the pre-trained models are significantly more data-efficient than models learning the extraction task from scratch. We also reveal valuable knowledge transfer abilities of this language model since the few-shot performance is improved when learning beforehand to extract information on another dataset, even if its targeted fields differ from the initial task.

Résumé

En raison de la quantité massive et croissante de documents reçus chaque jour et du nombre d'étapes pour les traiter, les plus grandes entreprises se sont tournées vers des logiciels d'automatisation des processus documentaires afin d'atteindre de faibles coûts de traitement. Une étape cruciale d'un tel logiciel est l'extraction de l'information des documents, en particulier la récupération des champs qui apparaissent régulièrement dans les documents entrants. Pour faire face à la variabilité de la structure de l'information contenue dans ces documents, les systèmes industriels et académiques sont progressivement passés de méthodes basées sur des règles à des modèles d'apprentissage profond pour effectuer la tâche d'extraction. L'objectif de cette thèse est d'apporter des méthodes pour apprendre à extraire l'information des documents commerciaux.

Dans la première partie de ce manuscrit, nous adoptons l'approche d'étiquetage de séquence en entraînant des réseaux de neurones profonds à classer le type d'information porté par chaque *token* des documents. Lorsque les étiquettes des *tokens* utilisées pour l'apprentissage sont parfaites, nous montrons que ces classificateurs de *tokens* peuvent extraire des champs tabulaires complexes de documents dont l'émetteur et la mise en page étaient inconnues au moment de l'apprentissage du modèle. Cependant, lorsque la supervision au niveau du *token* doit être déduite de la vérité terrain de haut niveau naturellement produite par la tâche d'extraction, nous démontrons que les classificateurs de *tokens* extraient l'information de documents du monde réel avec une précision nettement inférieure en raison du bruit introduit dans les étiquettes.

Dans la deuxième partie de cette thèse, nous explorons des méthodes qui apprennent à extraire de l'information directement à partir de la vérité terrain de haut niveau à notre disposition, évitant ainsi une supervision au niveau des *tokens* coûteuse. Nous adaptons un modèle séquence à séquence basé sur un mécanisme d'attention afin de copier les *tokens* du document portant de l'information pertinente et de générer les balises XML structurant le schéma d'extraction en sortie. Contrairement aux travaux antérieurs en extraction d'information de bout en bout, notre approche permet de retrouver n'importe quel schéma d'information, quelle que soit sa structure. En comparant ses performances d'extraction avec les classificateurs de *tokens* précédemment étudiés, nous montrons que les méthodes de bout en bout sont compétitives avec les approches d'étiquetage de séquence et peuvent largement les surpasser lorsque les étiquettes des *tokens* ne sont pas immédiatement accessibles.

Enfin, dans une troisième partie, nous confirmons qu'utiliser des modèles pré-entraînés pour extraire de l'information réduit considérablement les besoins en documents annotés. Nous exploitons un modèle de langage existant basé sur l'architecture Transformer qui a été pré-entraîné sur une large collection de documents commerciaux. Lorsqu'il est adapté à une tâche d'extraction d'information via l'approche d'étiquetage de séquence, le modèle de langage nécessite très peu de documents d'entraînement pour atteindre des performances d'extraction proches du maximum. Cela souligne que les modèles pré-entraînés

sont significativement plus efficients en matière de données que les modèles apprenant la tâche d'extraction à partir de zéro. Nous révélons également de précieuses capacités de transfert de connaissances pour ce modèle de langage puisque les performances sont améliorées en apprenant au préalable à extraire de l'information sur un autre jeu de données, même si ses champs ciblés diffèrent de la tâche initiale.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence . . . 19

ANN Artificial Neural Network . . . 26

BERT Bidirectional Encoder Representations from Transformers . . . 41

BLSTM Bidirectional Long Short-Term Memory . . . 32

CNN Convolutional Neural Network . . . 27

EDI Electronic Data Interchange . . . 2

ERP Enterprise Resource Planning . . . 2

GPU Graphics Processing Unit . . . 69

IE Information Extraction . . . 3

IR Information Retrieval . . . 12

LSTM Long Short-Term Memory . . . 30

ML Machine Learning . . . 19

MLP Multi-Layer Perceptron . . . 29

NER Named Entity Recognition . . . 12

NLP Natural Language Processing . . . 11

OCR Optical Character Recognition . . . 2

OOV Out-Of-Vocabulary . . . 35

PDF Portable Document Format . . . 2

PGN Pointer-Generator Network . . . 81

ReLU Rectified Linear Unit . . . 29

RNN Recurrent Neural Network . . . 29

SGD Stochastic Gradient Descent . . . 25

XML EXtensible Markup Language . . . 2

VRD Visually Rich Documents . . . 6

List of Notations

Wherever possible, they follow the notation suggestions from Xu et al. (2020d)

$\boldsymbol{x}$	input
d	input dimension
$\mathcal{X}$	input space
$\boldsymbol{y}$	output
d_{o}	output dimension
$\mathcal{Y}$	output space
$\mathcal{Z}$	example space ($= \mathcal{X} \times \mathcal{Y}$)
$\boldsymbol{z}$	example ($\in \mathcal{Z}$)
$\mathcal{D}$	distribution of $\mathcal{Z}$
$\mathcal{H}$	hypothesis space
$\boldsymbol{\theta}$	model parameters
$f_{\boldsymbol{\theta}}$	hypothesis function
$\hat{y}$	model prediction
ℓ	loss function ($\mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$)
n	number of samples
S	sample set ($S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$)
$L_n(\boldsymbol{\theta})$	training loss
σ	activation function ($\mathbb{R} \to \mathbb{R}^+$)
η	learning rate
$^\top$	transpose of a matrix or vector
$\circ$	Hadamard product (element-wise product)

Chapter 1

Introduction

Contents

1.1 Context

The companies daily exchange transactional documents in either paper or native digital formats. These business documents can be purchase receipts, delivery notes, insurance policy documents, vendors contracts, employment agreements, custom declaration forms and a wide many more. In those documents, both layout organization and textual content are critical for the document recognition and understanding. Text is the key information in such documents but cannot be easily serialized into single sequences without losing information.

Practically, in those companies, each received document carefully follows a number of processing steps which depend on its type. As an example, take the case of invoices: this is a document sent by a supplier to a buyer, listing the products or services bought, the debt owed, and when and how to pay this debt. The steps required for processing an invoice are illustrated in the Figure 1.1. When a company receives an invoice, the accountant in charge of invoices must recognize its key data, verify that it matches a previous purchase order or contract, detect and correct any potential error, make the document approved by some superiors and stakeholders before finally paying the supplier.

This sequential process involves multiple employees among the organization, resulting in a high processing cost per invoice. The companies included in the survey of Cohen and York (2020) report an average cost of 10 $ and 8 days to process a single invoice. Besides, the largest companies deal with more than a thousand of invoices per day (iPayables, 2016). This volume and complexity of process impose to use dedicated software systems to

Figure 1.1: **Invoice processing steps: from arrival to payment.** This figure depicts how incoming invoices are typically handled in companies. Image reproduced from Shivalker (2020).

automate the maximum of steps of the cycle and therefore maintain reasonable processing costs. Such document automation software may be standalone like the solutions provided by Esker[1], the company funding this PhD thesis. As offered by SAP[2] for example, a document processing software may also be directly embedded in wider systems that keep track of every operation occurring in a company. These systems are commonly referred to as Enterprise Resource Planning (ERP) systems (Enterprise Ressource Planning) (Chang et al., 2008).

Using these software products, there has been a huge decline of document processing costs in the largest corporations. Yet, they still face some serious challenges when aiming at full automation since only 11 % of the companies reported in Team (2020) process more than 80 % of their invoices without any human intervention.

One of the major obstacle observed for many business document types including invoices is the presence of unstructured document formats, i.e. formats which are meant only for human readers. When received as a paper per mail or fax, a document is first scanned to start its processing by the software system. It results in an image representation of the document, i.e. a raw matrix of pixels, which does not allow for direct access to the text. An Optical Character Recognition (OCR) engine must be employed to retrieve the text but this process is not flawless. Otherwise, when the document is transmitted in a electronic manner, e.g. as an attachment of an email, the text of the document can be perfectly preserved. This is the case of formats such as Portable Document Format (PDF). Yet, there is still the need to extract key information of the document from its heap of words.

There also exist some structured electronic formats which contain a machine-readable schema of the information from the document. We can mention formats like Electronic Data Interchange (EDI), EXtensible Markup Language (XML) or formats generated from web portals. These structured formats allow to automatically extract key information without any error and effort. However, even if they are progressively adopted, such formats are still far from being predominantly used in the business document exchanges.

[1] https://www.esker.com/
[2] https://www.sap.com/

For invoices, according to a yearly conducted survey (Cohen and York, 2020), structured electronic formats have been more exchanged than unstructured formats for the first year only in 2020 and by a short head (50.3 % of documents).

1.2 Problem statement

The business documents contain recurring information types that the recipient may be interested to extract (Cristani et al., 2018). Such an information type is also called a *field*. Some fields are generic and shared by multiple document types, e.g. the document date and the address of the issuer. Others are specific to a document type, e.g. the tax amount included in an invoice. For an incoming document flow, the recipient defines a *schema* gathering all the fields that are supposed to be retrieved. A schema can be viewed as a set of case frames holding the information contained in a document. Take the example of a flow of incoming documents composed of only purchase orders (a purchase order is a document emitted by a client to its supplier to order some products or services). A typical schema for this document flow would look like the Python[3] dictionary given in Figure 1.2. We note that the schema has a particular structure:

- Some fields appear only once such as the order number and date.
- Other fields occur an unknown number of times for a single document. Furthermore, each instance may be attached to an instance of another field, thus forming a structured entity. In this schema, the ID number, description, quantity, unit price and total price fields are grouped together to describe each ordered product.

Some fields are associated with a well-defined and rigid data type, e.g. a date for the issuing date field. Others have rather flexible formats, e.g. the document number is often a sequence of alphanumerical characters but is not constrained to be. Finally, certain fields may be optional, i.e. not always be filled for a document. For instance, the product prices do not always appear in the purchase order as they are sometimes unknown to the sender at issuing time.

Extracting information from a business document comes down to fill the predetermined schema of information. To illustrate the Information Extraction (IE) task, we show a document sample as well as its expected extraction results in Figure 1.3. The schema is filled with a string for each identified field instance. Each string is normalized according to the type of the field, if any. Indeed, in order to be valuable, extracted information must be machine-readable so that it can be used for pursuing automation of document processing. For example, the ISO 8601 format (YYYY-MM-DD)[4] may be chosen to represent the extracted dates. For this purchase order, the document date field is represented by the word `9/3/2018`. Being emitted by a U.S. company, the U.S. date

[3] https://www.python.org/
[4] https://en.wikipedia.org/wiki/ISO_8601

```
{
    "OrderNumber": None,
    "OrderDate": Date,
    "DeliveryDate": Date,
    "CustomerName": None,
    "CustomerAddress": Address,
    "ShippingName": None,
    "ShippingAddress": Address,
    "Total": Amount,
    "Products": [
        {  # first product
            "IDNumber": None,
            "Description": None,
            "Quantity": Amount,
            "UnitPrice": Amount,
            "TotalPrice": Amount
        },
        {  # second product
            "IDNumber": None,
            "Description": None,
            "Quantity": Amount,
            "UnitPrice": Amount,
            "TotalPrice": Amount
        },
        # ... more products
            ]
}
```

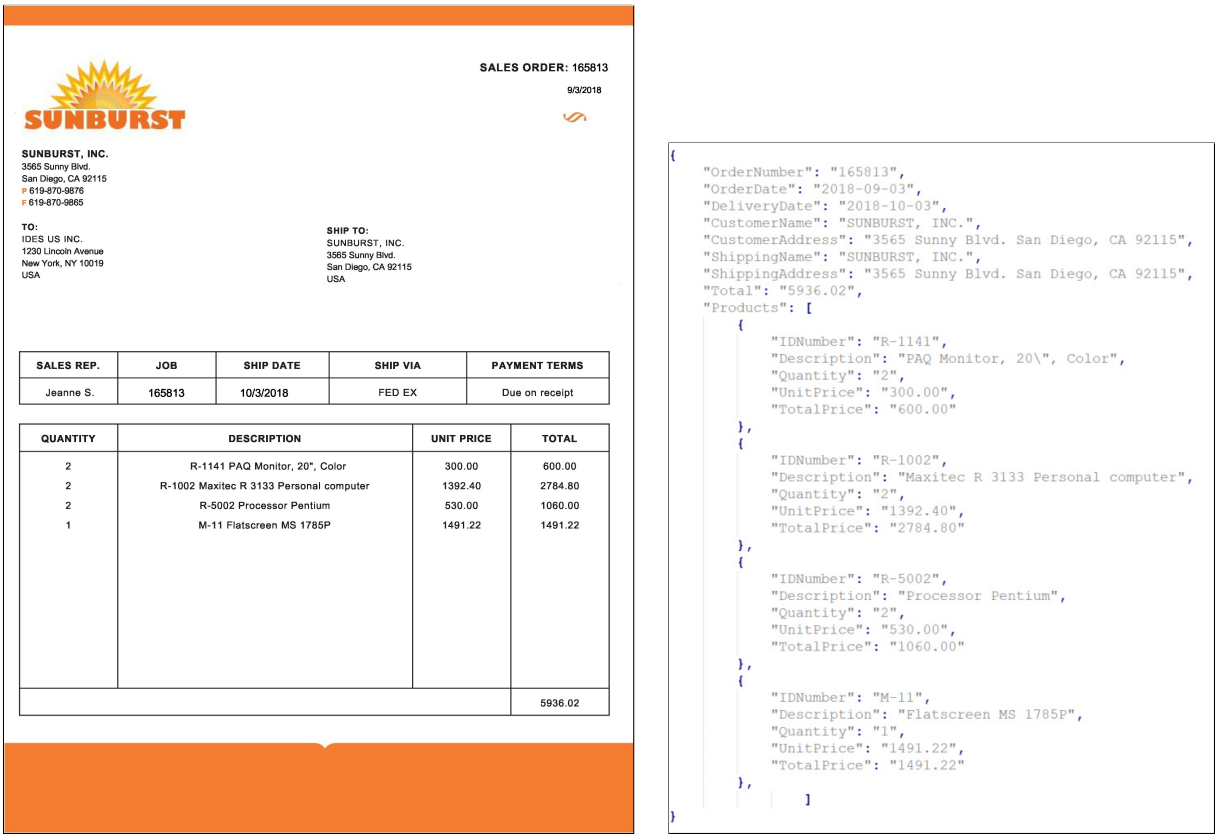
Figure 1.2: **An example of extraction schema for purchase orders.** The keys of this dictionary-style schema correspond to the name of fields to extract while the values inform about their expected type. The *None* value is attributed to the fields that do not have a well-defined type.

notation, i.e. MM/DD/YYYY[5], is employed in this document. This piece of information is therefore normalized to the string `2018-09-03` to fill the schema.

We also note that the ordered products are arranged in a table where each row lists one product entity and the columns contain its different fields. This display is often observed for structured entities in business documents since using tables improves the human readability of structured information. Yet, extracting structured entities goes beyond detecting the tables containing the entities and recognizing their physical structure (Schreiber et al., 2017). Indeed, there is not a one-to-one correspondence between the physical columns and the fields of structured entities in the schema, e.g. in Figure 1.3a the second leftmost column of the table of products gathers the description as well the ID number fields.

While some authors perform information extraction before (Loginov et al., 2020) or at the same time (Zhang et al., 2020a; Wang et al., 2021) as text recognition, the vast majority of IE works assume that the text of documents has already been transcribed before starting to extract their information. Hence, we also make this assumption in this

[5] https://en.wikipedia.org/wiki/Date_format_by_country



SUNBURST

SALES ORDER: 165813

9/3/2018

SUNBURST, INC.
3565 Sunny Blvd.
San Diego, CA 92115
P 619-870-9876
F 619-870-9865

TO:
IDES US INC.
1230 Lincoln Avenue
New York, NY 10019
USA

SHIP TO:
SUNBURST, INC.
3565 Sunny Blvd.
San Diego, CA 92115
USA

SALES REP.	JOB	SHIP DATE	SHIP VIA	PAYMENT TERMS
Jeanne S.	165813	10/3/2018	FED EX	Due on receipt

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
2	R-1141 PAQ Monitor, 20", Color	300.00	600.00
2	R-1002 Maxitec R 3133 Personal computer	1392.40	2784.80
2	R-5002 Processor Pentium	530.00	1060.00
1	M-11 Flatscreen MS 1785P	1491.22	1491.22
			5936.02


```
{
    "OrderNumber": "165813",
    "OrderDate": "2018-09-03",
    "DeliveryDate": "2018-10-03",
    "CustomerName": "SUNBURST, INC.",
    "CustomerAddress": "3565 Sunny Blvd. San Diego, CA 92115",
    "ShippingName": "SUNBURST, INC.",
    "ShippingAddress": "3565 Sunny Blvd. San Diego, CA 92115",
    "Total": "5936.02",
    "Products": [
        {
            "IDNumber": "R-1141",
            "Description": "PAQ Monitor, 20\", Color",
            "Quantity": "2",
            "UnitPrice": "300.00",
            "TotalPrice": "600.00"
        },
        {
            "IDNumber": "R-1002",
            "Description": "Maxitec R 3133 Personal computer",
            "Quantity": "2",
            "UnitPrice": "1392.40",
            "TotalPrice": "2784.80"
        },
        {
            "IDNumber": "R-5002",
            "Description": "Processor Pentium",
            "Quantity": "2",
            "UnitPrice": "530.00",
            "TotalPrice": "1060.00"
        },
        {
            "IDNumber": "M-11",
            "Description": "Flatscreen MS 1785P",
            "Quantity": "1",
            "UnitPrice": "1491.22",
            "TotalPrice": "1491.22"
        },
            ]
}
```


(a) Document (b) Filled extraction schema

Figure 1.3: **Illustration of the Information Extraction (IE) task.** The information is extracted from this purchase order according to the schema given in Figure 1.2. Each extracted field value is normalized in accordance with the data type of the field.

PhD thesis.

Of course, even if it is mostly performed in this domain, the extraction of information based on a predefined schema is not limited to the business documents. For example, Qian et al. (2019) extract patient attributes and drug names from medical reports recording drug-related side effects while Chen et al. (2016) retrieve personal information and educational background from resumes.

1.3 Challenges

Information extraction is an arduous task in Document Image Analysis. There exist multiple challenging points that an IE system would encounter when extracting information from business documents.

1.3.1 Beyond the text modality

The business documents are mostly text-intensive documents, thus the text content conveys a significant portion of the semantics of these documents. Yet, considering only this modality is not sufficient to fully understand a business document. Unlike plain text structured in sentences and paragraphs, the text of business documents is non-uniformly displayed. Firstly, the words are freely distributed across the two dimensions of the document pages, making the spacing between two consecutive words non-regular. This reveals word alignments and blocks of text that give substantial clues for understanding a document. When extracting information, one would generally look for words that are aligned or close to some specific keywords. For example, in the Figure 1.3a, the words `SALES ORDER:` serve as an anchor to identify on its right side the actual document number (`165813`). The alignment and spacing of words play also a significant role in recognizing the physical and logical structure of tables, and thus in helping to extract their main information. Secondly, all the words do not have the same size and shape, e.g. the name of the issuing company and the table headers are generally displayed in a larger font that the rest of the document text. All these positioning and scaling characteristics of the document describe what is called the layout or template of the document (Esser et al., 2012) and must be integrated into the developed IE system for understanding business documents.

Moreover, the image modality also encodes a non-negligible amount of the semantics of documents. It can either be visual clues related to the text like font types and colors or more graphical elements such as logos, table borders, stamps and signatures.

Therefore, besides the text itself, both the layout and image modalities should be analyzed in order to correctly interpret the business documents. This explains why such documents are commonly described as Visually Rich Documents (VRD) (Liu et al., 2019a; Wei et al., 2020).

1.3.2 Class variability

A document class can be defined as a subset of documents which share a similar logical structure of the contained information. It is thus often related to the document layout which corresponds to the physical structure (Bartoli et al., 2010).

Intra-class variability A layout may be rigid, i.e. the textual and graphical elements are located at the same exact position for all documents from this class. The documents with such layouts are called *structured* documents. This is the case of forms which contain pre-printed text and fixed-size frames to fill.

Other documents types may exhibit more flexible layouts with some elements varying in terms of absolute position across the documents of a defined class. We then speak of *semi-structured* documents (Dengel, 2003). For instance, while the header and footer of invoices and purchase orders are relatively static, their body tends to vary across documents of the same class. Indeed, the body of such documents often contain tables

of invoiced or ordered products which are of variable length. This makes the extraction of fields contained in the body, e.g. the product quantities, more complex than for fields located in the header or footer, e.g. the document date.

Inter-class variability Except few cases, like the Chinese government which imposes a nation-wide layout for tax invoices exchanged between its domestic companies (Liu et al., 2019a), there usually does not exist a unique and widely adopted layout for each type of business document. Each issuer is free to generate documents with any layout, resulting in a diversity of positioning of the information. In this case, a document class is generally equivalent to a specific issuer. However, two issuers may share the same logical structure of documents and thus be assimilated to the same class. To illustrate the diversity of layout across the issuers, the Figure 1.4 shows various thumbnails of business documents.

As the number of companies worldwide is large and ever-increasing, an IE system that is deployed in an industrial context must therefore be class agnostic. In other words, the system must perform relatively well when facing document layouts that have not been seen during its design.

1.3.3 Localizing the information in the document

The extraction results, i.e. the filled schemas, only offer *what* is the textual value of the information but not *where* it is located in the document, i.e. which of its words carry the information (Palm et al., 2019). Despite appearances, the knowledge about the positions of field instances is not directly retrievable from their textual values. Indeed, two main factors constitute hurdles to find the matching words.

Normalization of fields Since each extracted field instance has been normalized according to the type of the field, its textual value may not appear verbatim in the document. Some fields might have a large number of formats to represent the same normalized textual value. This is the case for dates and amounts whose formats heavily depend on the language and culture of the document. For example, the document date in Figure 1.3a is extracted from the word `9/3/2018` but it could also have been represented by the words `03 Sept. 18`. This diversity of formats imposes significant domain specific knowledge for retrieving the document words containing the expected information.

Multiple occurrences in the document text Even for the fields that does not require normalization, a field instance may still be hard to localize within the document. The value can appear multiple times in the document, thus not knowing which occurrences are semantically correct. These occurrences may be semantically equivalent in the case of redundancy of the information, e.g. the ID number of a product is sometimes duplicated when the vendor and client share the same reference. However, in the general case, the matching occurrences do not necessarily carry the same information type, e.g. the same integer value may refer to a street number or a product quantity depending on its position

within the document. Once again, it requires some domain specific knowledge to make this disambiguation.

1.4 Outline

The systems introduced in the literature for automatically extracting information from business documents can be roughly divided into two categories (Chiticariu et al., 2013) that are discussed in Chapter 2. The first proposed systems were heavily based on rules which explicit exploit the patterns observed in the documents. Then, machine and deep learning approaches have emerged for tackling IE and are now massively adopted in both academic works and industrial applications. They are more easily adaptable to specific extraction needs and constitute the approaches of choice to deal with the diversity of layouts across document classes. Therefore, all the extraction methods that we proposed in this thesis are based on deep learning.

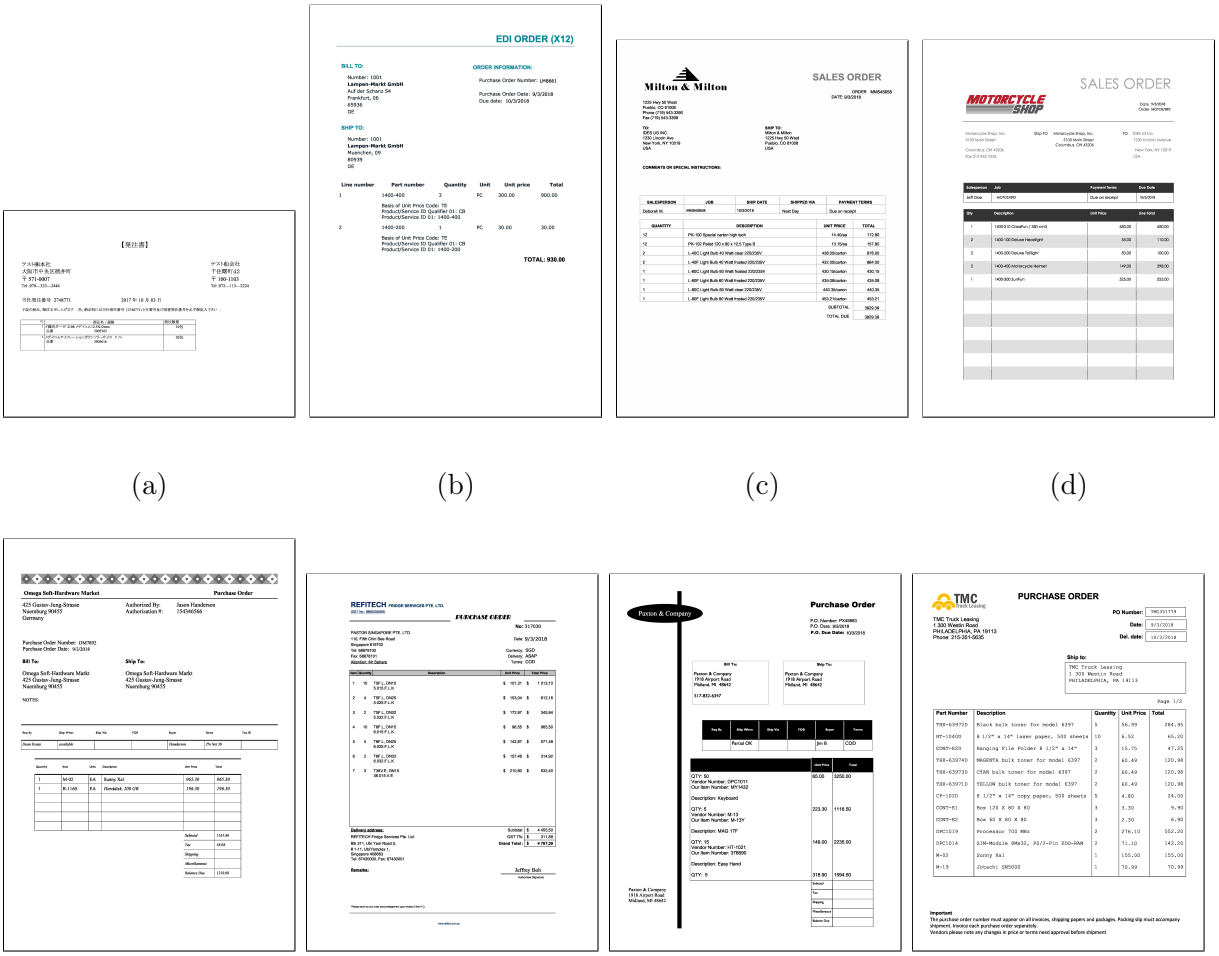
In Chapter 3, we follow the sequence labeling approach which is the historical and still predominant approach for extracting information from business documents with machine learning models. We propose deep models that learn to classify the information type carried by each word in a document. We show that these word classifiers are able to extract tabular information from document layouts that were unknown at the model training time. However, by design, these models impose to know where the expected field instances are located in the document in order to be trained. When such knowledge is not available, it must be deduced from the filled extraction schemas. For the reasons detailed in section 1.3.3, this matching process is prone to errors and thus might introduce noise in the word level supervision. In these conditions, we demonstrate that the word classifiers extract information with a lower accuracy, notably when processing real-world documents.

To overcome the limitations of supervision from sequence labeling approaches, we investigate in Chapter 4 deep models that can learn directly from the results naturally produced by the IE task, i.e. the extraction schemas. To achieve this, we propose a sequence-to-sequence model that is able to map the input sequence of the document words to an output sequence of tokens constituting its filled extraction schema. Unlike the prior end-to-end IE approaches, our method was designed to process any arbitrarily structured schema. By comparing the extraction performance of the sequence-to-sequence model with the previously studied word classifiers, we show that end-to-end methods are competitive with sequence labeling approaches and can greatly outperform them when their word level supervision must be deduced from the extraction schemas of the training documents.

In Chapter 3 and 4, the deep models are trained directly on the information extraction task without prior knowledge about the language and the structure of business documents. While successful when provided enough annotated documents for training, we show in Chapter 5 that the methods, learning from scratch to extract information, are severely impacted in data-constrained settings. To mitigate the decline of performance when

learning IE from few data, we leverage an existing language model that has been pre-trained without any supplementary annotation cost in order to understand the text and layout of business documents. When later adapted to extract information through the sequence labeling approach, the language model is proved to dramatically reduce the needs for annotated documents compared to the methods without pre-training since it requires very few training documents for achieving close to maximal extraction performance.

In Chapter 6, we recap the work done in this thesis and give some perspectives for future work.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 1.4: **Layout diversity across the issuers of purchase orders.** There exist many factors of variability such as the aspect ratio and orientation of the image, the size of the header and footer, table formatting (presence of ruling lines, number and spacing of columns) and many more.

Chapter 2

Related works

Contents

In this chapter, we review the literature about schema-based IE from documents, mainly but not only from business documents. Following Chiticariu et al. (2013), we divide the prior work in IE into rule and pattern based approaches (section 2.1) and machine learning based approaches (section 2.3). We also give notions of machine learning in the section 2.2 in order to better understand the latter IE approaches. We conclude in the section 2.4 by giving a brief overview of the publicly released datasets that are currently used to evaluate extraction models.

2.1 Rule-based information extraction

Information extraction refers to the automatic extraction of structured information from unstructured sources of text (Sarawagi, 2008). While the field now involves more research communities like the document analysis and databases communities, IE takes its roots in the Natural Language Processing (NLP) community, back in the mid 1960's. At that time, two long-term academic research projects (Sager, 1981; Schank and Colby, 1973) launched IE as an autonomous area of research. In the 1980's, commercial IE systems began to appear, one of the first being ATRANS from Lytinen and Gershman (1986) which was dedicated to extract financial information from money transfer messages exchanged between banks. For analyzing the text, the ATRANS system employed a knowledge-based approach with a hierarchy of "scriptal" lexicons that are activated upon the context to

solve lexical ambiguities, e.g. *Credit* may be a synonym of *Pay* or belong to the bank *Credit Lyonnais* depending on the nearby words. After human verification, the extracted information about the actors and the transaction amount serves to initiate automatic money transfers.

The field grew very rapidly from the late 1980's when DARPA, the US defence agency, funded and encouraged research in IE by organizing a series of seven Message Understanding Conferences (MUC) from 1987 to 1998 (Grishman and Sundheim, 1996). The conferences were hosting competitions of information extraction from a wide variety of sources, from military reports of fleet operations during the first two editions to news reports of terrorist attacks, corporate management succession events and airplane crashes in the subsequent editions. The organizers established a quantitative evaluation regime for IE by adapting the precision, recall and F1 score metrics from the Information Retrieval (IR) field (Manning et al., 2008) and by evaluating on data that has not been seen during the system training. According to Weischedel and Boschee (2018), "[t]his was no small accomplishment, because most prior work in natural language processing had not measured progress so rigorously on substantial blind test data." The main task tackled during the MUC conferences was *template filling*, which is roughly equivalent to the schema-based IE problem that we have defined in Chapter 1. In this context, a template was defined as a frame of slots to fill when extracting information of the source documents. Specific rules were associated to the template to detail the instructions for filling the slots. Year after year, the task was made gradually more complex, either in terms of the text corpus complexity and dimensions or the template characteristics (Sundheim, 1993). For the fifth edition (MUC-5), the template had a nested structure and was containing not only categorical data and raw copies of the original text like the name of a company, but also text entries normalized into their canonical form and pointers to other slots of the templates. Facing low performances of proposed systems on the holistic template filling task (Sundheim, 1991), the NLP community started to redefine IE as the study of a collection of smaller and more-constrained tasks. From the sixth edition, two new tasks were introduced, namely Named Entity Recognition (NER) (Nadeau and Sekine, 2007) - which aims at identifying within the source text all the occurrences of names like people and location names and numeric expressions such as times and dates - and coreference resolution (Elango, 2005) - which aims at the identification of noun phrases in the text that referred to the same entities -. This was completed by the relation and event extraction (Pawar et al., 2017) tasks in the follow-up Automatic Content Extraction (ACE) program (Doddington et al., 2004) which took place from 1999 to 2008.

Most of the IE systems proposed in the MUC conferences had emerged from research into rule-based systems in computational linguistics and natural language processing. As a synthesis of the principal methods used during MUC-4, Hobbs (1993) described a generic IE system that consists of a sequence of ten modules that at each step add structure by applying rules. The modules ranged from the detection of text segments to the generation of the filled templates based on the constructed semantic structures and included filters, parsers and lexical disambiguation. Originally geared towards MUC-6, the Large Scale Information Extraction (LaSIE) system from Gaizauskas and Wilks (1998) illustrated in

(a) software	(b) notebook	(c) invoice number	(d) phone
Fireworks 4.0	z800	11-093	+1-734-936-6395
Fireworks MX	z800 AAB	#545465	(313)593-5131
Word 2010	d700 ASE	2008-035465	847.494.5626
Frontpage 2007	z40y	2010.08338	212/949-8058
Antivirus 2007	d50t ATX	2000.348	313/593-5131
[A-Z][a-z]$^{+}$ ([0-9]\.[0-9]|[A-Z]{2}|20[0-9]{2})	(d|z)([0-9]0{2}|[0-9]0[a-z]) ([A-Z]$^{+}$)?	(#[0-9]{6}|(([0-9]{2}|20[0-9]{2})(−|\.)[0-9]$^{+}$)	(\(|\+)?[0-9]{1,3}\)?[0-9]{3}(−|\.|/)([0-9]$^{+}$)(−|\.)?([0-9]$^{+}$)?

Figure 2.1: **Extraction of information through matching of regular expressions.** From top to bottom, this table displays fields, instances of these fields and regular expressions that are likely to extract unknown instances of the same fields. Image reproduced from Brauer et al. (2011).

more details the stages of lexical pre-processing, parsing plus semantic interpretation and discourse interpretation In particular, the system gathered hundreds of rules and many more gazetteer and dictionary entries for performing the Part-Of-Speech (POS) tagging and NER subtasks.

As pointed out by Sarawagi (2008), the IE task is highly dependent of the types of unstructured sources that are processed, and so are the approaches developed to tackle this task. For example, while the sentences and paragraphs in plain text exhibit complex syntactic structures including nouns, pronouns, adjectives, verbs, adverbs and so on, the syntax of business documents generally involves fewer element types, i.e. mostly noun phrases referring to the attributes and values of business entities. However, understanding the syntax of these documents implies to analyze the spatial distribution of the text in the document, i.e. its layout. Therefore, in the rest of this chapter, we focus on IE works in the business domain and domains presenting similarities.

2.1.1 Using only domain knowledge

There are some relatively regular patterns across all business documents and a rule-based approach may leverage these patterns to extract information. One intuitive approach would be to search within the source text if some substrings match one or several predefined regular expressions (Thompson, 1968). Typically, for each field, a set of dedicated regular expressions would be associated. In Figure 2.1, examples of regular expressions are given for extracting various information types. The stricter the syntactical pattern of the field to extract is, the more efficient a regular expression based approach is. For instance, when recognizing the Value Added Tax (VAT) numbers[1] in 39 multilingual invoices, Bureš et al. (2020) achieve perfect extraction with a limited set of regular expressions, presumably one for each country. For example, the Czech VAT identifiers are always composed of `CZ` followed by a block of either 8, 9 or 10 digits. This regular pattern is thus easily cap-

[1]The syntax of European VAT numbers is available at: https://ec.europa.eu/taxation_customs/business/vat/eu-vat-rules-topic/vat-identification-numbers_en

tured by a unique regular expression, namely `CZ[0-9]{8,10}`. For capturing less regular fields, there is usually a trade-off between precision and recall metrics. Permissive regular expressions advantage the recall but at the cost of the precision, e.g. a social security number may be recognized as a phone number if the matching scope of an expression is too broad (Li et al., 2008). To improve precision, further domain heuristics can be apply to the set of strings that match a regular expression. For instance, a regular expression may recognize a few dates in a purchase order. To extract its due date, i.e. the customer's deadline for receiving the ordered products, a heuristic simple as choosing the furthest date in the future may be sufficient to remove the ambiguity with other dates present in the document such as the issue date. In some cases, one may leverage the context of the matched strings by searching within the nearby words for terms that are characteristics of the information type to extract (Klein et al., 2004). These terms which are often called keywords, tags, or prefixes are gathered in domain-specific and hand-crafted lexicons. For example, when looking for the total amount of an invoice, a regular expression is likely to detect a lot of numerical values, but relatively few candidates would have the word `total` or one of its abbreviated or upper cased form in their surrounding words.

The regular expressions are usually manually designed by a domain expert which interpret the specifications of the fields (Bhatt et al., 2019). Some incremental systems like TEXTMARKER (Atzmueller et al., 2008) additionally allows their users to refine the set of extraction rules formalized by the expert. However, manually constructing regular expressions may result in tedious work when extracting an amount of fields, even with a moderate variability in their syntax. To reduce human efforts, automatic generation methods can be employed. The regular expressions may be learned from a corpus of documents that are annotated for IE (see the surveys of Sarawagi (2008); Turmo et al. (2006)). But they may also be derived from a set of entities examples (Brauer et al., 2011). This is particularly useful in enterprise settings where generally a lot of structured data is available in databases. To keep the generation tractable, Ciravegna (2001) makes the choice of restricting the space of constructed regular expressions. In contrast, Li et al. (2008) accept initial regular expressions supplied by the experts to naturally provide domain knowledge about the structure of the entities being extracted. This knowledge meaningfully restricts the space of output expressions under consideration while still allowing the algorithm to produce complex regular expressions in a tractable manner.

Rule-based IE is not restricted to regular expressions. Indeed, other methods are employed in the literature for extracting more complex structures, particularly tabular data. SmartFix from Deckert et al. (2011) tackles the table content understanding problem in orders, invoices and medical documents by an expectation-driven approach. To that end, a global quality measure over the set of possible column configurations that involves local and global expectations is optimized by exhaustive search after application of heuristics eliminating a large part of clearly irrelevant configurations. Belaïd and Belaïd (2004) resort to a bottom up morphological tagging approach for delimiting products in invoice bodies and extracting their different fields. After a primary tagging of the words using only their morphology, a secondary contextual tagging is generated by exploiting regular expressions as well as redundancy and regular factors, e.g. the repetition of numerical

terms in similar column cells.

As a post-processing step to improve the extraction accuracy, the consistency and validity of extracted information is often checked with the help of additional input resources. This may be performed by matching the extraction results with entries of a domain specific database, e.g. a vendor (Klein et al., 2004; Bureš et al., 2020) or bank (Lytinen and Gershman, 1986) database. The look up may also be done in pre-built ontologies which are more powerful and formal representations of the domain knowledge (Raoui-Outach et al., 2017).

The strengths of rule-based IE systems are their flexibility and ease of understandability and interpretability for both experts and users (Chiticariu et al., 2013). When an error of extraction occurs, one can trace the specific rules that lead to the error and correct them. Thus, by evaluating the rules against a set of development documents, such IE systems can be gradually improved by incorporating new rules or revising existing rules that are incorrect or suboptimal. Yet, in the process of constantly refining the set of rules, the risk is that the rules become too brittle and break with minor variations of the information patterns (Palm, 2019). Primarly, the non-standardized terminology and layout of business documents makes cumbersome to produce generic rules that accurately extract information for a wide range of document classes.

2.1.2 Using also class specific knowledge

To design more efficient rule-based IE systems, a number of authors exploit the semi-structured or structured nature of business documents. They create or learn rules that are specific to a class of documents by leveraging the similarity of physical and logical structure between documents of the same class. Naturally, IE methods that rely on class specific knowledge operate in two steps as illustrated in the Figure 2.2. For an incoming document to process, its class is firstly detected either directly or by matching with some of the documents previously processed by the system, dubbed as support or reference documents. Secondly, if the class is known by the system, the information is extracted from the document thanks to the fine-grained knowledge emanating from a model of this class or from the support documents representative of this class. After the extraction is completed and verified, the processed document helps to refine the model of the class or is added as a new support document. In the following, we review in detail these two steps.

2.1.2.1 Document classification

If the classes are directly linked to the issuers, one could simply search for information identifying the issuer within the text of the incoming document. Rather than relying on a single field read that would trigger false alarms, Rusinol et al. (2013) cross-check the class prediction by spotting several fields that can be easily detected with domain knowledge rules, including addresses, emails, phone and tax numbers. The issuing company could also be identified with its logo (Cesarini et al., 1997; Alaei et al., 2016). This visual comparison is particularly judicious for scanned documents with a high degree of

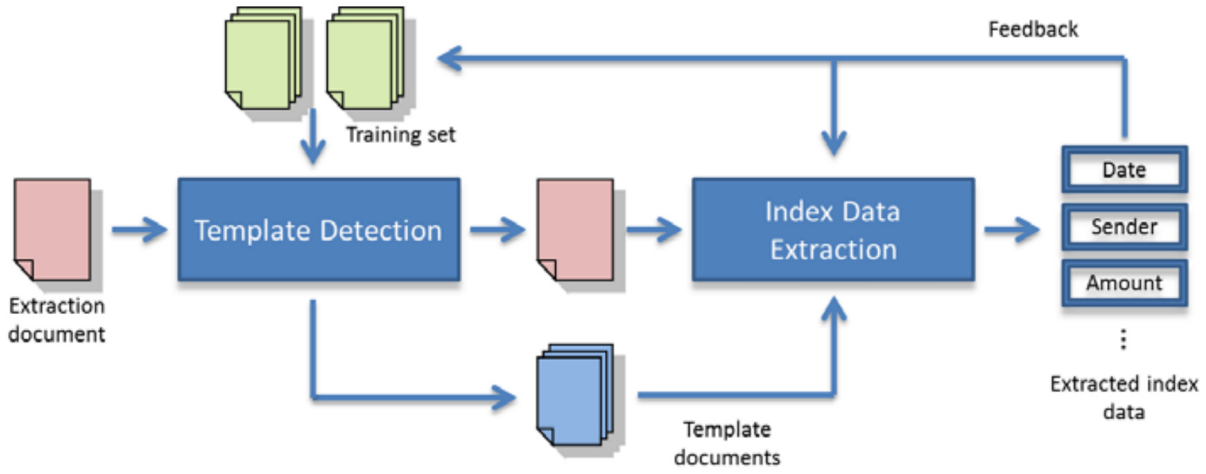


Figure 2.2: **Overview of an extraction system using class specific knowledge.** For extracting information like the date and the sender address from the incoming document (in red), such an IE system first determines to which class the document belongs among the classes of documents known by the system (in green). It then uses a model or the previously processed documents (in blue) corresponding to the identified class in order to provide class specific knowledge helping the extraction. Image reproduced from Esser et al. (2012).

degradation and noise where a text-based classification is likely to fail due to poor OCR engine performances.

However, for legal or aesthetic reasons, an issuer could change the template of its documents over time, sometimes also changing the position of valuable information. Therefore, it is preferable to directly classify the documents according to their structure. To that end, a diversity of learning-based approaches have been proposed in the literature. The first type of methods exploit the visual similarity between documents (Hu et al., 2000). Appiani et al. (2001) resort to decisions trees over the sets of recursive cuts performed over white spaces while Bartoli et al. (2010) employ either a Support Vector Machine (SVM) or a distance-based classifier with the spatial density of black pixels and of image edges as features. Esser et al. (2012) describe a document in a graphical way by binarizing the document over a coarse grid and consider that each line of the grid is a binary string. The visual similarity between two documents is then computed by the Levenshtein distance[2] between the two sets of strings.

Esser et al. (2012) also report an alternative method, named *wordpos*, that leverages the document text. The authors represent a document as a tf-idf[3] weighted bag of frequent and specially formatted (bold, italic, underlined) words with their low resolution position, typically within a 30 by 40 grid for the A4 format. This document representation intending to capture the keywords at invariant positions is then injected in a k-nearest

[2] https://en.wikipedia.org/wiki/Levenshtein_distance
[3] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

neighbors (kNN) classifier to obtain the k support documents that have the closest structure. Among the k matching documents identified by the *wordpos* method, Schuster et al. (2013b) further discard the support documents having a normalized similarity score below an empirically determined threshold. Finally, Dhakal et al. (2019) combine visual and textual similarity measures for a more robust document classification. The visual part is calculated with the cosine distance between the Singular Value Decomposition (SVD) (Klema and Laub, 1980) of document images while the textual part is computed with the edit distance of the top-n and bottom-n text lines between the two compared documents. The similarity scores of both modalities are then summed and the highest overall score determines the template of the incoming document.

2.1.2.2 Information extraction

For highly structured documents such as forms, the information is located in the same position for the documents of a given class (Cesarini et al., 1998). Therefore, to actually extract the fields of an incoming document, the system just needs to apply a fixed spatial mask that has been previously constructed from the location of the targeted information in support documents of the same class (ABBYY, 2004). For instance, the Position-based Indexer from Schuster et al. (2013b) scores the field candidates by computing their bounding box overlap with the field values of the support documents using the Jaccard index (Jaccard, 1901). This extraction strategy is greatly effective for structured documents, providing that the variability in the scanning process of paper-based documents (skewing, shifting, distortion, orientation, etc.) is correctly handled (Casey et al., 1992).

In the case of semi-structured documents, the field values are likely to change of absolute positions between the documents of a class while keeping a regularity in their relative positioning. Such a phenomenon is usually observed for fields that can occur a variable number of times in a document, like the data gathered in tables. For example, for the document class illustrated in the Figure 1.4c, the total amount of the purchase order is located at the bottom of the table of ordered products, making its absolute position dependent of the table size, i.e the number of products. In order to deal with these floating fields, less rigid approaches have been developed in the literature. For extracting information of bank cheques, payment slips and bills, Tang et al. (1995) detect and rely on the ruling lines of tables that structure the documents. Targeting not only the table-like documents, Cesarini et al. (1998) propose a more generic approach based on graphs. Particularly, they model the document's layout with an attributed relational graph over the relevant objects pointed by an operator, such as lines, logos, keywords and fields instances. The edge between two nodes represents the vector between the barycenters of the nodes within the document. The search for the keywords helping in the extraction of the field values is performed with algorithms based on morphological operations and connected components as well as connectionist models. Similarly, for each document class and each targeted field, Rusinol et al. (2013) define a star graph that encodes the pairwise spatial relationships between the field instance and the rest of the words in the support document. For a given word of the incoming document, all the

nodes of the graph that have the same transcription are retrieved. Each matching node then supplies an estimation of the position of the field instance. After having processed all the words, the closest word bounding-box to the maximum peak in the voting space is selected as the field value to extract. Instead of looking for common words in the whole documents, Dhakal et al. (2019) propose to restrict the search to an approximate region of interest that is determined by maximizing the correlation coefficient between image patches of the incoming and support documents. Medvet et al. (2011) rather represent a class of documents with a probabilistic model estimating the likelihood that a set of text blocks contain the expected field. The model is made of a set of univariate distributions for all spatial and textual attributes in order to keep the maximum likelihood estimation tractable. Bart and Sarkar (2010) also make use of a probabilistic model based on a wide variety of perceptual cues to not only extract repeated structures across the documents sharing the same class but also within a single document. Once a single occurrence of the structure to be found is annotated, e.g. the fields contained in the first row of a table, the remaining occurrences of the document are extracted with the probabilistic model.

Such above-mentioned approaches satisfyingly extract information once having processed multiple documents of the same class. However, they naturally suffer from poor performances when dealing with classes that are unknown or have not been extensively processed. Evaluated on 10 commonly used fields, Esser et al. (2014) show that the average F1 score of their system drops from 88% to 22 and 78% respectively for zero and one support document of the same class. To mitigate this cold start performances issue, these authors propose in a later paper a cooperative approach to information extraction by sharing the class specific knowledge between the end users of their system (Schuster et al., 2015). Assessed on a test set with 25% of the document classes that are used by at least two users, they show that the zero-shot F1 score is improved from 5% to 49% with their collaborative approach, thus creating a much better user experience. As offered by the system of Klein et al. (2004), another direction for improving the extraction is to combine the class specific knowledge with the general domain knowledge shared by all the document classes. Cesarini et al. (2003) show that resorting to generic rules when the class is unknown or the class-specific strategies have failed results in greater performances since it increases the reliability of the keywords localization. d'Andecy et al. (2018) suggest a generic method to inject a-priory knowledge into the star graph of each class from Rusinol et al. (2013). Assuming that the documents follow a Manhattan organization[4], they enhance the importance of the closest words located in the same line as a field instance. Such boosting of the horizontal words efficiently prunes the irrelevant nodes in the star graphs and thus notably improve extraction performances in the one-shot setting, with a gain from 87% to 94% accuracy on the difficult fields.

Despite these enhancements tackling the cold start performances, the annotation and correction efforts still scale linearly with the number of document classes processed by the IE system. For industrial applications, the number of classes are usually very high and increasing over time (Aslan et al., 2016). Therefore, this still leads to significant manual

[4] https://en.wikipedia.org/wiki/Taxicab_geometry

work from either the domain experts or the end users while we aim at keeping the human involvement to a minimum in the extraction process. Besides, the high number of classes also makes the document classification step more error prone. This further negatively impacts the success of class-specific strategies for extracting information since the whole process is greatly dependent of the document class decision (Esser et al., 2014).

Overall, the rule-based approaches have stayed for a long time the approaches of choice for IE, particularly in the commercial systems due to their ease of explainability and interpretability (Chiticariu et al., 2013). Humans and machines can contribute to the same extraction model, making easy to incorporate domain knowledge.

However, the design and maintenance of rules require deep expertise and tedious manual labour. These efforts rapidly become overwhelming as the number of targeted fields and document classes grow. Besides, most of the IE methods using rules focus on recognizing field values that are characterized by strong syntactical patterns or have particular keywords in their close vicinity. Very few works tackle the extraction of recurring and more complex structures of information such as tabular data that impose to understand longer range dependencies between the document words. Indeed, this would require extensive human efforts to figure out rules that capture such information types. For all these reasons, Xu et al. (2020c) have declared that "it is inevitable to leverage statistical machine learning approaches in the Document Analysis and Recognition (DAR) research."

Before delving into its application to IE, we review in the next section the main principles and techniques of machine learning.

2.2 Background in machine learning

Overview Belonging to the Artificial Intelligence (AI) field, Machine Learning (ML) is "the study of computer algorithms that improve automatically through experience" (Mitchell, 1997). More concretely, ML involves building models learning from data provided so that the computers perform tasks without being explicitly programmed to do so (Koza et al., 1996). The resulting models may then be used to make predictions of the future if we can assume that the future is similar to the past (Alpaydin, 2020). The models may also be descriptive to gain knowledge from data. Developing a ML model is particularly relevant for advanced tasks where it can be challenging for a human programmer to specify every step required for solving the problems at hand. For instance, consider the problem of email filtering, where we need to separate spam emails from legitimate messages (Friedman et al., 2001). To solve it, one needs to transform the input sequence of characters representing the email text into a binary output indicating whether the email is spam or not. However, this transformation is rather complex to manually encode and what can be considered spam changes in time and from individual to individual. Indeed, the spammers may observe and analyze the predictions of some spam detection models and change the content of their emails to better fool the models. Therefore, since data is generally abundant nowadays, it turns out to be more effective to help the machine

develop its own algorithm by providing the expected answers — spam or legitimate in this case — to some available inputs.

Machine learning has been extensively applied in various application domains such as computer vision, natural language processing, speech processing, bioinformatics, telecommunication and banking (Tzanis et al., 2006). Apart from email filtering, ML has thus been used for a wide variety of tasks like DNA classification, customer market segmentation, face recognition, document categorization, medical diagnosis, data center optimization, movie recommendation and game solving (Das et al., 2015). For building mathematical models, machine learning relies on the theory of statistics with the branches of probability, decision and information theory (Deisenroth et al., 2020). It also involves the computer science discipline for designing efficient algorithms for training and making use of such models as well as storing and processing the data at our disposal (Alpaydin, 2020).

The ML approaches are usually divided into three broad categories, depending on the nature of the *feedback* signal available for learning (Bishop, 2006):

- In supervised learning, the computer is presented with examples of inputs and their outputs, whose correct values are given by a supervisor, and the aim is to learn the mapping from the input to the output.

- In unsupervised learning, we only have a set of input data without their corresponding target values. The goal is to find the regularities among the input, e.g. retrieving groups of similar inputs as in the clustering task or determining the distribution of the input data. It may also deals with projecting the data into lower dimensional spaces, either for visualization purposes or for constructing meaningful representations of the input to improve the learning of a subsequent supervised algorithm (Van Der Maaten et al., 2009).

- In reinforcement learning, the feedback are rewards that a computer program receives after each of its action when interacting with a dynamic environment. The program, called an agent, learns a policy of actions maximizing the cumulative rewards in order to achieve a high-level goal, e.g. escaping from a maze. In contrast to supervised learning, here the learning algorithm is not given examples of optimal outputs but must instead discover them by exploring the environment through a trial and error process.

In the rest of the thesis, we will exclusively mention and propose information extraction algorithms that learns in a supervised fashion. Therefore, in what follows, we only describe in further details supervised learning although unsupervised and reinforcement learning share some key ideas, tools and techniques with the supervised setting (Bishop, 2006).

Problem formulation We note $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ respectively the input and output of a supervised learning system. $\mathcal{X}$ and $\mathcal{Y}$ represent the input and output spaces while $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ denotes the example domain. The examples are drawn from a distribution $\mathcal{D}$

$\xrightarrow{f}$ strawberry

density	pH	sulphates	alcohol
0.9978	3.51	0.56	9.4

$\xrightarrow{f}$

quality
5

(a) Classification (b) Regression

Figure 2.3: **Supervised learning: examples of classification and regression tasks**. The goal is to learn a model, represented by f, in order to predict either categorical (2.3a: detecting the presence or absence of an object within an image (Deng et al., 2009)) or continuous (2.3b: rating of a wine (Cortez et al., 2009)) values.

over the domain $\mathcal{Z}$. $\boldsymbol{x}$ can correspond to any type of data, e.g. an image (Figure 2.3a) or a set of tabular entries (Figure 2.3b). Likewise, $\boldsymbol{y}$ can take multiple forms but is usually divided in two categories (Alpaydin, 2020). If the output is a discrete value, then we tackle a classification problem. If $\boldsymbol{y}$ is a continuous value, then we pursue regression. We give an example for both problem types in Figure 2.3. Even if this figure only illustrates one-dimensional output values, $\boldsymbol{y}$ may be multi-dimensional. We note d_{o} its dimension and d the dimension of the input $\boldsymbol{x}$.

The goal of a supervised learning system is thus to learn the mapping $f \in \mathcal{H}$ that correctly predicts the value $\boldsymbol{y}$ from the input $\boldsymbol{x}$, with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{Z}$ an example from the distribution $\mathcal{D}$. The prediction produced by the hypothesis function f is denoted $\hat{y}$:

$$\hat{y} = f(\boldsymbol{x}) \tag{2.1}$$

The mapping f is determined during the training phase on the basis of some training data that are drawn from the distribution $\mathcal{D}$ of examples. This constitutes the training dataset $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n = \{\boldsymbol{z}_i\}_{i=1}^n$ that contains n examples. f is obtained by minimizing the mean prediction error on this dataset, also known as the training loss L_n:

$$L_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f, \boldsymbol{z}_i) \tag{2.2}$$

$$f = \arg\min_{f' \in \mathcal{H}} L_n(f') \tag{2.3}$$

where ℓ is the loss function measuring the prediction error of a hypothesis function for given examples. There's no one-size-fits-all loss function in machine learning, with various factors involved in choosing a loss function. The main factor is the type of learning task we are dealing with, i.e. regression or classification. For the former, one common choice is the L2 loss:

$$\forall \boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d, \boldsymbol{y} \in \mathcal{Y} \subset \mathbb{R}^{d_{\text{o}}},\ \ell(f, \boldsymbol{z}) = \|f(\boldsymbol{x}) - \boldsymbol{y}\|_2^2 \tag{2.4}$$

For classification problems, one usually chooses the cross-entropy loss:

$$\forall \boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d, \boldsymbol{y} \in \mathcal{Y} \subset \{0,1\}^{d_o},\ \ell(f, \boldsymbol{z}) = -\sum_{c=1}^{d_o} y^c\, log(f(\boldsymbol{x})^c) \tag{2.5}$$

We refer the reader to Janocha and Czarnecki (2017) for a more exhaustive list of usual loss functions.

In practice, we seek to acquire an hypothesis function f for making predictions on input values $\{\boldsymbol{x}_i\}_i$ for which we do not have their output $\{\boldsymbol{y}_i\}_i$. To generate useful and accurate predictions, the model must thus have a low prediction error not only on the training dataset but also for examples from the distribution $\mathcal{D}$ that are not part of it. This generalization ability is crucial since the variability of the input is such in practical distributions that the training set can comprise only a tiny fraction of all possible input values (Bishop, 2006). In machine learning, we generally distinguish two extreme cases based on the performance of the hypothesis function on the training dataset as well as on the distribution samples outside this subset. They are called *overfitting* and *underfitting* and are explained in Figure 2.4.

Model parametrization When looking for an hypothesis function f that minimizes the training loss, we generally constrain the size and complexity of the hypothesis space $\mathcal{H}$ in order to make the search process tractable. We seek for a restricted hypothesis space that is believed to contain a function that is a good approximation of the true mapping. Therefore, there exists a "trade-off between the expressiveness of a hypothesis space and the complexity of finding a good hypothesis within that space" (Russell and Norvig, 2009). In practice, choosing a relevant hypothesis space is very challenging and it is often more efficient to spot-check a range of different hypothesis spaces. Concretely, determining a hypothesis space mainly involves choosing the type and configuration of algorithm that a hypothesis function may represent.

For most machine learning algorithms, the hypothesis function is parametrized by a set of parameters $\boldsymbol{\theta} \in \Theta$, where Θ is the parameters space. We thus note the model $f_{\boldsymbol{\theta}}$ and its predictions $\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{\theta})$. In this case, solving a supervised learning problem boils down to determine the values $\boldsymbol{\theta}^*$ of model parameters that minimize the training loss $L_n(\boldsymbol{\theta})$. This changes the equations (2.2) and (2.3) to:

$$L_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}, \boldsymbol{z}_i) \tag{2.6}$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} L_n(\boldsymbol{\theta}) \tag{2.7}$$

One of the simplest and oldest parametrized ML algorithm is the linear regression which solves regression problems (Yan and Su, 2009). As the name implies, the output $\boldsymbol{y} \in \mathbb{R}^{d_o}$ is a linear combination of the input $\boldsymbol{x} \in \mathbb{R}^d$. For a single output value (i.e. $d_0 = 1$), a linear regression model produces the following predictions:

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} \tag{2.8}$$

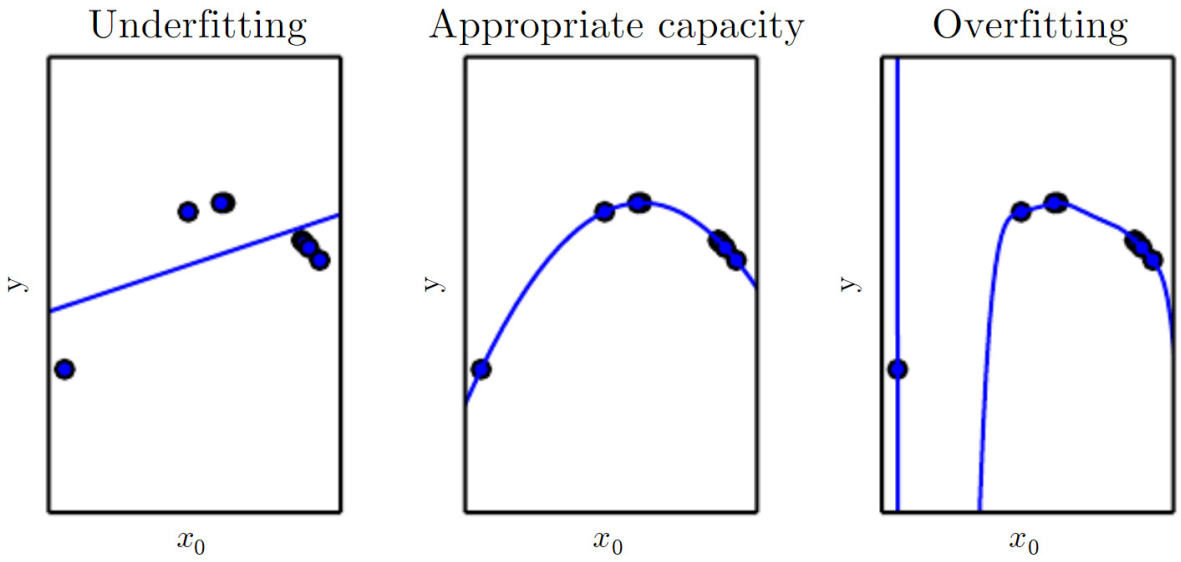


Figure 2.4: **Balance between underfitting and overfitting the training data.** In this example, we aim to learn the quadratic mapping between the output y and the input x_0 given few random samples of the distribution. We present three different hypothesis functions that have been fitted on these training samples. All models are polynomial functions of x_0. (Left) A trained linear function would suffer from *underfitting* since it cannot capture the curvature that is present in the data, thus underperforming both on the training set as well as the rest of distribution. (Right) A polynomial of degree 9 fit to the data suffers from *overfitting*. The solution passes through all the training points exactly, resulting in a zero training loss. Yet, due to its inappropriate complexity, it does not capture the underlying quadratic structure but rather the small stochastic noise of the distribution. Indeed, it now has a deep valley between two training points, inducing high prediction errors for new samples within this portion of the domain. (Center) A trained quadratic function performs well on the training examples while generalizing to unseen points over all the distribution domain. Image reproduced from Bengio et al. (2017).

where $\boldsymbol{w} \in \mathbb{R}^d$ are the model parameters.

A number of other ML algorithms have been later proposed such as kernel-based methods (Hofmann et al., 2008), e.g. Support Vector Machines (SVMs), and tree-based methods (Safavian and Landgrebe, 1991), e.g. decision trees, random forests and gradient boosting trees. However, for much ongoing work in machine learning, including its application to the Information Extraction field, the neural networks have become the dominant approach in the last decade (Alpaydin, 2020). Neural networks are composed of a series of layers, each layer containing neurons that act as a small hypothesis function. The mapping $f_{\boldsymbol{\theta}}$ of a neural network is obtained by repeatedly applying each layer to the output of the previous layer, the first layer being applied to the input $\boldsymbol{x}$:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_{\boldsymbol{\theta^L}}(f_{\boldsymbol{\theta^{L-1}}}(...f_{\boldsymbol{\theta^l}}(...f_{\boldsymbol{\theta^1}}(\boldsymbol{x})...)...)) \tag{2.9}$$

$$\boldsymbol{h^l} = f_{\boldsymbol{\theta^l}}(\boldsymbol{h^{l-1}}) \; with \; \boldsymbol{h^0} = \boldsymbol{x} \tag{2.10}$$

where $L \in \mathbb{N}$ is the number of layers in the neural network. $\boldsymbol{\theta} = \{\boldsymbol{\theta^1}, ..., \boldsymbol{\theta^L}\} \in \Theta$ are its parameters to optimize. $f_{\boldsymbol{\theta^l}}$ represents its l-th layer whose input, output and parameters are respectively $\boldsymbol{h^{l-1}}$, $\boldsymbol{h^l}$ and $\boldsymbol{\theta^l}$. Neural layers that are not the final output layer, i.e. $l = 1...L-1$, are called *hidden* layers. This structure for the hypothesis function $f_{\boldsymbol{\theta}}$ allows to learn complicated concepts by building them out of simpler ones as we move from the lowest to the highest layers (Bengio et al., 2017). Such hierarchies of concepts to learn may comprise an amount of levels for input types such as images, videos or text. This implies that the number of layers may be quite important, exceeding one hundred layers for some models like ResNet (He et al., 2016) used for image recognition. For this reason, the neural networks are characterized as *deep* learning methods.

We describe in a later section (2.2.1) the most commonly used neural network layers.

Solving the optimization problem In this paragraph, we detail how to find the values of model parameters $\boldsymbol{\theta} \in \Theta$ that minimize the training loss given in the equation (2.6). For some ML algorithm and loss function ℓ choices, one can get a closed form of the optimal parameter values. This is done by computing the gradient of the training loss $\nabla_{\boldsymbol{\theta}} L_n(\boldsymbol{\theta})$ and searching parameter values $\boldsymbol{\theta}^*$ that make it equal to zero. For instance, for the single output linear regression with the L2 loss (equation (2.4)), the optimal parameters $w^* \in \mathbb{R}^d$ are given by[5] $\boldsymbol{w}^* = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$, where $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y} = [y_1, ..., y_n] \in \mathbb{R}^n$ are the concatenation of the input and output of the n training examples.

Yet, in the general case, closed forms of optimal parameters are either unavailable or too expensive to compute, e.g. computing the term $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ for the linear regression is intractable when facing large numbers of examples n and input dimensions d. Rather, we iteratively update the parameters in the opposite direction of the gradient of the training loss. This technique, known as *gradient descent*, is nowadays one of the most popular algorithms to solve this optimization problem, especially when dealing with neural networks (Ruder, 2016). The learning rate η determines the size of the steps we take to

[5] View the subsection 5.1.4 of Bengio et al. (2017) for the derivation proof.

reach a minimum of the training loss. There exists three variants of gradient descent, which differ by the amount of training data we use for computing the gradient. The vanilla version, called batch gradient descent, computes the loss gradient on the entire training dataset before making an update:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} L_n(\boldsymbol{\theta}) \tag{2.11}$$

with $\boldsymbol{\theta} \in \Theta$ that is randomly initialized to start the optimization process. Batch gradient descent is guaranteed to converge to a minimum of the training loss, either to a global optimum for convex error surfaces or to a local optimum for non-convex surfaces. However, this convergence usually requires many parameters updates, which makes batch gradient descent very slow for large training sets. In contrast, Stochastic Gradient Descent (SGD) performs a parameter update for each training example $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i \in \{1,..,n\}}$:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}, \boldsymbol{z}_i) \tag{2.12}$$

Benefiting of frequent parameter updates, this second version of gradient descent is generally much faster. Still, the updates suffer from high variance that makes the loss fluctuate heavily. This complicates the convergence to the minimum at the end of the optimization process. Finally, mini-batch gradient descent takes the best of both worlds by performing an update for every mini-batch of m training examples $\{\boldsymbol{z}_j\}_{j \in \{i,..,i+m\}}$:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{j=i}^{i+m} \ell(f_{\boldsymbol{\theta}}, \boldsymbol{z}_j) \right] \tag{2.13}$$

It reduces the variance of the parameter updates thus stabilizing convergence while still performing quite frequent updates. Common mini-batch sizes range between a few units and a few hundreds. Mini-batch gradient descent is typically the algorithm of choice when training a neural network (Bottou, 1991). In the literature, we traditionally also employ the term SGD when mini-batches are used.

We achieve one *iteration* of the learning process when a parameter update have been performed. One *epoch* is realized when we have loop through the entire training set. To avoid providing a meaningful order of examples that may bias the optimization process, we commonly shuffle the training dataset at the beginning of each epoch. The number of iterations or epochs to perform until reaching a minimum of the training loss is highly dependent of the task, data distribution and the used ML model. One simple method for deciding when to stop the training is to keep track of the prediction error on data samples outside the training set, thus constituting a development or validation set. When this validation error ceases to improve with new parameter updates, the model training should be stopped. This technique is known as *early stopping*.

Mini-batch gradient descent still presents some challenges to address for achieving a good convergence. Indeed, the learning rate η must be carefully chosen to avoid painful slow convergence (η too low) or fluctuations around the minimum or even divergence (η too large). Besides, the same learning rate is applied to all parameters which is not

desirable since parts of the input vectors may have very different frequencies in the training set. Vanilla gradient descent is also easily trapped in suboptimal local optima (Dauphin et al., 2014). Therefore, more advanced optimizers have been proposed to improve the convergence speed and quality of the training process (Qian, 1999; Duchi et al., 2011; Kingma and Ba, 2015; Dozat, 2016). Overall, these optimizers feature momentum terms for accelerating the descent of the gradient in the relevant direction and adaptive learning rates. For rapidly training a neural network, Ruder (2016) recommends an optimizer with adaptive learning rates and mentions Adam (Kingma and Ba, 2015) as the best overall choice at his time of writing.

Finally, even if they all affect the value of the training loss, not all the parameters of a model can or are desired to be learned with gradient based optimization methods. Such parameters to be ignored from the gradient are called *hyperparameters* and are mainly parameters describing the structure and complexity of the model (Bergstra and Bengio, 2012), e.g. the number L of layers in a neural network. Their value must therefore be set before starting the gradient based optimization. Although a wide variety of methods have been introduced for finding their optimal values (Claesen and De Moor, 2015), hyperparameter search is still often performed by manual trial and error on the development dataset.

Backpropagation When applying SGD or any other gradient based optimizer to a neural network, the gradient of the training loss must be computed for the parameters of each of its layer. To that end, the chain rule is utilized to compute the gradient one layer at a time, starting with the layer L that produces the final outputs:

$$\frac{\partial L_n(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^l} = \frac{\partial L_n(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^L} \left(\prod_{k=l+1}^{L} \frac{\partial \boldsymbol{h}^k}{\partial \boldsymbol{h}^{k-1}} \right) \frac{\partial \boldsymbol{h}^l}{\partial \boldsymbol{\theta}^l} \tag{2.14}$$

This principle is known as *backpropagation* (Rumelhart et al., 1986), which is a short name for backward propagation of errors. Indeed, the training error of a neural network is iteratively propagated from the output layer until the first layer that is connected to the input. Note that the most popular machine learning frameworks (Abadi et al., 2016; Paszke et al., 2019) used for building neural networks provide automatic differentiation that efficiently computes the gradient for all their layers.

We have introduced all the key components for training neural networks in a supervised setting. We will now detail their most common architectures.

2.2.1 Main neural network architectures

Taking their roots in the 1950's, Artificial Neural Network (ANN) have a rich history that alternates between massive enthusiasm and disillusionment (Kurenkov, 2020). Inspired by their biological counterparts, the psychologist Rosenblatt (1957) was the first to propose an ANN, called the Perceptron. Bearing some strong resemblance with the linear regression, the Perceptron is a single-layer network that accepts a set of binary inputs,

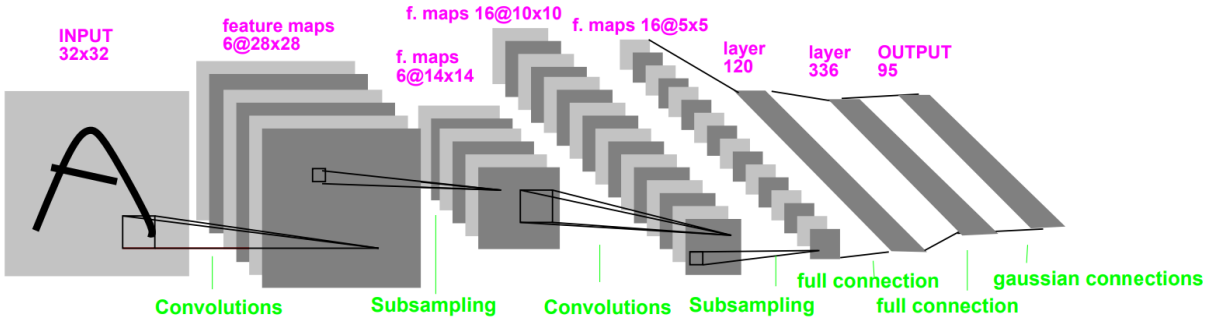


Figure 2.5: **Typical architecture of a Convolutional Neural Network (CNN).** A CNN is composed of convolutional, subsampling (also known as pooling) and fully-connected layers. In this example, the LeNet 5 model was used to decide which ASCII character among 95 elements is contained in a 32×32 image. Image reproduced from Le Cun et al. (1997).

multiply their values by continuous valued weights and sum all obtained values. Finally, a threshold is applied for producing a binary output value. This thresholding is one example of an *activation function*, usually denoted σ. These functions are major components of neural networks since it allows to introduce non-linearities in the model, rendering possible the learning of complex mappings between the input and output. Nevertheless, its single-layer structure prevents the Perceptron to learn any given function. In particular, it fails to represent the exclusive disjunction (XOR) operation (Minsky and Papert, 1969). To remedy this, hierarchical multilayered neural nets were later proposed (Fukushima and Miyake, 1982; Hornik et al., 1989) but the authors were struggling to learn their optimal weights until the backpropagation technique was popularized (Rumelhart et al., 1986). The first notable real-world results appear in the fields of speech recognition (Lang, 1988) and handwritten image recognition (LeCun et al., 1989). Going beyond mere application of backpropagation, LeCun et al. (1989) proposed a set of key modifications of the network architecture to recognize the zip code digits contained in the images. Specifically, they introduced an architecture that embeds *a priori* knowledge about the computer vision task in order to drastically reduce the number of learnable parameters without sacrificing its computational power. This network design is now widely referred to as Convolutional Neural Network (CNN) or ConvNet.

2.2.1.1 Convolutional neural networks

A CNN is a neural network that is composed of different types of layers, at least one layer being convolutional. We depict in Figure 2.5 an example of CNN architecture and give details about each layer type in the following:

- *Convolutional* layers are the key components of CNN models. It consists in applying a convolutional operation with learnable spatial filters. Following the two dimensional structure of images, neurons of such layers are organized in a set of

planes, called feature maps. The input $\boldsymbol{h^{l-1}}$ and output $\boldsymbol{h^l}$ of a layer $l = 1...L$ are 3D tensors of respective size $w^{l-1} \times h^{l-1} \times k^{l-1}$ and $w^l \times h^l \times k^l$, where the first two dimensions refers to the width and height of the feature maps and the third dimension to the number feature maps, also called the layer's depth. A convolutional layer l has k^l filters, each filter producing one of the feature maps. The i-th output feature map denoted $\boldsymbol{h_i^l}$ is given by:

$$\boldsymbol{h_i^l} = \boldsymbol{B_i^l} + \sum_{j=1}^{k^{l-1}} \boldsymbol{K_{ij}^l} * \boldsymbol{h_j^{l-1}} \tag{2.15}$$

where $*$ is the convolution operator, $\boldsymbol{B_i^l}$ is a learnable bias matrix and $\boldsymbol{K_{ij}^l}$ is the filter connecting the j-th feature map of $\boldsymbol{h^{l-1}}$ to the i-th feature map of $\boldsymbol{h^l}$. Having multiple convolutional filters per layer allows to detect different features at the same image location. Taking into account the fact that pixels of the image, and thus neurons within a feature map, that are spatially close are more semantically related than distant elements, each filter has usually a limited size compared to the image dimensions, i.e. from 1×1 up to a few dozens wide and high. In the Figure 2.5, the first layer applied to the input image is a convolutional layer, having 6 filters of size 5×5 that produce feature maps of size 28×28. Since we employ the same filters for each neuron of a feature map, this allows to efficiently detect an element whatever its location in the image.

- *Pooling* layers consist in subsampling the feature maps of the previous layer, often from a convolutional layer. This operation looses some positional information of features but allows to more efficiently learn higher level features in the next layers of the network. This is performed by a sliding window of size $f^l \times f^l$ that browses each feature map of the previous layer with a stride of $s^l \times s^l$. The reduction of size is performed with a parameter-free aggregation operator, usually the mean or maximum of the neuron values contained in the window. Hence, the pooling layer that takes as input a feature map of size $w^{l-1} \times h^{l-1} \times k^{l-1}$ produces a feature map of size $w^l \times h^l \times k^l$ with:

$$w^l = (w^{l-1} - f^l)/s^l + 1 \tag{2.16}$$
$$h^l = (h^{l-1} - f^l)/s^l + 1 \tag{2.17}$$
$$k^l = k^{l-1} \tag{2.18}$$

  In practice, there are only two commonly seen configurations of the pooling layer: $(f^l = 3, s^l = 2)$ and more frequently, $(f^l = 2, s^l = 2)$ like for the two pooling layers in Figure 2.5. Pooling sizes with larger sizes are too destructive.

- *Fully-connected* or *dense* layers are extensions of the Perceptron (Rosenblatt, 1957). Unlike this former neural network, such a layer can accept a set of continuous values $\boldsymbol{h^{l-1}} \in \mathbb{R}^{m^{l-1}}$ as input and output multiple values $\boldsymbol{h^l} \in \mathbb{R}^{m^l}$. The dense layer applies

a linear transformation on the input such as:

$$\boldsymbol{h^l} = \boldsymbol{W^l h^{l-1}} + \boldsymbol{b^l} \tag{2.19}$$

where $\boldsymbol{W^l} \in \mathbb{R}^{m^l \times m^{l-1}}$ is a matrix of learnable weights and $\boldsymbol{b^l} \in \mathbb{R}^{m^l}$ is a vector of learnable bias. Unlike the pooling and convolutional layers, each neuron of a fully-connected layer is connected to all the neurons of the previous layer. Stacking multiple fully-connected layers on top of each other is referred as a Multi-Layer Perceptron (MLP). In Figure 2.5, a MLP constituted of 3 dense layers of size 120, 336 and 95 is added on top of the convolutional and pooling layers for predicting the output classes. Note that we first flatten and concatenate the feature maps of the last locally-connected layer to feed the first fully-connected layer.

- In general, we employ an activation function σ after both convolutional and fully-connected layers to help the training process. In CNNs, the most commonly functions employed nowadays are the Rectified Linear Unit (ReLU) function ($\forall x \in \mathbb{R}$, $ReLU(x) = max(0, x)$) and its derived forms (Clevert et al., 2016). For classification tasks like in the Figure 2.5, the last fully-connected layer usually resorts to the *softmax* function ($\forall \boldsymbol{x} \in \mathbb{R}^{d_o}, \forall i = 1...d_o$, $\text{softmax}(x)_i = \frac{e^{z_i}}{\sum_{j=1}^{d_o} e^{z_j}}$) to obtain the predicted probabilities of the d_o target classes, e.g. ASCII characters.

2.2.1.2 Recurrent neural networks

The CNN and MLP models that we have just described are *feedforward* networks, meaning that the output of neurons in a given layer acts as input to only neurons in a next layer. However, the feedforward property makes such networks impractical to process variable-length inputs such as temporal sequences $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_t, ..., \boldsymbol{x}_T)_{T \in \mathbb{N}}$. Indeed, information cannot easily persist in these traditional networks, preventing reasoning about previous events when making predictions in the present time. This is problematic since sequential data is ubiquitous in real-world systems, e.g sequences of images, phonemes, words and so on. To remedy this, Recurrent Neural Network (RNN) models have been proposed by just connecting the output of a neuron to itself. We illustrate this principle in the Figure 2.6. The unrolled version of RNNs clearly reveals that they are naturally tailored for handling discrete streams of information. By having the output looping back into the network, a RNN is given the possibility to read all the past inputs for the current step, solving the flaws of the feedforward neural networks. Formally, the layer l of a RNN is defined as:

$$\forall t \in \mathbb{N},\ \boldsymbol{h}_t^l = \text{Cell}(\boldsymbol{h}_t^{l-1}, \boldsymbol{h}_{t-1}^l) \tag{2.20}$$

where $\forall t \in \mathbb{N}, h_t^0 = x_t$, $\forall l = 1...L, h_0^l = 0_{d_o}$ and Cell is the chosen recurrent cell. The optimal weights of a RNN are learned by back-propagating the training error not only from the last layer to the first one but also from the last time step to the first one. This generalization of the standard backpropagation, known as *BackPropagation Through Time (BPTT)*, has been independently formalized by several researchers (Mozer, 1989; Robinson and Fallside, 1987; Werbos, 1988).

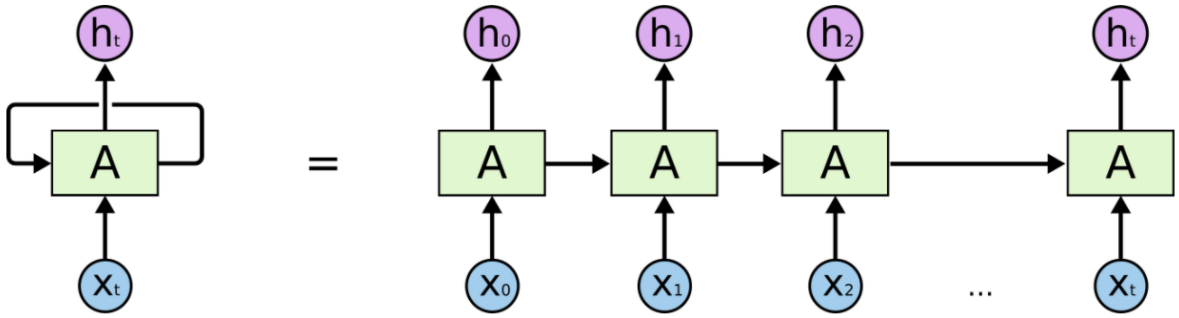
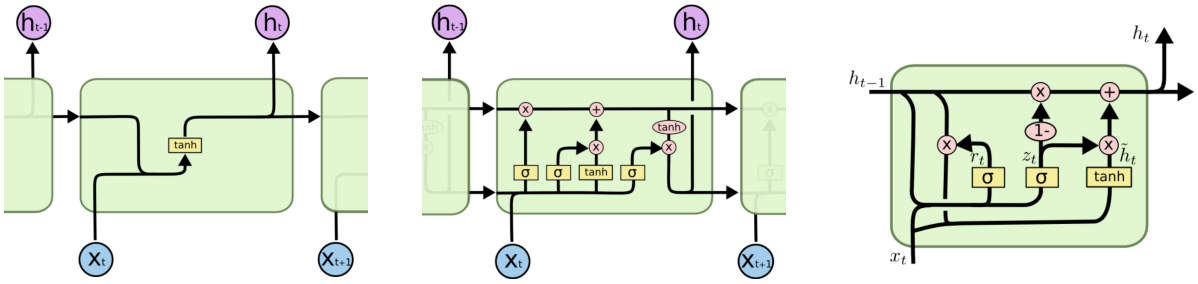


Figure 2.6: **Unrolling a Recurrent Neural Network (RNN).** (Left) Like any neural network, a RNN layer accepts some input $\boldsymbol{x}_t$ and outputs a value $\boldsymbol{h}_t$. Yet, a loop re-injecting the output $\boldsymbol{h}_t$ as input of the layer for the next time step $t+1$ allows to store useful information. (Right) This neural architecture can also be viewed as multiple copies of the same network by unrolling the loop. Image reproduced from Olah (2015).

In the literature, there are multiple choices for RNN cells. We depict three popular choices in Figure 2.7. The most basic RNN architecture (Figure 2.7a) uses a single fully-connected layer taking as input the concatenation of the current input vector $\boldsymbol{x}_t \in \mathbb{R}^d$ and the previous hidden state $\boldsymbol{h}_{t-1} \in \mathbb{R}^{d_o}$. The current hidden $\boldsymbol{h}_t$ is obtained after application of the activation function, historically the hyperbolic tangent:

$$\boldsymbol{h}_t = \tanh\left(\boldsymbol{W}[\boldsymbol{x}_t, \boldsymbol{h}_t] + \boldsymbol{b}\right) \tag{2.21}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_o \times (d+d_o)}$ and $\boldsymbol{b} \in \mathbb{R}^{d_o}$ are respectively the matrix weights and vector bias of the RNN cell. If we only need to look at recent information to do the present task, e.g. predicting the last word in the sentence `the clouds are in the sky`, vanilla RNN architectures perform pretty well. Yet, when the gap with the relevant information is large, e.g. considering some context words from another paragraph, such RNNs struggle to learn to use past information. Bengio et al. (1994) formally proved that vanilla RNN suffer from vanishing or exploding gradients of the error when dealing with long sequences, which makes their training difficult. Therefore, improved RNN cells have been introduced in later works to learn long-term dependencies. The first prominent version was the Long Short-Term Memory (LSTM) by Hochreiter and Schmidhuber (1997) (Figure 2.7b). Compared to the vanilla RNN, the LSTM cell also has a cell state $\boldsymbol{c}_t \in \mathbb{R}^{d_o}$ that aims to store longer term information than the hidden state $\boldsymbol{h}_t \in \mathbb{R}^{d_o}$. The LSTM removes or adds information to the cell state by resorting to some structures called *gates* which are critical for improving memorization abilities (Tallec and Ollivier, 2018). There are three



(a) Vanilla RNN (b) LSTM (c) GRU

Figure 2.7: **Examples of cells for Recurrent Neural Networks (RNNs).** Vanilla RNNs have a single block mixing the input $\boldsymbol{x}_t$ and the previous hidden state $\boldsymbol{h}_{t-1}$ to compute the current hidden state $\boldsymbol{h}_t$. Instead, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) cells have a more complex structure for better capturing long-term dependencies within the input and output sequences. Image reproduced from Olah (2015).

gates impacting the cell state, namely the input $\boldsymbol{i}_t$, the forget $\boldsymbol{f}_t$ and the output $\boldsymbol{o}_t$ gates:

$$\boldsymbol{f}_t = \sigma_g\left(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \tag{2.22}$$
$$\boldsymbol{i}_t = \sigma_g\left(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \tag{2.23}$$
$$\boldsymbol{o}_t = \sigma_g\left(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right) \tag{2.24}$$
$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) \tag{2.25}$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t \tag{2.26}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \tanh\left(\boldsymbol{c}_t\right) \tag{2.27}$$

Where $c_0 = 0_{d_o}$, $\circ$ is the element-wise product, $\boldsymbol{W} \in \mathbb{R}^{d_o \times d}$, $\boldsymbol{U} \in \mathbb{R}^{d_o \times d_o}$ and $\boldsymbol{b} \in \mathbb{R}^{d_o}$ are weight matrices and bias vectors to learn during model training. The gates takes values in the $[0, 1]$ segment thanks to the sigmoid function $\sigma_g(x) = \frac{1}{1+e^{-x}}$. A number of follow-up works have slightly modified the original LSTM formulation. Some added "peephole" connections between the cell state and certain gates (Gers and Schmidhuber, 2000). Other authors simplified the gating mechanism by coupling or combining the forget and input gates like the Gated Recurrent Unit (GRU) from Cho et al. (2014a) (Figure 2.7c). However, Chung et al. (2014); Greff et al. (2016) showed that none of the proposed variants significantly and consistently outperform the standard LSTM architecture.

So far, we only consider browsing the sequence of inputs $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_t, ..., \boldsymbol{x}_T)_{T \in \mathbb{N}}$ from the past to the future, i.e. from $t = 1$ to $t = T$. Yet, if we have access to the whole sequence at prediction time, we can also leverage the context from future inputs when learning a RNN. To that end, a Bidirectional RNN (BRNN) layer (Schuster and Paliwal, 1997) connects two RNN layers of opposite time directions to the same output. As illustrated in the Figure 2.8, the output thus gets information from the past and future simultaneously through the *forward* and *backward* states. BRNNs are trained using similar algorithms to RNNs, because the two directional layers do not have any

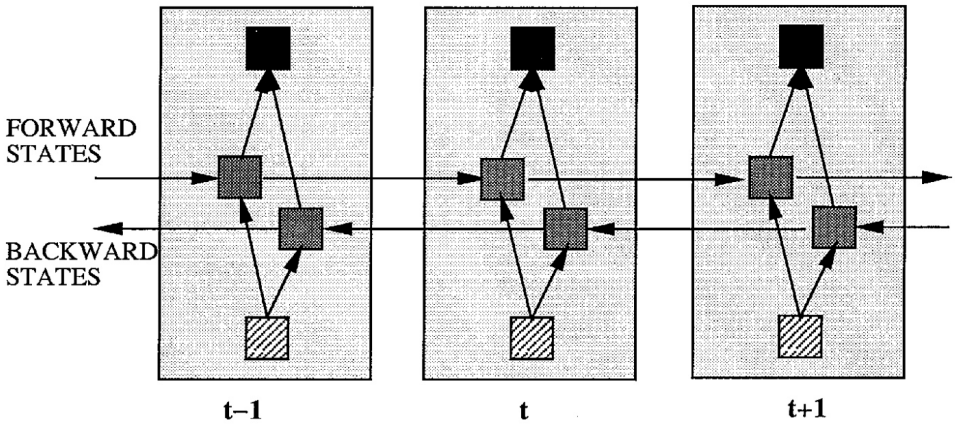


Figure 2.8: **Bidirectional Recurrent Neural Networks (BRNNs).** A BRNN is composed of two independent RNNs that move in opposite directions, one from the past to the future (*forward* states) and the other from the future to the past (*backward* states). The output of both RNNs are concatenated at each time step t for getting context information from both sides. Image reproduced from Schuster and Paliwal (1997).

interaction. Naturally, the bidirectional configuration can be used with any RNN cell type, e.g. Bidirectional Long Short-Term Memory (BLSTM) (Graves and Schmidhuber, 2005).

2.2.1.3 Transformer based networks

For many years, RNNs had been firmly established as the state-of-the-art approaches for sequence modeling problems like machine translation or language modeling (Cho et al., 2014b; Bahdanau et al., 2015). However, such ANNs align the sequence positions to the steps in computation time since the determination of the hidden state $\boldsymbol{h}_t$ at time step t depends on the previous hidden state $\boldsymbol{h}_{t-1}$. Their inherent sequential nature thus prevents parallelization of RNN output computation which becomes critical with long sequence lengths. Therefore, Vaswani et al. (2017) have recently proposed a neural architecture, called *Transformer*, that avoids recurrence and instead relies entirely on an attention mechanism to extract global dependencies between the sequence elements. Experiments on multiple NLP tasks show that these new models are superior to RNN in quality while being more parallelizable and requiring significantly less time to train.

Transformer networks are composed of several identical Transformer layers that are stacked one on top of the other. We depict a Transformer layer in Figure 2.9, specifically the first layer that is directly applied to the input sequences of examples. Each Transformer layer has two sub-layers. The first is a multi-head self-attention mechanism while the second is a two-layer fully-connected network that is applied separately and identically to each sequence position. Residual connections (He et al., 2016) are added

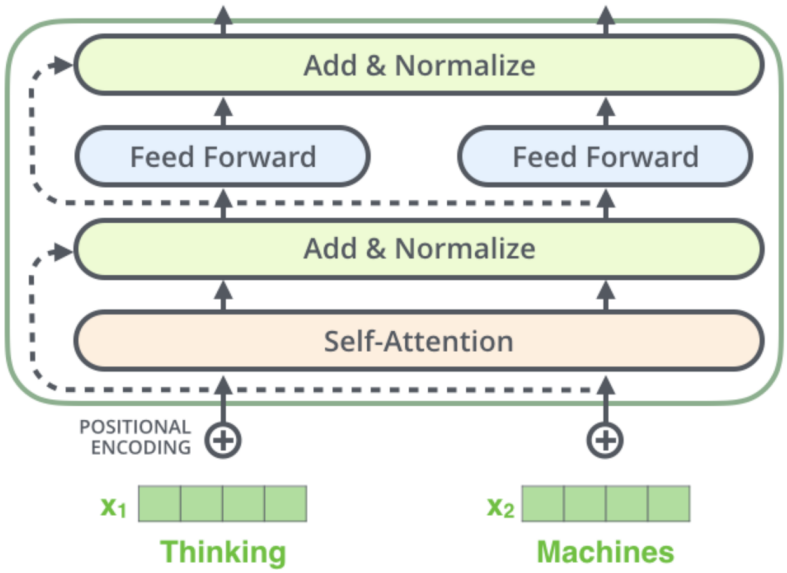


Figure 2.9: **A Transformer layer applied to some input words.** Such a layer successively applies self-attention, normalization and feedforward layers to the word representations of the `Thinking Machines` example. Image reproduced from Alammar (2018a).

between the input and output of each of the two sub-layers to help the training process. Layer normalization (Ba et al., 2016) is also employed to reduce the training time. Since both sub-layers do not explicitly consider the order of sequence elements, the absolute position of elements are added to their original representation before being fed to the Transformer network. The position is encoded through sine and cosine functions of different frequencies, resulting in a positional encoding vector for each input element.

In Figure 2.10, we illustrate the attention mechanism used in the self-attention layers. An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Vaswani et al. (2017) compute the output of attention as follows:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2.28}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are the queries, keys and values matrices. The query matrix packs together a set of queries to simultaneously compute the attention output for multiple elements. In self-attention layers, all the keys, values and queries come from the same place, i.e. the input sequence representations or the output of the previous layer in the Transformer network. Instead of performing a single attention function, the queries, and keys and values are linearly projected $h = 8$ times with distinct parameters before independently performing the attention computation. Attention output of the h heads

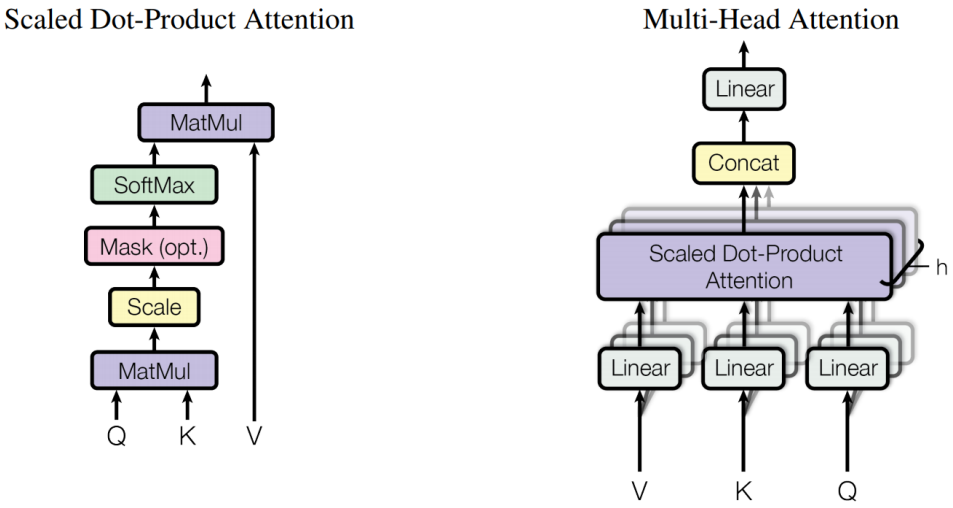


Figure 2.10: **The attention mechanism used by Transformer self-attention layers.** (Left) The attention module computes the compatibility between the query $\boldsymbol{Q}$ and key $\boldsymbol{K}$ matrices, that is then used to weight the value $\boldsymbol{V}$ matrix. (Right) Multi-Head Attention consists of h attention modules running in parallel. Image reproduced from Vaswani et al. (2017).

are then concatenated before a final linear projection. This multi-head attention allows the Transformer to jointly attend to information from different representation subspaces at different positions.

Unlike RNNs, self-attention models like Transformers are able to connect all the sequence elements in a single pass, making the learning of long-range dependencies easier. The main drawback of this architecture is that the computation cost is quadratic in terms of the number of input elements[6]. This hinders the application of Transformer models to problems with very long sequences like document-level text analysis.

2.2.2 Focus on natural language processing methods

2.2.2.1 Tokenization

Making textual data assimilable by a machine learning model is not trivial. Indeed, we need to map raw strings like `The quickest brown fox` to numerical representations that are understandable by a trainable model. To that end, the input text traditionally undergoes several pre-processing steps before being fed to a ML model. The most inevitable step is undoubtedly tokenization. Whatever the NLP model later used, the incoming string, e.g. a sentence, a paragraph or even a document, needs to be broken down into meaningful subsequences (Grefenstette, 1999; A. Mullen et al., 2018). Each resulting sub-

[6] A number of follow-up works have since explored ways to reduce the complexity of self-attention while maintaining model expressiveness, see the survey from Tay et al. (2020)

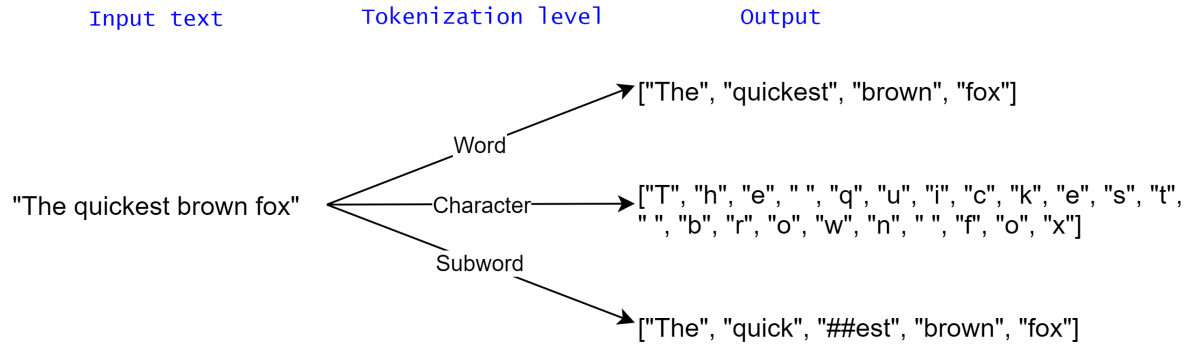


Figure 2.11: **Levels of text tokenization.** A natural language sentence can be splitted with different levels of granularity, i.e. into words, characters or subwords. For subwords, `##` characters designate that a subword is the continuation of a word in the original string.

sequence is called a *token* and groups some of the characters of the original string. In most natural languages, one intuitive way to tokenize text is to split the string each time a space character is encountered, thus forming a list of words. However, as illustrated in Figure 2.11, the tokenization could also be performed at the character or subword level.

The ultimate goal of tokenization is to attribute a numerical value to each token of the input string. To that end, we resort to a fixed-size dictionary of unique tokens, where each token is attributed an integer index. This vocabulary may be constituted by applying the chosen tokenizer to the input strings of the training dataset and enumerating the unique tokens that have been created. To maintain a reasonable size, the vocabulary can be finally restricted to the top K frequently occurring tokens.

So, which tokenization is preferable when tackling an NLP task? Although intuitive for humans, tokenizing at word level may exhibit drawbacks. First, some words of an input string may be not present in the vocabulary, e.g. if these words are not part of the training examples from which we derive the vocabulary. For handling such Out-Of-Vocabulary (OOV) words, a special token — usually denoted `[UNK]` — may be added to the vocabulary. But this means that all the unknown words will be attributed the same representation which may result in non-negligible loss of information. When facing huge corpus of text, the vocabulary may also become very large, making impractical the learning of subsequent ML models. Finally, word tokenization may be too rigid. In case of a brittle modification of a word due to a user typo or to an OCR engine error, e.g. `Nachine` instead of `Machine`, the vocabulary index attributed to the impacted word would be completely different.

On the other side, tokenizing at each character overcomes the downsides of word segmentation. This tokenization preserves the information of the whole input text since there are not any OOV tokens. Indeed, all the words, even rare words, are decomposed

in characters that are all known beforehand, e.g. ASCII[7] or Unicode[8] characters. Using characters as tokens also significantly reduces the vocabulary size. Yet, the character tokenization considerably increases the lengths of the produced token sequences and thus the scale of the inputs that the ML model would have to process. Moreover, unlike words, characters don't have any inherent meaning: their semantic need to be derived with the help of nearby characters.

Finally, subword tokenization brings the best of both worlds by splitting the input strings into units smaller than words but larger than individual characters. Sennrich et al. (2016) showed that encoding (rare) words via subword units is more effective than using large vocabularies of words. Subword tokenization also maintains a reasonable vocabulary size and prevents the number of tokens for processed strings from exploding compared to the character segmentation. It may be performed by splitting at regular intervals to obtain fixed-size character n-grams (Cavnar et al., 1994) or to follow a more complex logic to produce variable-length character subsequences such as Byte Pair Encoding (BPE) (Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012), Unigram (Kudo, 2018) and SentencePiece (Kudo and Richardson, 2018). All these algorithms segment the input strings using a vocabulary of sub-components that frequently appear in the training corpus. For example, the BPE method (Sennrich et al., 2016) constructs the vocabulary by first selecting only the individual characters. It then iteratively merges the most frequently occurring pairs of characters or character sequences until reaching the desired vocabulary size. Once the vocabulary obtained, BPE applies the same merge operations to new input strings for obtaining their subword segmentation. In practice, with such tokenizing approaches, the most common words like `and` are tokenized as whole words while rarer words are broken into smaller grammatically logical chunks. For example, with a relatively small vocabulary size, the comparative adjective `faster` would be split in two more frequent units that are the base adjective `fast` and the comparative suffix `er`.

Depending on the specific NLP task and chosen model, other pre-processing steps could be applied to the text before tokenization. It could be stemming or lemmatizing steps that aim to reduce the words to their roots by removing their attached suffixes and prefixes (Jivani et al., 2011). One widely performed operation is to lowercase all the characters to solve the sparsity issue of some word forms (Uysal and Gunal, 2014). Indeed, without lowercasing, `CANADA` and `Canada` would be treated as two distinct tokens while both case forms convey the same information.

2.2.2.2 Token representations

Once we have performed all our desired pre-processing steps, including tokenization, each input string $\boldsymbol{s}$ is represented by a sequence of integer values $\{s_1, ..., s_k\}_{k \in \mathbb{N}}$ that correspond to vocabulary indexes. The text data has thus been transformed into structured categorical inputs. Since the vocabulary has not any natural ordering, one simple way

[7] http://www.asciitable.com/
[8] https://unicode-table.com/

for modeling vocabulary tokens is to employ *one-hot encoding*. In this schema, a token $i \in \{1, ..., |V|\}$ is represented by a binary vector $x_i \in \{0, 1\}^{|V|}$, where $|V|$ is the vocabulary size. The vector x_i consists of 0 values in all dimensions with the exception of a single 1 in the i-th dimension. One-hot encoding is effective when the vocabulary is relatively small, e.g. a character-only vocabulary (54 tokens in Katti et al. (2018)). Yet, for vocabularies of words and subwords that are typically made of thousands of tokens (Chen et al., 2019), it produces very sparse and uninformative representations of text that can further lead to suboptimal model learning.

Token *embeddings* methods are an alternative to one-hot encoding by representing each token with a real-valued vector that actually encodes its meaning. Unlike one-hot encoding where all token vectors are equidistant, tokens that are close in the embedding space are expected to be similar in meaning (Teller, 2000), e.g. the embedding vectors of the tokens `cat` and `dog` are expected to be closer than the embedding of `computer`. The dimension m of the embedding space is also desired to be much lower than the vocabulary size $|V|$, i.e. $m \ll |V|$. Token embeddings can be obtained using a wide range of language modeling and feature learning techniques such as dimensionality reduction on the token co-occurrence matrix (Lebret and Collobert, 2014) or knowledge based methods (Qureshi and Greene, 2019). However, since some foundational works from Morin and Bengio (2005); Mnih and Hinton (2009), most embeddings methods are based on neural networks.

In neural models, token embeddings are used in the first layer of the network to map all the vocabulary indexes s_i of a tokenized string $\boldsymbol{s} = \{s_1, ..., s_k\}_{k \in \mathbb{N}}$ to real-valued vectors. This is performed by using a lookup table $\boldsymbol{C} \in \mathbb{R}^{|V| \times m}$ that gathers the embedding vectors $\boldsymbol{c}_i \in \mathbb{R}^m$ of all the elements of the vocabulary V (Bengio et al., 2003). As exemplified in Figure 2.12, embeddings of the string tokens are obtained independently from one another. For a token $s_i \in \{1, ..., |V|\}$ of the string $\boldsymbol{s}$, the output of the embedding layer $E_{\boldsymbol{C}}$ is:

$$E_{\boldsymbol{C}}(s_i) = \boldsymbol{c}_{s_i} \tag{2.29}$$

The rest of the neural network utilizes the obtained token representations to perform the problem at hand, e.g. language modeling in Figure 2.12. In this example, the embeddings of n consecutive words are concatenated and passed through two fully-connected layers — with tanh and softmax activation for respectively the hidden and output layers — to determine which is the next word of the sentence. Note that the embedding matrix $\boldsymbol{C}$ are learned like the rest of the model parameters, i.e. with the same instance of gradient descent (Bengio et al., 2003). The embedding matrix can be learned from scratch on the target task, by randomly initializing its weights at the beginning of the learning process (Collobert et al., 2011). However, this may be not very ideal for tasks with a small training dataset since direct training on end tasks can lead to poor representations of tokens, particularly for rare tokens (Bahdanau et al., 2017). This can be alleviated by learning on several related tasks and datasets at once, i.e. in a multi-task setting. For instance, Collobert and Weston (2008) simultaneously learn various classical NLP tasks, namely Part-Of-Speech (POS) tagging, Named Entity Recognition (NER), Semantic Role

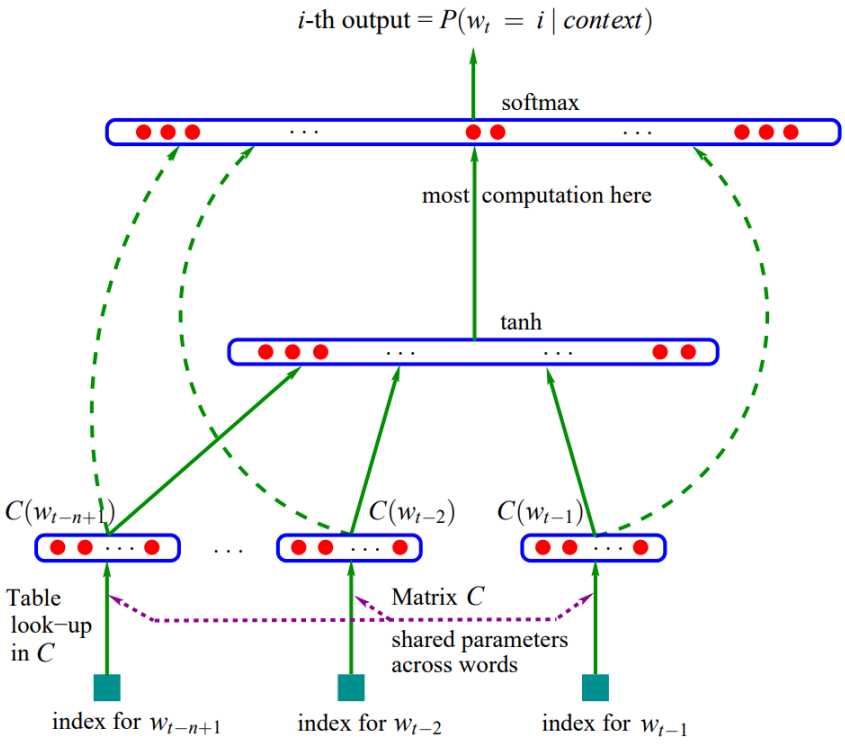


Figure 2.12: **A neural language model using word embeddings.** The model aims to predict the t-th word of a sentence given the n previous words $\{w_{t-n+1}, ..., w_{t-1}\}$ as context. In the network input, each context word w_i is represented by $\boldsymbol{C}(w_i) \in \mathbb{R}^m$, i.e. the w_i-th row of the embedding matrix $\boldsymbol{C}$. Image reproduced from Bengio et al. (2003)

Labeling (SRL), chunking and detection of semantically related words.

Yet, labeled text data are not mandatory to derive powerful semantic representations of the tokens. Indeed, distributional semantics — which hypothesizes that a word is characterized by the company it keeps (Firth, 1957) — can be leveraged to learn directly from raw text that is widely available on the web "for free" (Chelba et al., 2013; Zhu et al., 2015). The unlabeled text is used as training data for a *self-supervised* NLP task by being automatically attributed labels without any human annotation cost. One example of such task is statistical language modeling which aims at estimating the probability of word sequences of arbitrary length. By factorization, the problem is generally reduced to predicting the next word in a sequence given the previous words as context (Bengio et al., 2003). Generating training data for this task is thus straightforward: it just demands to sample some text sequences, keep all the tokens except the last as input and treat the last token as ground truth for the language modeling task. Mixing unlabeled text data with traditionally labeled data for learning token embeddings is one instance of semi-supervised learning (Zhu and Goldberg, 2009). There are two major approaches in semi-supervised learning:

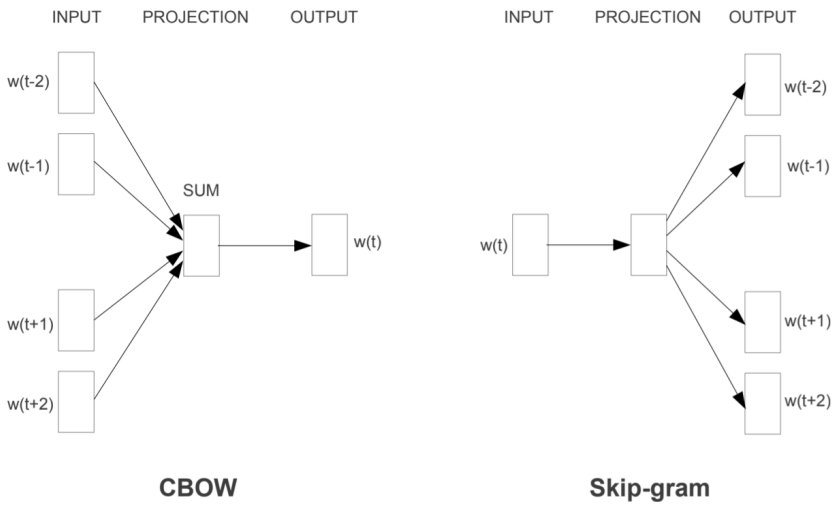


Figure 2.13: **Word2Vec architectures for learning word embeddings.** Two reversed architectures are proposed: CBOW predicts the current word $w(t)$ based on its context words while Skip-gram predicts the surrounding words given the current word $w(t)$. Image reproduced from Mikolov et al. (2013a).

- Learn a single model on both labeled and unlabeled data at the same time (Collobert and Weston, 2008).
- First learn a model to produce high-quality generic token representations on unlabeled text (*pre-training* phase) and then reemploy (parts of) this model for further training on task-specific labeled data.

One of the first successful technique to learn token embeddings from unlabeled text in reasonable training times was undoubtedly *Word2Vec* (Mikolov et al., 2013a,b). They proposed a two-layer neural network that comes in two flavours depicted in Figure 2.13. The Continuous Bag-of-Words (CBOW) architecture predicts the current word $w(t)$ given a fixed-size context, i.e. the words that immediately precede and succeed it. The output of its hidden layer, also called the projection layer, is the average of embedding vectors of all the context words, disregarding the word positions. On the contrary, the Skip-gram architecture predicts which words are in its neighbourhood based on its embedding vector. Once one of these two models is trained, one can extract the weights of the projection layer that correspond to the embeddings of all the vocabulary words. The embedding matrix can then be reused to initialize the embedding layer of an ANN for tackling any downstream NLP task. Word2Vec work further inspires improved embeddings methods. *Glove* (Pennington et al., 2014) incorporates corpus-wide word co-occurrence statistics for model learning instead of relying only on local contexts. The *fastText* approach (Bojanowski et al., 2017) constructs word embeddings by dividing each word in a set of characters n-grams and summing their vector representations. This extension of the Skip-gram model thus takes into account the morphology of words which is profitable for

handling the many rare words in morphologically rich languages, e.g. French or Spanish verbs have more than forty different inflected forms.

Using language models Nevertheless, all these previous self-supervised methods for constructing token representations suffer from the same major drawback. They do not consider the polysemy of words, i.e. the fact that words have different meanings depending on their linguistic context. To remedy this, Peters et al. (2017, 2018) proposed to learn *contextualized* representations of words, i.e. representations that depends on the entire context in which words appear. The representations are issued from language models that are pre-trained on large text corpora. Specifically, *ELMo* (Peters et al., 2018), which stands for Embeddings from Language Models, computes the word representations by a linear combination of the hidden states of a two-layer bidirectional LSTM language model. Like traditional embeddings methods, Peters et al. (2017); McCann et al. (2017) transfer language knowledge to the target NLP tasks by freezing the contextualized representations and inserting them at various stages of existing task-specific NLP systems. They demonstrate that this feature-based approach for transfer learning significantly improves the state-of-the-art for a wide range of challenging language understanding problems.

On the opposite, Howard and Ruder (2018) claim that treating pre-trained embeddings as fixed parameters integrated into task-specific architectures limits the usefulness of embeddings. Rather, they advocate for directly extending the language model with a few additional linear layers in order to tackle the end task. In this paradigm, a minimal amount of parameters needs to be trained from scratch on the target task while the parameters of the underlying language model are only *fine-tuned*. In their paper, they propose novel processes and techniques to retain previous knowledge and avoid catastrophic forgetting during the fine-tuning stage. They conduct experiments on several text classification tasks and show that fine-tuning a basic LSTM language model outperforms the highly engineered state-of-the-art models, particularly with few labeled data. Their work have pushed the NLP community to follow the fine-tuning approach for achieving robust transfer learning.

As yet, all language models were based on recurrent networks — often LSTMs — in order to efficiently model the relationships between tokens. Starting with the Generative Pre-trained Transformer (*GPT*) (Radford et al., 2018), the language models progressively moved to the freshly introduced Transformer architecture. Radford et al. (2018) indeed showed that this change of neural architecture allows the language model to capture longer range linguistic structure, which is particularly helpful for multi-sentence reasoning (Lai et al., 2017). To do so, they pre-trained GPT on a large-scale dataset (1 billion tokens) of long stretches of contiguous text (Zhu et al., 2015). Radford et al. (2018) were then able to tackle a wide range of NLP problems with minimal task-specific customization of the pre-trained model. They not only dealt with text classification tasks like Howard and Ruder (2018) but also with question answering or textual entailment tasks that feature structured inputs and were traditionally solved by introducing a significant amount of task-specific parameters that do not benefit from pre-training. Instead, as illustrated in Figure 2.14, they convert structured inputs into an ordered sequence that GPT can

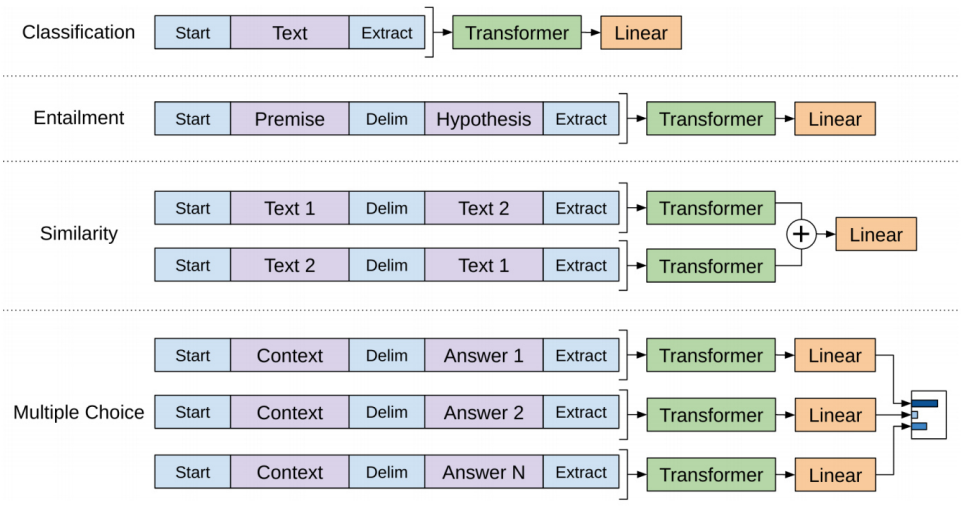


Figure 2.14: **Input transformations for fine-tuning GPT language model on NLP tasks.** For textual entailment (Bowman et al., 2015), sentence similarity (Dolan and Brockett, 2005) and multiple choice question answering (Lai et al., 2017), all the structured inputs are converted into token sequences that the GPT's pre-trained Transformer layers can directly process. The only parameters introduced at the fine-tuning stage belong to the final task-specific linear layer. Image reproduced from Radford et al. (2018).

process without further altering its architecture.

While meaningful for sentence-level tasks, fine-tuning a unidirectional language model like GPT (Radford et al., 2018) may be harmful for token-level problems, such as extractive question answering (Rajpurkar et al., 2016), where it is crucial to incorporate linguistic context from both directions. Devlin et al. (2018) proposed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all Transformer layers. To achieve bidirectional pre-training, they modified the traditional language modeling objective in order to avoid the tokens to indirectly see themselves in later network layers. Such shortcuts would likely lead to uninformative token representations. Instead, they introduced a masked language model (MLM) pre-training objective. As shown in Figure 2.15, it randomly masks 15 % of the tokens from the input and the model is tasked to retrieve which were the original words based on the context. Their model was named Bidirectional Encoder Representations from Transformers (BERT). This change of pre-training objective, from a generative task to a discriminative task, explains that the model is called an encoder. As GPT (Radford et al., 2018), Devlin et al. (2018) resort to special tokens for marking the end of a piece of contiguous text ($[SEP]$) and for getting a high-level representation of the whole token sequence ($[CLS]$) that is used for sentence-level predictions. Segment embeddings are also added to the input token representations in order to better separate two pieces of text to compare, e.g. a paragraph with an associated question about it. In addition to BookCorpus (Zhu et al.,

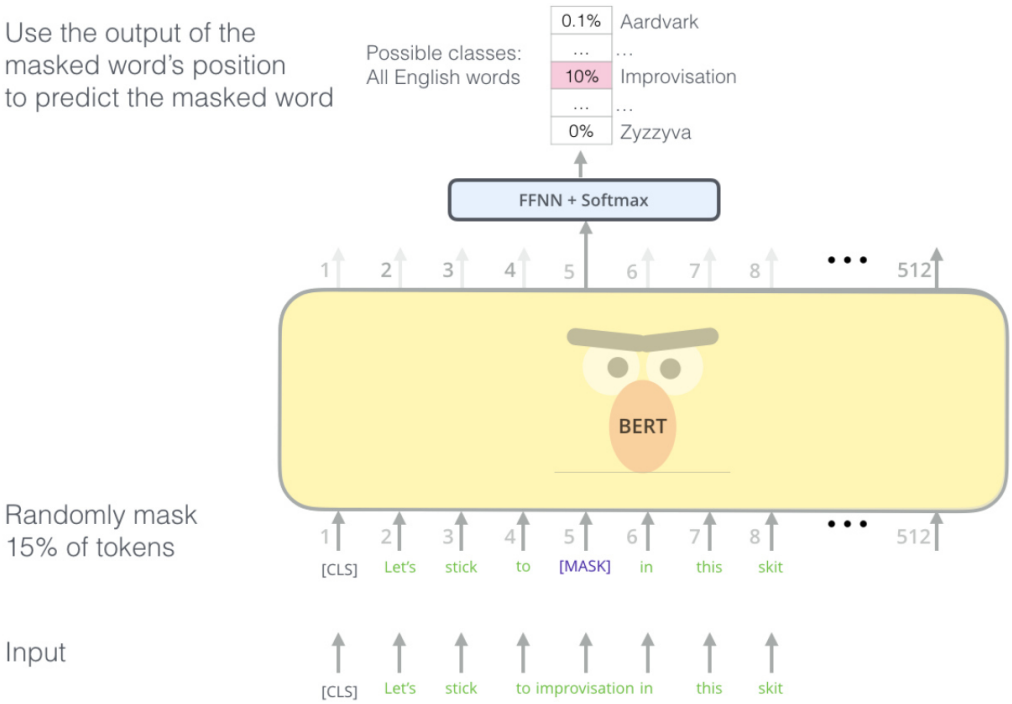


Figure 2.15: **Masked language model objective for pre-training BERT model.** Image reproduced from Alammar (2018b).

2015) used for pre-training GPT, BERT also employed larger-scale and open-domain data, namely 2.5 billion words from English Wikipedia pages. Once BERT is pre-trained, a single fully-connected layer is added on its top and is learned on task-specific labeled data while fine-tuning the encoder weights. Although conceptually simple, BERT significantly improved the state-of-the-art on eleven diverse NLP tasks, including both token-level and sentence-level problems.

BERT's strong empirical results have triggered an amount of works within the NLP community for improving and better understanding bidirectional Transformer based language models. These ongoing works range from mere optimization of its hyper-parameters (e.g. RoBERTa (Liu et al., 2019b)) to new pre-training objectives, larger unlabeled corpora, smaller models to deliver faster training and inference, studies about their learned linguistic knowledge and extensions to languages other than English (Xia et al., 2020).

2.3 Machine learning based information extraction

With the development of machine learning techniques described above, statistical ML approaches have become the mainstream for a wide range of document analysis tasks, including information extraction (Xu et al., 2020b). Indeed, learning a ML model offers a greater adaptability and better generalization capabilities than rule and pattern based IE methods that were detailed in section 2.1. Even if some works use class-specific knowledge

Text	*IO*	*IOB2*	*IOE2*	*IOBES*	*BI*	*IE*	*BIES*
Gamma	I-gene	B-gene	I-gene	B-gene	B-gene	I-gene	B-gene
glutamyl	I-gene	I-gene	I-gene	I-gene	I-gene	I-gene	I-gene
transpeptidase	I-gene	I-gene	E-gene	E-gene	I-gene	E-gene	E-gene
(	O	O	O	O	B-O	E-O	S-O
GGTP	I-gene	B-gene	E-gene	S-gene	B-gene	E-gene	S-gene
)	O	O	O	O	B-O	I-O	B-O
activity	O	O	O	O	I-O	I-O	I-O
in	O	O	O	O	I-O	I-O	I-O
the	O	O	O	O	I-O	I-O	I-O
...	...	...	...	...	...	...	...

Figure 2.16: **Main labeling schemes used for tackling Information Extraction tasks.** In this example, we are interested in extracting instances of gene from the tokenized input text given in the first column. The rest of the columns provide the expected label sequence for each annotation scheme. Image reproduced from Cho et al. (2013).

to improve the extraction efficiency of known document classes (Cheng et al., 2020), most machine learning approaches are class agnostic in order to extract information whatever the document issuer.

Generally, ML methods used for extracting information first decompose a document into various homogeneous parts and then classify all parts, either jointly or independently, according to the information type that they carry (Sarawagi, 2008). The most common decomposition is into a sequence of tokens, by applying a tokenizer to the document text (cf. section 2.2.2.1). The extraction objective then becomes a *sequence labeling* problem, where we need to attribute a label to each token among a set of predefined labels $\mathcal{Y}$. The possible labels comprise the set of target information types $\mathcal{F}$ as well as a special label (`Other` or just `O`) for tokens that do not carry relevant information. For handling multi-token fields, the ground truth set $\mathcal{Y}$ includes subpart field labels to denote the beginning, continuation and end of a unique field value (Ramshaw and Marcus, 1999). These subparts are respectively indicated by the prefixes `B`, `I` and `E` in front of each targeted field. There exists multiple labeling schemes using some or all these subpart labels. The most common choices are illustrated in Figure 2.16. We note that the IOBES annotation scheme also introduces a `S` label for designating single-token fields. As an example, if we expect to extract two fields $\mathcal{F} = \{\text{gene}, \text{protein}\}$ from input documents, the IOBES scheme would induce a label set of size 9:

$$\mathcal{Y} = \{\text{B-gene}, \text{I-gene}, \text{E-gene}, \text{S-gene}, \text{B-protein}, \text{I-protein}, \text{E-protein}, \text{S-protein}, \text{O}\} \quad (2.30)$$

Usually, the choice of an annotation scheme in the IE system is arbitrary, without any proper test. However, Cho et al. (2013) have shown that the different labeling schemes cause little difference in extraction performance.

Albeit less frequently performed, the document text can also be decomposed into segments, i.e. groups of consecutive tokens (Sarawagi, 2008). Each segment is expected to entirely contain a field value, thus allowing to handle multi-token entities without expanding the set of labels $\mathcal{Y}$. The segmentation can be performed following a manually

encoded logic, e.g. by grouping close tokens when the text is spatially distributed as in invoices (Aslan et al., 2016). The decomposition into segments can also be learned while performing the classification of segments (Sarawagi, 2006).

Once the document text is decomposed into tokens or segments, a ML model is tasked to tag each document part with one label from $\mathcal{Y}$. To that end, the model exploits sets of *features* that capture various properties of each document part and possibly the context in which it lies, i.e. its nearby tokens or segments. We denote the feature vectors that represent the document tokens or segments by $\{\boldsymbol{x}_i\}_{i=1...k}$, where k is the number of parts in the document. A wide variety of ML models have been proposed in the literature for classifying information types.

2.3.1 Feature-based approaches

In the early days of ML based information extraction, the prominent methods for classifying document parts were using models with limited expressiveness. The low model complexity was mitigated by manually designing highly informative features of each token or segment (Sarawagi, 2008). While their precise forms depend on the extraction objectives, the feature sets were gathering recurring clue types. For describing a token, its text value was used to derive categorical or boolean features by checking if the token belongs to vocabularies of domain-specific keywords and entities like cities or persons (Han et al., 2003). For less rigid data types likes dates or amounts, matching against a set of regular expressions may also be performed (Schuster et al., 2013a). Some orthographic properties were generally included in the token features, such as capitalization patterns, the presences of special characters like digits or punctuation and the numbers of characters (Peng and McCallum, 2006). Syntax features may be added, classically with the detection of part-of-speech tags, i.e. whether a word is a noun, adjective, verb or adverb (Htay and Lynn, 2013). For Visually Rich Documents (VRDs), layout features such as font types and text alignments were generally incorporated (Zhu et al., 2007) as well as image features, e.g. Histograms of Oriented Gradient (HOG) vectors for detecting company logos (Aslan et al., 2016). Some context knowledge can finally be added to a token's representation by inserting the features of its nearest neighbors (Palm et al., 2017b). When describing a segment, a feature usually captures joint properties of all the tokens belonging to the segment. For instance, Aslan et al. (2016) represent a segment with tf-idf vectors that enumerate all the words of the segment.

After having performed feature extraction, a ML model is applied to the feature vectors $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1...k}$ of a document to determine the labels $\boldsymbol{y} = \{y_i\}_{i=1...k}$ of its tokens or segments. The employed models are seamlessly applicable to both token-level and segment-level feature vectors. Therefore, for sake of readability, we only mention token classification in the following but the same applies for segments.

One simple IE approach is to independently predict the label $y_i \in \mathcal{Y}$ of a token $i \in \{1, ..., k\}$ given its feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$. The traditionally used classifiers when considering this approach are logistic regression (Aslan et al., 2016), kernel-based (Han et al., 2003) and tree-based models (Holt and Chisholm, 2018). However, even if there exists

interactions between the tokens of a document when inserting context knowledge into feature vectors, the labels predicted by such models are independent across a document. This behaviour is prejudicial for typical extraction tasks where the labels of adjacent tokens are rarely independent, e.g. in the Figure 2.16, the token `glutamyl` is highly correlated to the previous token `Gamma` that marks the beginning of a gene instance.

To remedy the decision independence across tokens, probabilistic graphical models (Lafferty et al., 2001) were proposed to extract information from a document. We can cite the applications of Hidden Markov Models (HMMs) (Seymore et al., 1999), Maximum Entropy Markov Models (MEMMs) (McCallum et al., 2000) and, most notably, Conditional Random Fields (CRFs) (Peng and McCallum, 2006; Zhu et al., 2007; Chen et al., 2016) that have empirically been found to be superior to the first two methods (Sarawagi, 2008). All these graphical methods jointly model the labels of document tokens given their feature vectors. They are theoretically able to express label dependencies across the whole token sequence of a document. However, their training and inference times grow exponentially with the size of the label set $\mathcal{Y}$ and the number of tokens k in the document. Therefore, in order to assure their tractability for practical IE tasks, their dependency graph are usually restricted to simple structures like linear chains. This later structure connects a token only with its immediate neighbors in the document sequence.

Except kernel-based methods which employ max-margin training (Taskar et al., 2004), all models mentioned in this section are traditionally learned through maximum likelihood estimation (Sarawagi, 2008). This model parameter search is performed by maximizing the logarithm of the model's output probabilities $P(\boldsymbol{y}|\boldsymbol{X})$ when observed on the training documents. For models that independently classify the tokens, i.e. $P(\boldsymbol{y}|\boldsymbol{X}) = \prod_{i=1}^{k} P(y_i|\boldsymbol{X})$, maximizing the log likelihood is equivalent to minimizing the sum of the cross-entropy loss — defined in equation (2.5) — over each document token.

Even if some extraction logic is learned by the ML model compared to rule-based methods, feature-based approaches still require substantial engineering efforts to design features that are informative enough to lead to accurate model predictions. This observation is particularly true for fields that exhibit important content variability across entities and documents. For instance, in the experiments of Zhu et al. (2007) performed on receipt documents, the extraction performance reported for the vendor name is significantly lower than more structured fields like the issuing date, the credit card and phone numbers. Finally, the feature sets are generally dependent on the application domain, making the feature-based approaches not easily transferable between extraction needs without additional manual costs.

While feature-based approaches are still proposed for tackling IE tasks (Szegedi et al., 2020), most ML methods have now turned to deep models that learn from restricted sets of features whose do not imply significant human intervention.

2.3.2 Deep models

Like for many tasks in the AI field (O'Mahony et al., 2019; Shin and Balasingham, 2017), the IE practitioners have recently transitioned from highly feature-based ML methods

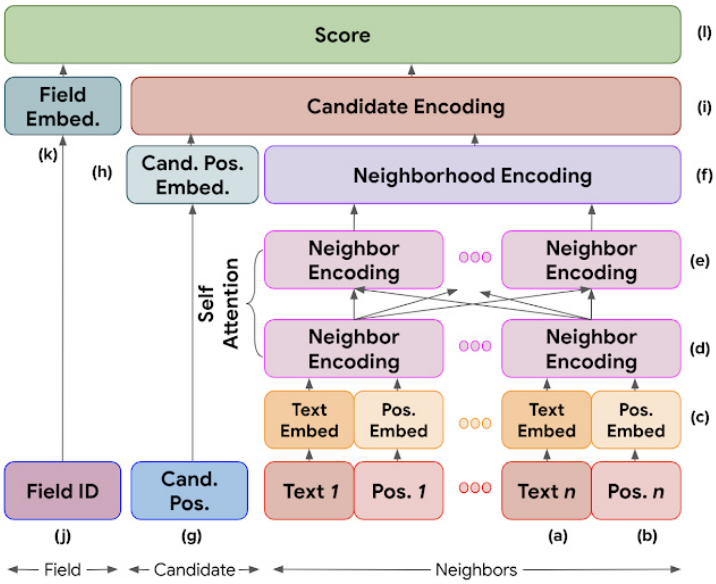


Figure 2.17: **Deep neural model ranking field candidates using local contexts.** The model produces contextualized representations of field candidates (= *Cand.*) by first combining embedding (= *Embed.*) vectors of their N token neighbors and then comparing the neighborhood encodings with the own embeddings of candidates. Image reproduced from Majumder et al. (2020).

to deep learning models that drastically reduce the engineering efforts while improving extraction performance. Such deep models, that are almost exclusively neural networks, can be viewed as the succession of an *encoder* and a *decoder*. The encoder is applied to the feature vectors of document tokens to produce high-level meaningful representations of tokens and their context. These computed representations are then processed by the decoder which is in charge of attributing the information labels to tokens.

One major component of deep neural networks solving IE problems is the embedding layer that transforms the vocabulary index of a token to a dense continuous vector as detailed in the section 2.2.2.2. Using a single feature that is obtained without any cost once the vocabulary has been constructed, we let the model derive linguistic representations of tokens that are helpful for achieving the IE task. The token embeddings are then combined in later layers of the encoder to learn contextualized document-level representations of tokens. The combination of embeddings can be performed locally using a fixed-size neighborhood around each document element. As illustrated in Figure 2.17, Majumder et al. (2020) rank field candidates — generated from high-recall data type specific detectors — by firstly mixing embeddings of their neighbors, i.e. tokens that appear to the left or slightly above the candidates on the document page. However, since Majumder et al. (2020) restrict the candidates neighbourhoods to fixed-size sets of close tokens, their model cannot capture long-range dependencies between the tokens of the documents. This may be prejudicial for extracting certain information types such

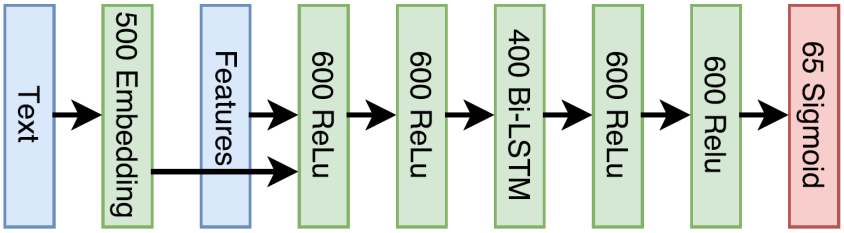


Figure 2.18: **Deep RNN based token classifier used for extracting information.** In this architecture, each token is first encoded into 500 dimensional textual embeddings alongside other features. Several neural network layers are then applied to the token representations, including a bidirectional LSTM layer, before tagging each token with one or several of the 65 classes corresponding to the targeted information types. Image reproduced from Palm et al. (2017b).

as tabular fields where some of the table cells can be relatively distant from the headers explaining the meaning of the rows and columns.

As in other IE tasks such as Named Entity Recognition (NER) (Yadav and Bethard, 2018), the first deep models leveraging document-level representations of tokens were based on Recurrent Neural Network (RNN) encoders (Palm et al., 2017b; Jiang et al., 2019). Such models extract information by processing a document as a sequence of tokens. By recursively computing the token representations across the sequence, the RNN layers, particularly when using the powerful LSTM and GRU cells, are able to store in their hidden states unbounded dependencies between the document tokens. The deep contextualized tokens representations are obtained by stacking multiple neural networks layers, not necessarily all of them being recurrent layers. For instance, as depicted in Figure 2.18, the encoder from Palm et al. (2017b) employs a single central BLSTM layer that is surrounded by four fully-connected layers using a ReLU activation. The final layer of the network is a fully-connected layer in charge of performing the token classification. For Palm et al. (2017b), this decoder layer has an output size of 2×65, where 65 is the number of classes in $\mathcal{Y}$ (32 target fields using the IOB2 labeling scheme). The layer is provided with the sigmoid activation in order to obtain an independently predicted probability for each class of $\mathcal{Y}$, assuming that the classes are not mutually exclusive. Yet, in the general case, a token does not belong to multiple fields and, *a fortiori*, to multiple output classes. Therefore, a common alternative is to employ a flat output layer of size $|\mathcal{Y}|$ with a softmax activation to produce exclusive class probabilities. In order to guarantee more structured predictions, the raw values predicted by this last fully-connected layer, i.e. before applying any activation function, may also serve as high-level feature functions for a graphical model such as CRFs (Jiang et al., 2019).

2.3.2.1 Multi-modal encoders for visually rich documents

As outlined in the subsection 1.3.1, a number of business document types are Visually Rich Documents (VRDs). For solving many document analysis tasks including IE, this characteristic implies to not only understand the textual values of tokens but also the layout and image modalities of such documents. To that end, the first IE works employing deep models were enhancing the token features with positional representations. Palm et al. (2017b) add the two dimensional coordinates of the token bounding boxes that are normalized with the page dimensions while Jiang et al. (2019) uses both learned positional embeddings and Transformer's sinusoidal positional vectors. Yet, in both works, the subsequent RNN encoder in charge of constructing the contextual representations of tokens is operating on an uni-dimensional arrangement of tokens. This model architecture forces to organize the tokens of a two dimensional document into a spurious 1D order, when there is none naturally. Traditionally, the tokens are arranged in a top-left to bottom-right order similarly to the reading order of plain text in most natural languages (Palm et al., 2017b). However, this choice induces that for text blocks spread on multiple lines like the issuer address, the tokens are not contiguous in the resulting sequence which is suboptimal for later detecting fields contained in these blocks (Hong et al., 2021).

To efficiently encode VRDs, the subsequent IE works have proposed document models and neural architectures that explicitly consider the multi-modality of business documents. Inspired by related document analysis tasks (Yang et al., 2017), two main approaches have emerged.

Graph approaches First, various IE practitioners have represented documents with graphs, where each node corresponds to a token or a group of spatially close tokens (Lohani et al., 2018; Liu et al., 2019a; Holeček et al., 2019; Qian et al., 2019; Krieger et al., 2021). Different strategies for connecting the nodes in the document graphs were proposed. The naive choice is to construct an edge between each pair of nodes (Liu et al., 2019a). Yet, the fully-connected graphs prove to be computationally expensive when the granularity of the graph is fine, e.g. when each token is a node (Lohani et al., 2018). Therefore, most IE works using graph encoders employ partially connected structures. Considering that spatially distant nodes are less likely to depend on each other than close nodes, a node is thus connected to only its closest nodes, usually with at most one edge in each cardinal connection (Lohani et al., 2018; Gal et al., 2020). The Figure 2.19 illustrates the representation of a receipt document with a locally-connected graph. Finally, Yu et al. (2021) suggest to automatically learn the graph structure while learning to perform information extraction. Once the document is modeled as a graph, multiple neural network layers involving graph convolution, recurrence or self-attention mechanisms are applied to hierarchically learn 2D contextualized representations of nodes.

Regular grid approaches Secondly, some works have decided to represent a document page as a regularly shaped 2D grid on which the tokens are embedded with their features (Katti et al., 2018; Denk and Reisswig, 2019; Zhao et al., 2019; Dang and Thanh,

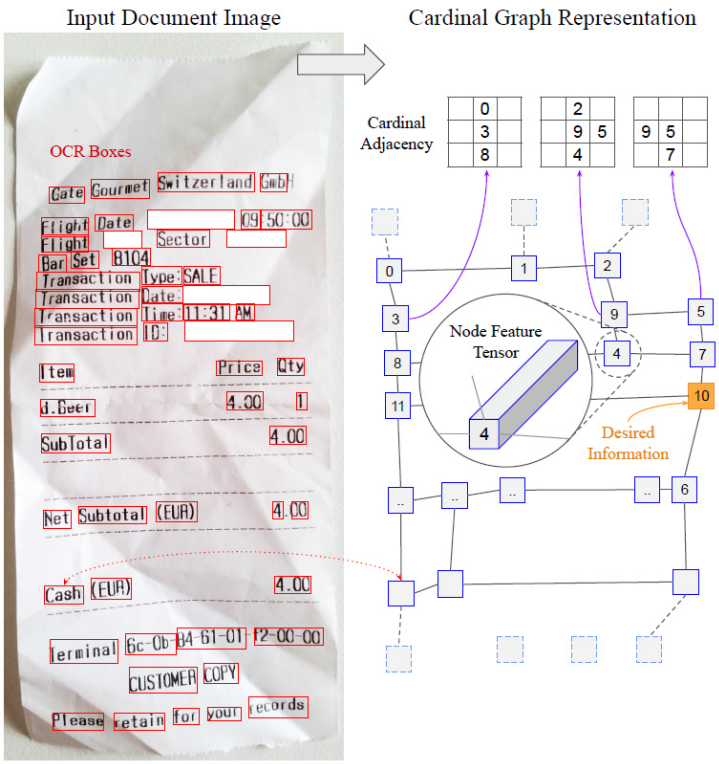
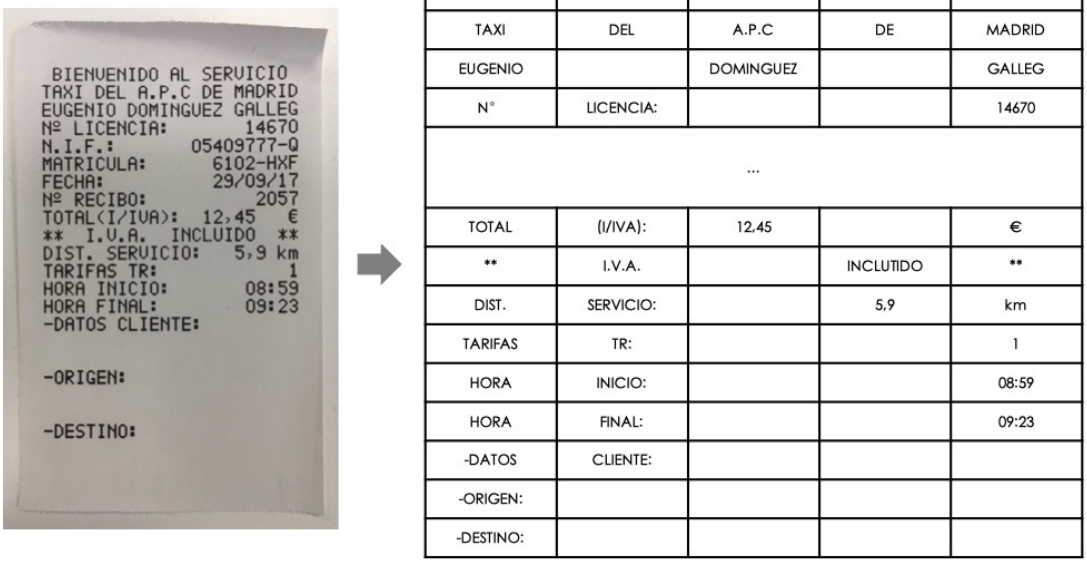


Figure 2.19: **Graph representation of a Visually Rich Document (VRD).** The receipt document is represented as an undirected graph where each word recognized by the OCR engine is a node. Each node is only connected to its closest words in the top, bottom, left and right directions. Image reproduced from Gal et al. (2020).

2019). Aiming to preserve the relative spatial relationships between tokens, this representation is usually obtained by downscaling the original document image such as each grid cell contains at most one token. The Figure 2.20 illustrates the grid representation of business documents. From these image-like models, Convolutional Neural Networks (CNNs) are then employed to obtain the contextualized token representations. In order to capture multi-scale patterns in the input documents, these encoders either resort to dilated convolutions (Zhao et al., 2019) or to a U-shaped architecture that alternates a contracting path and an expansive path (Katti et al., 2018; Denk and Reisswig, 2019; Dang and Thanh, 2019).

Since there are no direct comparisons in the literature, it is difficult to conclude which approach between graph-based and regular grid based approaches is the most efficient to learn token representations that consider the layout modality. However, as outlined by their authors, both multi-modal encoders outperform uni-dimensional RNNs for extracting information from VRDs (Katti et al., 2018; Zhao et al., 2019; Liu et al., 2019a; Castilho, 2020).



	BIEVENIDO	AL	SERVICIO	
TAXI	DEL	A.P.C	DE	MADRID
EUGENIO		DOMINGUEZ		GALLEG
N°	LICENCIA:			14670
...				
TOTAL	(I/IVA):	12,45		€
**	I.V.A.		INCLUTIDO	**
DIST.	SERVICIO:		5,9	km
TARIFAS	TR:			1
HORA	INICIO:			08:59
HORA	FINAL:			09:23
-DATOS	CLIENTE:			
-ORIGEN:				
-DESTINO:				

Figure 2.20: **Regular grid representation of a Visually Rich Document (VRD).** The receipt document is transformed into an image-like representation that embeds its tokens and on which standard convolution layers can be later applied. Image reproduced from Zhao et al. (2019).

Integrating the image modality In addition to better understanding the document layout, Katti et al. (2018); Kerroumi et al. (2020) also include the pixel values of the original pages in the input features. Such integration of the image modality is helpful for capturing clues that are not conveyed by the text such as figures, table ruling lines, logos and stamps. Unsurprisingly, Sarkhel and Nandi (2021) show that leveraging the image modality is beneficial for IE from VRDs such as emails, research papers and news articles.

Similarly to text-only models, a decoder made of one or a few fully-connected layers is added on top of these multi-modal encoders in order to classify the tokens according to their carried information. Even if the tokens are not arranged as an unidimensional sequence in the input of such encoders, the document text still needs to be correctly serialized for the decoder in order to detect fields that are spread over multiple tokens. Therefore, the sequence labeling term is still valid for describing this information extraction paradigm where all the document tokens undergo classification. In the following, we use interchangeably the terms of sequence labeling and token classification based IE for referring to such approaches.

2.4 Information extraction datasets

Due to the sensitivity of data contained in business documents, most prior works in information extraction are carried out on in-house datasets which are not publicly released (Motahari et al., 2021). Unfortunately, such document corpora scarcity prevents IE re-

searchers to extensively compare their proposed methods, thus hindering progress in this field. We below list the most popular public datasets that have been annotated for IE tasks, focusing on datasets that exhibit a diversity of document layouts and issuers:

- Scanned Receipts OCR and Information Extraction (SROIE) dataset (Huang et al., 2019)[9]. This dataset gathers around 1,000 restaurant receipts and its IE task aims at retrieving the name and address of the company issuing the receipt, the total amount and date. Further details about the dataset and task alongside document samples will be given in the next chapters since this resource is massively used for training and evaluating our extraction models.

- CORD: A Consolidated Receipt Dataset for Post-OCR Parsing (Park et al., 2019)[10]. This dataset also puts together 1,000 restaurant receipts. Unlike SROIE which is annotated for just 4 fields, the target information schema is fine-grained and hierarchical with 4 categories and 30 fields to extract. For example, for each menu ordered in the restaurant, one must recognize its identification number, price, quantity, unit and total amounts. However, due to privacy reasons, most of the receipt text is blurred to render it unrecognizable. This limitation makes the IE task less representative of real-world applications.

- Kleister (Graliński et al., 2020). This benchmark is composed of two corpora of multi-page and long documents (several thousands of words on average), namely 540 Non-Disclosure Agreements (NDAs)[11] and 2,778 reports of charities [12]. These documents are annotated for extracting 4 and 8 flat fields, respectively. Each ground truth field value is normalized according to its type, e.g. the ISO 8601 format for the effective date of the NDA.

- Very recently, Chua and Duffy (2021) have mentioned the idea to manually annotate the invoices of the RVL-CDIP dataset (Harley et al., 2015) in order to perform IE tasks. The aim would be to extract the invoiced items as well as the date, number and total amount of the invoice. On their GitHub page[13], they have already released 869 annotations up to April 21st, 2021. Yet, there are still no proper resources that document these annotations and report model results on this dataset.

[9] https://rrc.cvc.uab.es/?ch=13
[10] https://github.com/clovaai/cord
[11] https://github.com/applicaai/kleister-nda
[12] https://github.com/applicaai/kleister-charity
[13] https://github.com/deepcpcfg/datasets

Chapter 3

Information extraction through deep token classifiers

Chapter abstract

The first deep learning models proposed to extract information from business documents were token classifiers which determine the information type carried by each token of the documents. Prior work had shown that those models reach remarkable extraction performance even when facing a rich diversity of incoming documents.

In this chapter, we conduct experiments to assess the behaviour of deep token classifiers in complementary IE settings. We propose an extraction model based on a Recurrent Neural Network (RNN) that operates at the word level. We evaluate our models on two in-house datasets of real-world documents annotated to extract structured information as well as on SROIE, a public IE dataset of receipts. Firstly, we found that the token classifiers are able to extract tabular information from document issuers and layouts that were not seen at training time. Moreover, we show that including fine-grained textual and layout features into the representations of the document words is primordial for successful extraction. Yet, the token classifiers learn to extract information from token level labels that are not naturally available and that must be deduced from the ground truth extraction schemas. In the general case, we have not access to the position of the information within the documents to automatically construct the token labels. The labeling process is then likely to introduce noise in the classifier supervision. We show that such lowering of the label quality can significantly reduce the extraction performance of an IE system based on token classifiers.

The work in this chapter has led to the publication of a conference paper:

- Clément Sage, Alexandre Aussem, Haytham Elghazel, Véronique Eglin, and Jérémy Espinas. Recurrent neural network approach for table field extraction in business documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1308–1313, Sydney, September 2019. IEEE.

Contents

3.1 Introduction

As underlined in Chapter 2, deep learning methods have lately became approaches of choice for extracting information from business documents. Indeed, unlike rule-based approaches, deep models are able to extract complex information structures even for document classes that have not yet been seen. They also require significantly less engineering efforts than shallow learning models that heavily rely on hand-crafted features.

Following the seminal work of Palm et al. (2017b), most of the deep models used for IE resort to classifiers that predict the information type of each document token. As detailed in the section 2.3.2, such classifiers are made of a neural encoder that learns powerful contextualized representations of the document tokens. A decoder is then applied to these representations to obtain the information classes of tokens. In their work, Palm et al. (2017b) utilize a RNN encoder that operates on sequences of document words while the decoder is a single fully-connected layer with a sigmoid activation function. Their deep neural network was empirically shown to outperform a strong logistic regression baseline where the feature vectors were enhanced with local context knowledge. The difference of performance was particularly striking when the extraction models were assessed on invoices whose layouts had not been seen at training time. However, they have only evaluated their approaches on single-value fields such as the invoice number and the total amount. Therefore, in our work, we first study whether the deep IE models behave similarly when dealing with fields that can appear an arbitrary number of times in the extraction schema, the number of field occurrences being unknown beforehand and depending on the input document. To that end, we evaluate deep classifiers when extracting

tabular fields from purchase orders that were processed on the Esker infrastructure. We compare a word level RNN based model with a feedforward neural network baseline where local context knowledge is introduced in the feature sets. The comparison is performed according to two splitting methods of the set of collected purchase orders in order to study the ability of both IE models to generalize to new document layouts (section 3.4.1).

We then conduct in section 3.4.2 ablation experiments on the input representations of words in order to highlight the essential components of the feature vectors among the textual and layout modalities.

By design, these deep classifiers learn to extract information from token level supervision, i.e. from labels that describe the information types carried by each token of the document. Such ground truth can be seamlessly obtained when knowing where the field instances of the extraction schema are located in the document. When experimenting on documents collected from Esker's automation platform, we have access to the position of the extracted field instances since this is a byproduct of another IE system deployed on this platform (see section 3.3.1 for more details). However, in the general case, the position of extracted information within the document is missing in the ground truth. For instance, this is the case for three out of the four public IE datasets mentioned in the section 2.4, namely Huang et al. (2019); Graliński et al. (2020); Chua and Duffy (2021). Without knowledge of the field positions, two possibilities remain for training a token classifier to extract information.

The first option is to manually annotate the position of fields based on their textual value given in the extraction schemas. Yet, this is time-consuming and requires significant domain expertise. For example, Katti et al. (2018) have declared that "[c]onsiderable efforts were spent to ensure that the labels are correct". Moreover, this choice is not scalable since human annotation efforts must be repeated each time we desire to process new fields or document types.

The second idea is to automatically generate the token labels from the filled extraction schemas. This process allows to annotate a high volume of documents without significantly involving domain specific experts. By automatically generating training labels, Palm et al. (2017b) gather 326,471 documents annotated for further token classifier learning whereas Katti et al. (2018) limit themselves to 12k invoices due to intensive manual labeling. Yet, deducing the position of field instances in the documents only from their textual values given in the extraction schemas is, most of the time, not trivial and error-free. Besides OCR errors which are hard to recover from, there are two main reasons which make automatic labeling a tricky task. These hurdles were exposed in the section 1.3.3 and are namely the normalization of fields and the fact that a field value may have multiple occurrences in the document text. With normalization according to their data types, the field instances may not appear verbatim in the text of the document. The normalization of information therefore forces to develop domain specific parsers in order to retrieve the tokens matching a field value. This is particularly challenging for data types presenting a large diversity of formats across documents, e.g. dates. In turn, multiple groups of document tokens can share the textual value of an extracted field while being semantically distinct, e.g. the token 1 may refer to both a street number and

a product quantity depending on its location on the document. Hence, having multiple text occurrences impose to resort to additional heuristics to perform disambiguation when labeling tokens. Even with careful attention, a significant amount of noise is generally introduced in the training annotations when automatically labeling document tokens from the extraction schemas (Tata et al., 2021). For example, Palm et al. (2017b) report that they were able to find only 88 % of the field values in their dataset, with as few as 81 % of matched values for the invoice dates. These recall figures mean that, for many tokens, their labels was mistakenly attributed to the `Other` class rather to an actual field class. These labeling errors act as an upper limit for the performance of the extraction system, no matter how effective is the underlying token classifier.

Consequently, the final experiments reported in section 3.4.3 deal with estimating the impact of token label quality on the extraction performance of an IE system. To that end, we compare performance when the token classifiers learn from two different sets of token labels. For the first set of ground truths, we allow to exploit the position of the field instances in the document, thus achieving near-perfect labeling of the document tokens. In the second case, we label tokens only from the textual value of information, introducing significantly more noise in the ground truth used by the token classifiers. We conduct experiments on both Esker's private data and the public SROIE dataset (Huang et al., 2019) and we demonstrate that a token classifier based system could perform significantly worse when the model is learning from labels which were not derived with the help of the information position.

Before delving into the experiments and their results (section 3.4), we first detail our extraction models (section 3.2) as well as the datasets on which they are trained and evaluated (section 3.3).

3.2 Models

As illustrated in the Figure 3.1 for extracting products from Esker's purchase orders, our IE approaches consist of a token classifier followed by a post-processing module. Like Palm et al. (2017b), we tokenize the document text at the word level.

3.2.1 Token classifiers

Our token classifiers are composed of a feature extractor stage that delivers initial representations of the tokens, an encoder and a decoder. Each model part takes as input the output of the previous part.

3.2.1.1 Token representations

We start by deriving independent representations of the document tokens, i.e. of words in our case. We denote the representation of a word w_i by the vector r_i, where $i = 1...k$ and k is the number of words in the document. As shown in the Figure 3.2, the representations

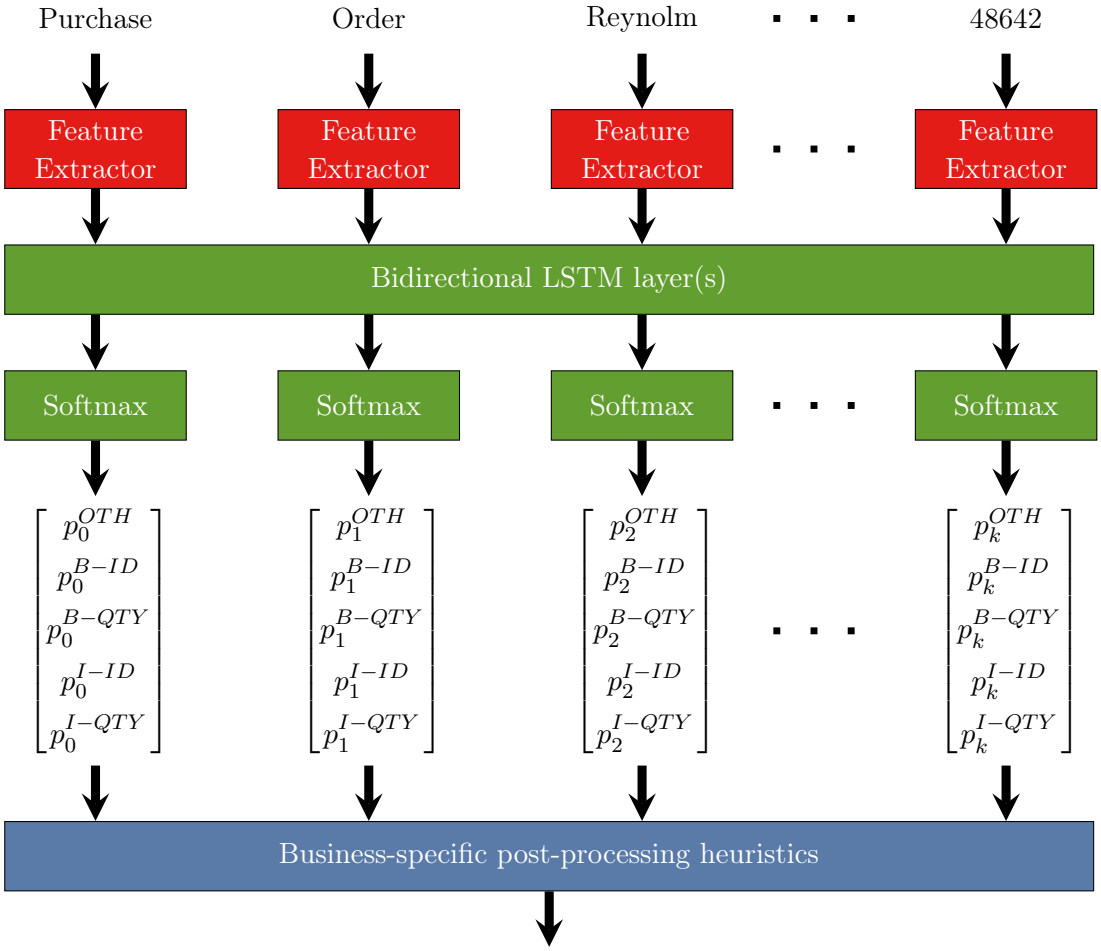


Figure 3.1: **Our extraction method based on a word RNN classifier.** The document text is tokenized at the word level and passes through a feature extractor stage to obtain independent representations of the k words. These representations are then injected into an encoder composed of several BLSTM layers to provide context knowledge. A decoder equipped with a single softmax layer is later applied to predict the word probabilities of carrying one of the target information types. In this example, we aim to extract the products contained in purchase orders, each product being constituted of a reference and a quantity (resp. abbreviated `ID` and `QTY`, while the `OTH` class is dedicated to irrelevant information). Note that each field is duplicated into beginning (`B`) and continuation (`I`) classes in order to handle multi-word field values. Finally, some post-processing operations refine the token classifier predictions and pair the reference and quantity instances to form the products.

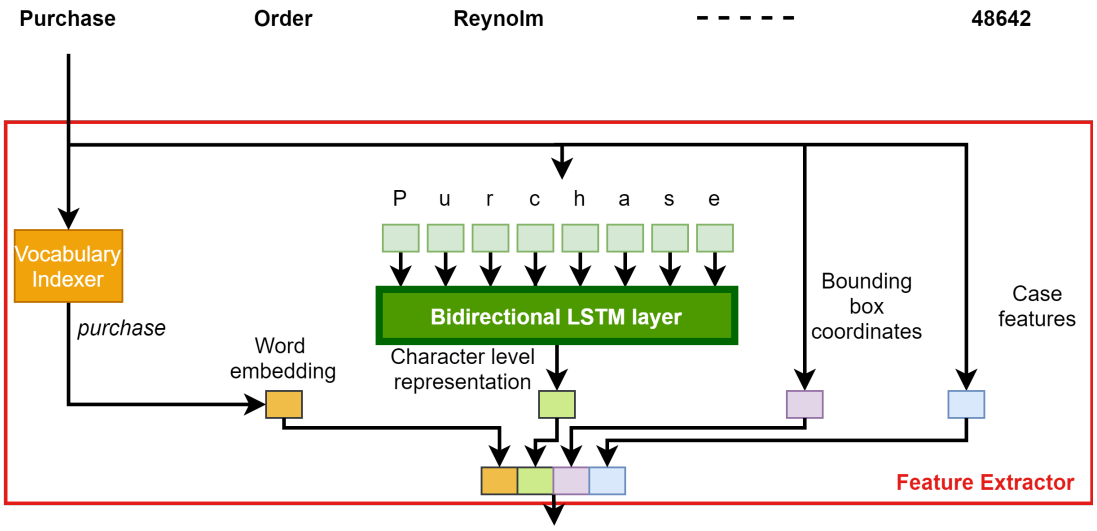


Figure 3.2: **Delving into our feature extractor stage.** The representation of a word is constructed from both textual and layout features. The textual part includes a word-level dense embedding, a fine-grained representation generated from a character-level BLSTM layer as well as case features. The layout part encompasses the 2D coordinates of the word bounding box. All these components are concatenated to form the final word representation.

obtained by the feature extractor stage are constituted of multiple components, from both text and layout modalities.

Word-level text embedding Each word w_i is first represented by a dense continuous embedding denoted by $q_i^w \in \mathbb{R}^m$, where m is the embedding dimension (Bengio et al., 2003). This vector is associated to the vocabulary index of the word. Since the appearing words are highly specific to the type of business documents that are processed, we construct our own vocabularies rather than relying on generic web-based vocabularies such as Pennington et al. (2014)[1].

The conventional approach for building a word vocabulary is to directly enumerate the words that occur within the training set. Yet, in the business field, a large number of words are rare and specific to an issuer or even to a single document, e.g. occurrences of product references, dates and amounts, while we expect that some of them contain relevant information to extract. Hence, if the vocabulary was constituted from the raw training text, a significant portion of words from documents outside the training set would not belong to the vocabulary — we then speak of OOV words —. Such words would all be attributed the same vocabulary index which is dedicated to unknown words. Their textual content would thus not be properly represented in the extraction model.

[1] https://github.com/stanfordnlp/GloVe

To remedy the OOV issue, we normalize the text before attributing the vocabulary indexes to the document words. Firstly, we design a small set of broad text categories which are built for business documents. If a word matches the regular expression corresponding to a category, the vocabulary index of this category is assigned to the word. As it may match multiple categories, we keep only the index of the first matched category. The ordered list of the categories is the following: *DigitSequence*, *ContainsDigitAndAlpha*, *ContainsDigitAndDash*, *ContainsDigitAndSlash*, *ContainsDigitCommaAndPeriod*, *ContainsDigitAndComma*, *ContainsDigitAndPeriod*, *PunctuationSequence*, *URL*, *EmailAddress*. The categories involving digits are rather generic, letting the later neural layers the task to distinguish what these categories may refer to, e.g. integers, floats, dates, phone numbers. Besides, these categories have also the advantage of making the model more robust to OCR failures, as recognition errors on some characters of a word are likely to have little impact on the category matching.

Secondly, the words that haven't matched any of the previous categories are standardized as follows. They are converted to lower case. The punctuation and whitespace characters are removed at the beginning and end of the word. The word vocabulary is finally determined by enumerating the standardized forms of words from the training documents.

Character-level textual representation While being a first step towards efficient representation of the text content, the business categories designed for words are likely to be too broad or incomplete. Also, even after standardization, there still remains a non-negligible amount of rare or OOV words. Therefore, besides word level embeddings, we enrich the representations with finer-grained textual features to cope with the open vocabulary observed within corpora of business documents. Specifically, we follow the C2W model of Ling et al. (2015) to form a textual representation $q_i^c \in \mathbb{R}^m$ at the character level. To that end, we apply a BLSTM layer over the sequence of characters composing the word w_i, each character being modeled with a dense embedding vector. We then concatenate the final hidden states of the forward and backward directions to derive the character-level representation q_i^c of the word w_i.

Additional textual features We also add the number n_i of characters in the word and two 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 (Chiu and Nichols, 2016). We concatenate all these features to form the textual component $r_i^t \in \mathbb{R}^{2m+3}$ of the word representation r_i:

$$r_i^t = [q_i^w, q_i^c, n_i, \alpha_i, \beta_i] \tag{3.1}$$

Layout features To take into account the document layout, we also compute four spatial features r_i^s for each word w_i. These encompass the 2D coordinates of the top-left and bottom-right edges of the word bounding box, normalized by the height and width of the page. Unlike Palm et al. (2017b), we do not create more complex layout features

like measures of alignment and spacing with neighbouring words. Rather, we rely on the abilities of the subsequent encoder to infer them from the absolute positions.

Finally, we concatenate the spatial component r_i^s with the textual part r_i^t to build the word representation $r_i \in \mathbb{R}^{2m+7}$.

3.2.1.2 Encoders

Next, the words of the document are organized as a unidimensional sequence of length k by reading them in a top-left to bottom-right order. In case of a multi-page document, the sequences of each page are concatenated based on the order of appearance of the pages. The word representations $\{r_i\}_{i=1\ldots k}$ are then fed to one of the two following encoders in order to obtain contextualized representations of the words.

Recurrent encoder We first employ a two-layer BLSTM network over the sequence of word representations. The output of this encoder is thus the hidden states $\{h_i\}_{i=1\ldots k}$ of the final layer, where $h_i \in \mathbb{R}^e$ and e is the number of BLSTM units in each layer. Its recurrent mechanism is theoretically able to capture unbounded dependencies between document words which will be helpful for later extracting information.

Feedforward encoder As a baseline, we also propose a two-layer MLP network, i.e. which is only composed of fully-connected layers. The network is independently applied to each word of the document. For a fair comparison with the recurrent encoder, we provide local context in the word representations $\{r_i\}_{i=1\ldots k}$. Specifically, we concatenate the feature vectors of the closest words in the left, right, top and bottom directions with the original representations. This results in 5 times bigger feature vectors. We also set the number of neurons in each of the two layers such as the total number of trainable parameters is equivalent to the recurrent encoder.

3.2.1.3 Decoder

On top of the encoder outputs, we add a final fully-connected layer with a softmax activation to predict the probabilities $\{p_i\}_{i=1\ldots k}$ of each word to belong to one of the information class, where $p_i \in [0,1]^C$ and C is the number of classes. We follow the IOB2 annotation scheme for the classification (cf. Figure 2.16). The decoder layer has thus $C = 2\,|\mathcal{F}| + 1$ sotfmax units, where $\mathcal{F}$ is the set of targeted fields. For each field $f \in \mathcal{F}$, an output class is dedicated to the words that begin (`B`) a new instance of this field while the `I` class is used for words that are the second and further words of a field instance. The remaining class (`Other` or just `O`) is employed for tagging words that carry irrelevant information.

The word classifiers are trained by minimizing the cross-entropy loss (cf. equation 2.5) which is averaged over all the words of the documents within the batch.

3.2.2 Post-processing module

In practice, the predictions of the token classifier, i.e. the class probabilities $\{p_i\}_{i=1...k}$, are not considered as the final results that will populate the extraction schema. Indeed, the class predictions undergo post-processing steps which take into account the structure of the extraction schema and some business logic. For instance, one can check that the due date of an invoice is chronologically after the date of its emission (Majumder et al., 2020). Palm et al. (2017b) filter out candidates that do not fit the syntax of the field and employ cost functions to evaluate the conformity of predictions with business constraints, e.g. Line Total + Tax Amount = Total Amount.

In our work, we keep the post-processing simple. For each document word, we first look for its highest probability across all the classes in order to predict its class:

$$\hat{y}_i = \underset{c \in \{1,...,C\}}{\arg\max}\, p_i^c \tag{3.2}$$

We then merge the predicted beginning and continuation classes of the same field to construct multi-word field values.

For the fields that are assumed to appear only once in the extraction schema, e.g. the document number or date, we choose as final prediction the candidate for which the word classifier confidence is the highest. This post-processing is used for the SROIE dataset that we describe in the next section.

For the Esker's datasets, we also design specific post-processing steps for gathering the instances of fields that appear in tables within the documents, thus aiming to extract the corresponding structured entities. We assume that each structured entity is located within one or several consecutive table rows while its constituting fields are spread over the columns of the table. Based on this hypothesis, we first consider one of the targeted fields as an anchor field. We preferably choose the field which is the most reliably extracted by the token classifier since the performance on the selected field will act as an upper limit of the global extraction performance measured over the entities. Then, for each remaining field of the table, we pair its predicted instances with the instances of the anchor field. This pairing is performed by minimizing the vertical distances on the document between the field instances of the constituted pairs. We formalize this optimisation problem as a linear sum assignment problem with the vertical distances as the matching costs. We show an example of such an assignment problem in Figure 3.3. As traditionally performed, we solve it with the Hungarian algorithm (Kuhn, 1955). While simple, this field pairing strategy is flawless if the field instances are perfectly extracted by the token classifier. Moreover, unlike vanilla heuristics that would pair a field instance with the closest instance of another type, solving it as an assignment problem is robust to missing or wrong predictions of the classifier.

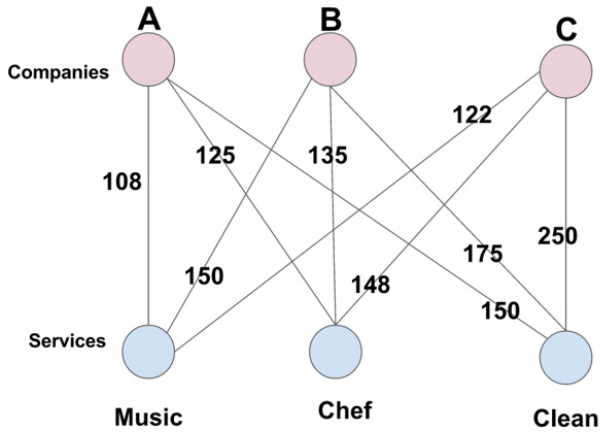


Figure 3.3: **A linear sum assignment problem.** This problem is also known as minimum weight matching in bipartite graphs. A problem instance is described by a matrix $\boldsymbol{C}$, where each element $\boldsymbol{C}_{i,j}$ is the cost of matching the vertex i of the first partite set (here companies) and the vertex j of the second set (here services). The goal is to find a complete assignment of companies to services at a minimal global cost under the constraints that a company can be attributed to at most one service, and vice versa. Here, the companies A, B, C would be respectively attributed to the services Clean, Chef and Music.

3.3 Datasets

We now review the datasets on which we train and evaluate our extraction models based on deep token classifiers. There are three different IE datasets that we name Esker-28k, Esker-47k and SROIE. As their name imply, the first two datasets are constituted of private real-world documents issued from the document automation solutions commercialized by Esker[2]. On the contrary, the SROIE dataset has been publicly released (Huang et al., 2019). For each dataset, we provide some statistics about their documents and targeted extraction schema as well as — real or fictive — samples with their expected extraction results. We also describe the protocol for evaluating the predictions of our IE models.

3.3.1 Esker-28k

We first consider Esker-28k, a dataset of 28,570 purchase orders coming from the Esker's infrastructure. The purchase orders are business documents that contain products or services that the issuers desire to buy from the recipients. The collected purchase orders are originating from 2,818 issuers. Each issuer has a minimum of 3 and a maximum of 31 documents. As we assume that most suppliers have their own layout, controlling their number of documents ensure a high diversity of layouts within the corpus. Although

[2]Unfortunately, we are not allowed to release our in-house datasets.

English is the language of all documents, the dataset is multi-cultural with European and American orders.

From these documents, we are interested in extracting two fields related to the ordered products, namely their reference or ID number and their quantity. These fields appear a variable number of times across the extraction schemas since a purchase order is free to contain any number of ordered products. In this dataset, there are on average 3.7 ID number and 3.5 quantity values per document. The ID number and quantity fields are almost always displayed in a table within such documents, where each (group of) row corresponds to a different product. We show in Figure 3.4 an example of purchase order on which we highlight the two targeted fields.

The dataset gathers both born-digital and scanned purchase orders. For the former, the text is immediately accessible without any error while for the latter, the text is retrieved thanks to an external commercial OCR engine[3]. We take the OCR results as they come without attempting to correct them. The extraction schemas of those documents have been directly filled and validated by the end users of Esker's solutions, assuring a high quality of the ground truth. On these automation solutions, an IE system using class specific knowledge (cf. section 2.1.2) was already in place to help the users to extract information from *known* document classes. This IE system relies on the position of the extracted information in order to properly function. We therefore reuse this knowledge to derive accurate word labels for training our classifiers. We have discarded from the dataset the documents for which we have not retrieved a ground truth field value among the document words. Finally, as a first step toward automated IE, we have restrained to documents that contain ID number and quantity values that are all made of a unique word.

On this dataset, we will conduct experiments to assess the ability of extraction models to generalize to document layouts that were not seen during training. To this end, similarly to Palm et al. (2017b), we split the dataset according to two ways that we name *DocumentLevelSplitting* and *IssuerLevelSplitting*. In the first case, we simply randomly separate the documents in the dataset to constitute the training, validation and test sets. In the second case, the data splitting is performed at the issuer level: all the documents from the issuers in a set of the *IssuerLevelSplitting* experiment then constitute the documents of that set. In order to accurately estimate performance, for each data splitting way, we randomly partition the dataset into five folds and evaluate the models on each fold while using the four remaining folds for its training. Each ensemble of four folds is further independently divided in training and validation sets so that they respectively represent 70 % and 10 % of the whole dataset.

Here, we only evaluate the performance of the word classifiers, i.e. without employing the post-processing steps for predicting extraction schemas. Hence, we employ the standard metrics for sequence labeling approaches which are the precision P, recall R and F1 score F. For each of the two fields and each of the five test folds, we compute these

[3]To be specific, we resort to Nuance OmniPage 19.

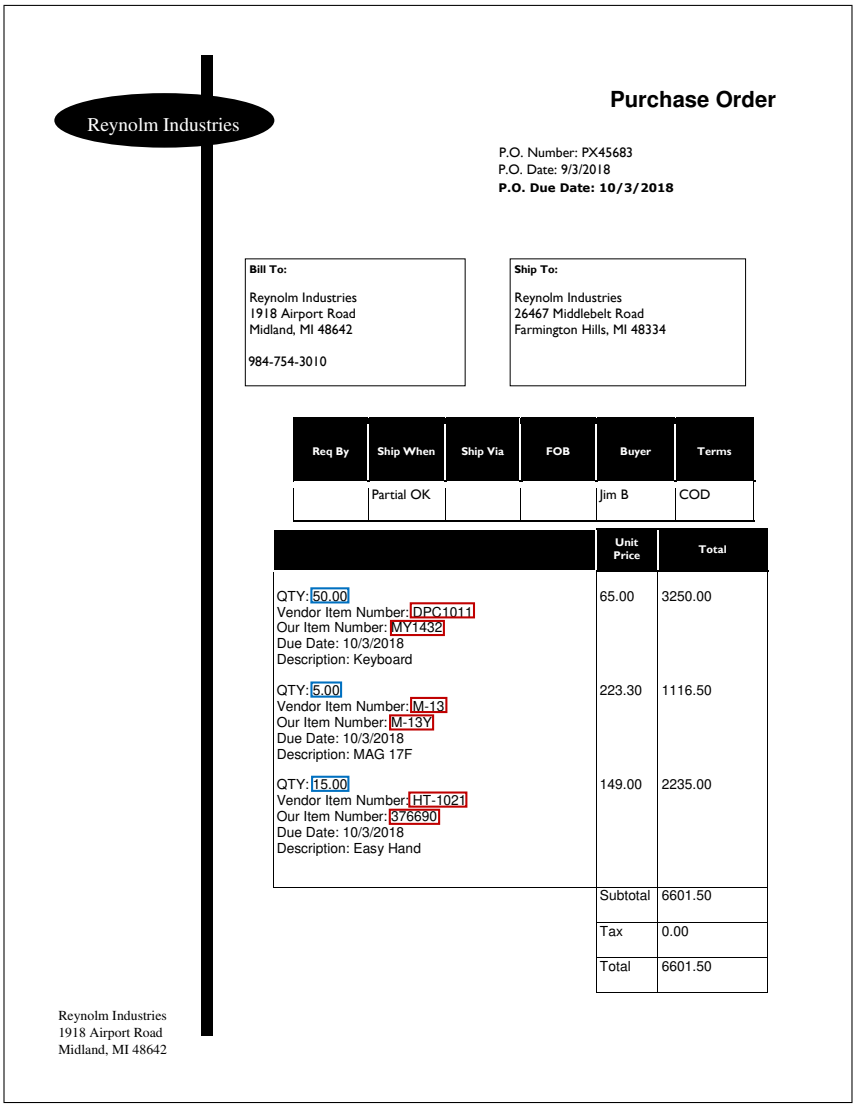


Figure 3.4: **A fictive sample representing the Esker-28k task.** The aim of this dataset is to extract the ID numbers (red) and quantities (blue) of the products contained in purchase orders.

metrics from the word level predictions of the classifiers as follows:

$$P = \frac{tp}{tp + fp} \tag{3.3}$$

$$R = \frac{tp}{tp + fn} \tag{3.4}$$

$$F = \frac{2PR}{P + R} \tag{3.5}$$

where tp, fp, fn are respectively the counts of true positives, false positives and false negatives (Lipton et al., 2014). The precision is the fraction of all positive predictions

that are actual positives, while the recall is the fraction of all actual positives that are predicted to be positive. Finally, the F1 score is the harmonic mean of the precision and the recall and is the metric of choice for comparing two classifiers. For obtaining global metrics, we perform micro averaging: the true positives, false positives and false negatives counters are first summed across the ID number and quantity classes and then, the micro recall and precision are computed from these total counters as in the equations 3.3 and 3.4. The micro F1 score is computed with the micro recall and precision as in equation 3.5. Finally, we report the average over the five test folds for both the field and micro metrics.

3.3.2 Esker-47k

We then constitute Esker-47k, an IE dataset that is significantly more demanding that Esker-28k for various factors. The documents are still purchase orders from Esker's solutions but they are now multi-lingual with not only English but also other European languages such as French, Italian and Spanish. Secondly, there are substantially more document issuers, i.e. 15,905 against 2,818 issuers before. All the issuers have 3 documents in this dataset, resulting in a dataset of 47,715 purchase orders. Concerning the information to extract, the ID number and quantity may now be spread over multiple words of the documents and must be paired together to predict the ordered products. Finally, we now judge the performance of an IE model by comparing the predictions not to the ground truth word labels but directly to the expected extraction schemas. To do so, we adapt the evaluation methodology proposed by Katti et al. (2018) to measure the performance when extracting structured entities like the ordered products. 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 determine the manual post-processing gain with the following edit distance:

$$1 - \frac{\#\text{ deletions} + \#\text{ insertions} + \#\text{ modifications}}{N} \tag{3.6}$$

where N is the number of ground truth instances in the document for this field. The micro averaged gain is calculated by summing the error counters of ID number and quantity fields and applying equation 3.6. 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 3.6. The post-processing gain can be interpreted as a measure of how much work is saved by using an IE model rather than manually doing the extraction. The higher the metric value is, the better is the extraction — 1 being the upper limit— . The value can be negative, meaning that it demands more efforts to correct the errors of the evaluated model than to perform the extraction manually.

Since we have already assessed the layout generalization abilities on Esker-28k, we only split the Esker-47k dataset at the issuer level for creating the training, validation

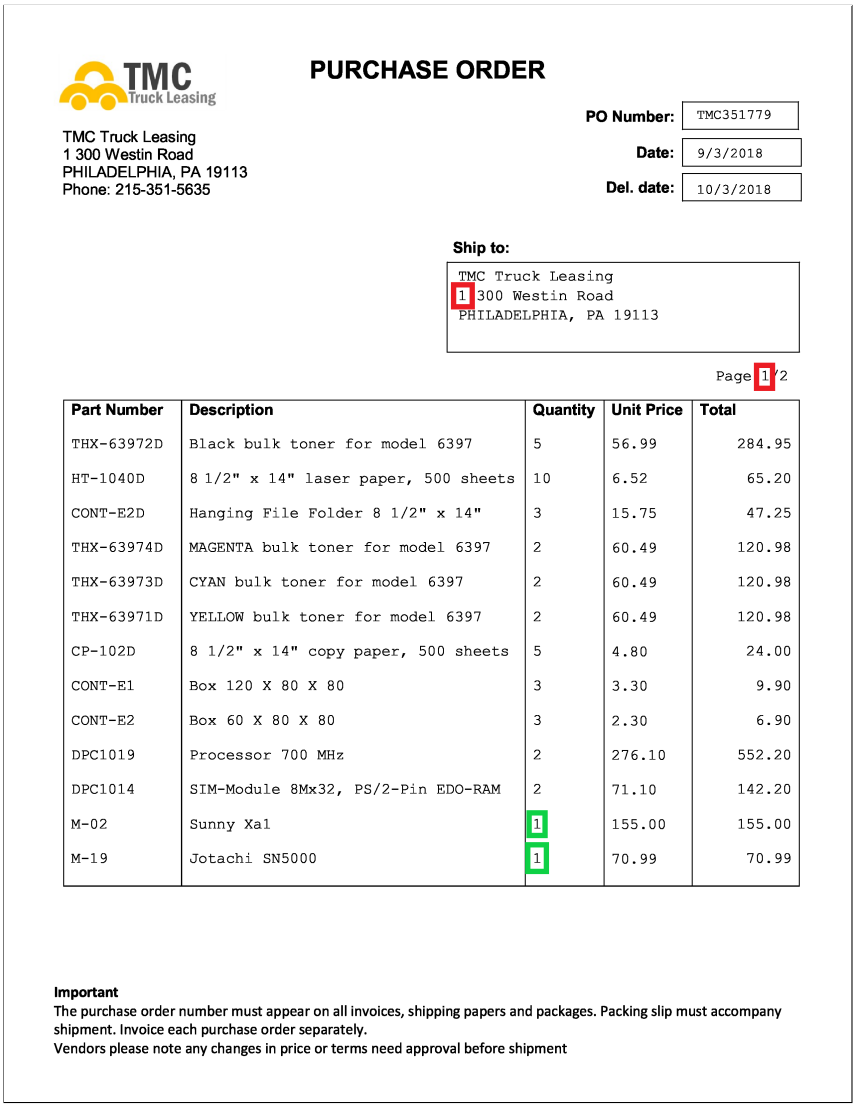
and test sets. These sets are respectively constituted of 70 %, 10% and 20% of all the issuers.

As mentioned in the chapter's introduction, we derive two different sets of word labels in order to study to what extent the label quality impacts the post-processing gains of IE models. In the first case, like for Esker-28k, we allow to leverage the position of the fields to establish the labels of all the words from the training documents. This ensures that the word labels are perfect and are thus not a limiting factor for the extraction performance. In the second case, we only resort to the textual value of the field instances in order to simulate the general case where their position is not provided in the ground truth extraction schemas. The ID Number and quantity are fields whose instances rarely require normalization while being extracted since they almost always appear verbatim in the document. Nevertheless, for those two fields, multiple N-grams of document words may share the textual value of a field instance and thus be candidates for being labeled as a product ID number or quantity. We show this issue of field ambiguity on a sample purchase order in Figure 3.5. We remark that for a single quantity instance, there may be N-gram candidates in the document that are indeed quantities but also that can carry completely different information types such as a page or street number. These later candidates may appear easy to discard from the word labeling process using some business heuristics since they are outside the table of the ordered products. Yet, in real samples of Esker-47k, we observe that there are a great number of quantity candidates that are spatially close to the ground truth occurrences, e.g. ambiguity with description elements and line numbers. For the ID number, there are typically less N-gram candidates for a given field instance since their textual value is highly specific to this field. The most notable situation where there are multiple ID number candidates is when both the vendor and client references are present and are equal. On the whole dataset, there are on average 1.16 and 1.55 N-gram candidates for each ID number and quantity instance, with 23 % and 46 % of the documents having at least one labeling ambiguity for these respective fields. Since we do not know which N-gram candidate semantically corresponds to a field instance, we randomly label one of the occurrence in the document for the second label set. We hypothesize that labeling the documents without the position of the information would introduce a non negligible amount of noise in the word labels that would significantly affects the final extraction performance (Frénay and Verleysen, 2013). We verify this assumption in the section 3.4.3.

3.3.3 SROIE

We finally perform experiments on the public SROIE dataset, which stands for **S**canned **R**eceipts **O**CR and **I**nformation **E**xtraction (Huang et al., 2019). Released during a competition of the ICDAR 2019 conference[4], this dataset encompasses 3 different tasks, namely text localization, text recognition and information extraction. Since the text transcription is out of scope of the PhD thesis, we only tackle the last task that aims to

[4] https://icdar2019.org/competitions-2/



TMC Truck Leasing

PURCHASE ORDER

TMC Truck Leasing
1 300 Westin Road
PHILADELPHIA, PA 19113
Phone: 215-351-5635

PO Number: TMC351779
Date: 9/3/2018
Del. date: 10/3/2018

Ship to:
TMC Truck Leasing
1 300 Westin Road
PHILADELPHIA, PA 19113

Page 1/2

Part Number	Description	Quantity	Unit Price	Total
THX-63972D	Black bulk toner for model 6397	5	56.99	284.95
HT-1040D	8 1/2" x 14" laser paper, 500 sheets	10	6.52	65.20
CONT-E2D	Hanging File Folder 8 1/2" x 14"	3	15.75	47.25
THX-63974D	MAGENTA bulk toner for model 6397	2	60.49	120.98
THX-63973D	CYAN bulk toner for model 6397	2	60.49	120.98
THX-63971D	YELLOW bulk toner for model 6397	2	60.49	120.98
CP-102D	8 1/2" x 14" copy paper, 500 sheets	5	4.80	24.00
CONT-E1	Box 120 X 80 X 80	3	3.30	9.90
CONT-E2	Box 60 X 80 X 80	3	2.30	6.90
DPC1019	Processor 700 MHz	2	276.10	552.20
DPC1014	SIM-Module 8Mx32, PS/2-Pin EDO-RAM	2	71.10	142.20
M-02	Sunny Xa1	1	155.00	155.00
M-19	Jotachi SN5000	1	70.99	70.99

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 3.5: **Which words from the document correspond to a given field instance ?** In this example, we aim to retrieve the words of the document that correspond to the product's quantity 1 in order to derive the training labels of a word classifier based IE model. Without the position of this field instance, there is ambiguity since there are multiple candidates: not only words that are semantically correct (highlighted with green boxes) but also words that carry other information types (in red).

retrieve the name and address of the company issuing the receipt, the total amount and date. We illustrate the task in the Figure 3.6 by showing a sample of the training set and its expected field instances. The dataset gathers 626 receipts for training and 347 receipts for test. We further randomly split the training set to constitute a validation set of 26 receipts. While not stated in Huang et al. (2019), the document issuers are shared between the training and test sets. Nevertheless, the dataset is quite diverse, with 234

Figure 3.6: **A receipt sample illustrating the SROIE task.** In this dataset, we aim to extract the name and address of the company that emits the receipt as well as its date and total amount (Huang et al., 2019).

distinct issuers in the training set according to Majumder et al. (2020).

As ground truth, each training receipt is only given the expected values of the four targeted fields, i.e. neither token level supervision nor the position of fields are provided. Therefore, we derive the IOB2 labels by looking for the receipt words which match the field values. Like the Esker-47k dataset, the fields are not normalized while being extracted. However, as illustrated in the Figure 3.6, the total amount value can appear multiple times within a receipt. Some of its occurrences then refer to other information types, e.g. the amounts without taxes or after rounding. For 72.7% of the receipts from the training and validation sets, there is ambiguity for labeling the total amount, with on average 2.56 N-gram candidates for each total amount instance. To assess the impact of word label quality, we also establish two sets of labels for training our classifiers to extract total

amounts:

- Even if we do not have the actual position of the total amount instances, we rely on strong *a priori* knowledge of this field to reach the best quality possible for the token labels. Among all the field candidates within the training receipt, we select the bottom most occurrence having the keyword `total` in its text line.
- To construct the second label set, we do not exploit any business specific heuristic and we randomly choose one occurrence of the total amount value within the receipt.

There is no ambiguity for the three remaining fields since the company name and address appear only once in the training documents while the date instance appears at least twice in 19 % of the receipts but all occurrences are valid duplicates of the date. Thus, the labels of these three fields are common to the two label sets constructed for the total amount.

We evaluate our models by submitting their predictions over the test set on the web platform of the competition[5]. As Esker-47k, the comparison with the ground truth is made in terms of exact matching of strings. The platform returns the precision, recall and F1 score metrics averaged over the four fields.

To fuel our extraction models, we use the provided OCR results that contain a list of the text segments and their bounding boxes. As noticed by many public submissions in the leaderboard, they contain a number of brittle text recognition errors, e.g. a comma interpreted as a dot. This highly impacts the evaluation results based on exact matching. However, unlike all the current top-scoring submissions, we have not corrected the OCR text of the test receipts for fair comparison between models. Still, we perform fuzzy matching when labeling the training receipts in order to avoid missing some of the field instances. Their order being sometimes faulty, we also re-arrange the text segments from top-to-bottom before tokenizing them at the word level.

3.4 Experiments

For each dataset presented above, we choose the values of model hyper-parameters based on the micro averaged performance on the validation set. We give their specific values in the following sections before exposing the results of the experiments. In all experiments, we stop the training of the word classifiers when their micro performance on the validation set have not improved in the last three epochs. To deal with exploding gradients, we apply gradient norm clipping (Pascanu et al., 2013) with a clipping threshold of 5. All extraction models are implemented using the TensorFlow library (Abadi et al., 2016). The training and evaluation are performed on a single 12Go TITAN X Graphics Processing Unit (GPU). This explains that, for computational reasons, we do not train the models on documents with more than 1,800 words, which amounts to less than 5 % of the training set being put aside for the Esker-28k and Esker-47k datasets. However, we evaluate the

[5] https://rrc.cvc.uab.es/?ch=13&com=evaluation&task=3

models on all documents in the validation and test sets. For the SROIE dataset, we do not discard any training documents since they all contain less than 1,800 words.

3.4.1 Generalization to unknown layouts

We first perform experiments on Esker-28k in order to assess the ability of token classifiers to generalize to unknown documents layouts, i.e. layouts that have not been seen at training time.

Experiment settings For all runs in this section, we use word level text embeddings of size 64 and do not resort to the character-level representations. The word vocabulary is restricted by removing its entries that rarely appear in the training set since they are not helpful for later processing unknown layouts or documents and make the extraction models more likely to overfit. To this end, we list the vocabulary entries by decreasing frequency of occurrence in the training set and we retain only the most frequent entries that, put together, match a minimum percentage of the total number of occurrences. In order to choose the threshold value, we plot the degree of coverage of the training words against the size of the vocabulary. For the *DocumentLevelSplitting* configuration, this leads to the Figure 3.7. We note that the plot has an exponential curvature, with a

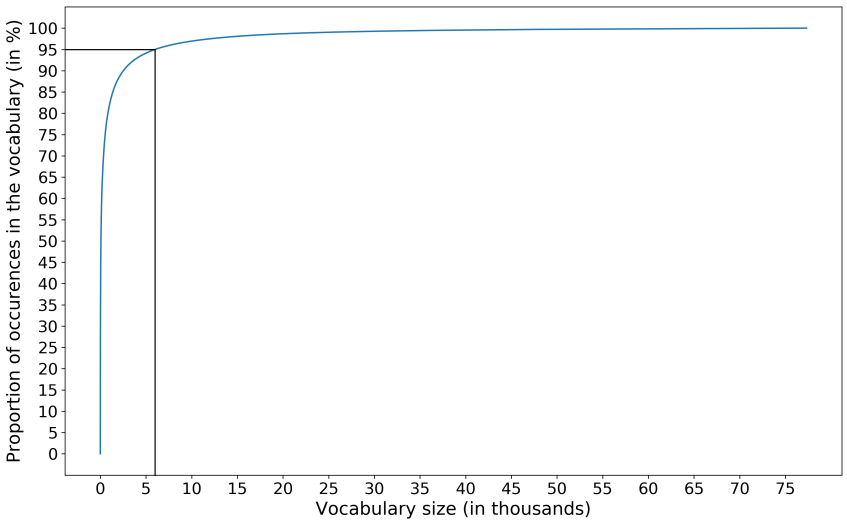


Figure 3.7: **Proportion of the training words that are covered by the vocabulary as a function of its size.** This plot, realized for one of the training set of the *DocumentLevelSplitting* configuration, shows an exponential form with a large number of words that rarely appear in the training documents and can thus be discarded from the word vocabulary.

relatively small fraction of the vocabularies entries that repeatedly appear in the training documents. By selecting the top-K entries that together account for 95 % of the training words, we still cover a large portion of the document text while drastically reducing the

vocabulary size. Indeed, such a thresholding results in a vocabulary of only 6,010 entries while there were originally more than 75k entries. We observe a similar behavior for the *IssuerLevelSplitting* configuration and thus also retain the threshold value of 95 %.

The RNN encoder is composed of two bidirectional LSTM layers of 1,300 cells each with outputs of each direction being concatenated while the baseline feedforward network has two fully-connected layers with 6,947 and 1,300 ReLU units.

All the network weights are randomly initialized following a uniform distribution of amplitude 0.05. The cross-entropy loss is minimized by the Adam optimizer (Kingma and Ba, 2015) with default recommended settings except for the learning rate which we set to 0.001 during the first 5 epochs and then exponentially decrease with a rate of 1/1.15. We use mini-batches of 32 documents. The early stopping conducts to about 18 and 12 training epochs for the *DocumentLevelSplitting* and *IssuerLevelSplitting* configurations, respectively.

Results We reveal the extraction performance of the RNN and feedforward classifiers on the Esker-28k dataset in Table 3.1 and Table 3.2. As detailed in the section 3.3, we report the precisions, recalls and F1 scores averaged over all the test folds for both targeted fields as well as their micro averaged metrics. In Table 3.1, we show the performance for the *DocumentLevelSplitting* configuration, i.e. when the classifiers face document layouts that have been seen at training time. On the contrary, in Table 3.2, the classifiers are tasked to extract information from unknown layouts (*IssuerLevelSplitting* configuration). In both tables, the bold font highlights the best performing model between the RNN and the feedforward baseline.

	F1 score		Precision		Recall	
Field	Baseline	RNN	Baseline	RNN	Baseline	RNN
ID number	0.853	**0.906**	0.863	**0.907**	0.844	**0.905**
Quantity	0.926	**0.964**	0.902	**0.955**	0.952	**0.974**
Micro avg.	0.889	**0.934**	0.882	**0.930**	0.896	**0.938**

Table 3.1: **Performance of word classifiers when facing *known* document layouts.** On the test folds of the Esker-28k dataset, we compare the predictions of a RNN classifier with a feedforward network baseline for the *DocumentLevelSplitting* setting.

For both dataset splitting ways and both fields, the RNN classifier substantially surpasses the baseline feedforward network in terms of precision, recall and thus F1 score. The gap between methods is even greater when facing unknown document layouts (+7.5% of micro F1 score in favor of the RNN compared to +5.1% for known layouts). Given the differences between the two classifiers, the increased performance of the RNN proves that the context knowledge modeled in its hidden states is more effective than the local context knowledge introduced in the feature vectors of the baseline.

	F1 score		Precision		Recall	
Field	Baseline	RNN	Baseline	RNN	Baseline	RNN
ID number	0.685	**0.752**	0.689	**0.769**	0.687	**0.738**
Quantity	0.848	**0.894**	0.842	**0.902**	0.859	**0.888**
Micro avg.	0.764	**0.821**	0.763	**0.834**	0.769	**0.810**

Table 3.2: **Performance of word classifiers when facing *unknown* document layouts.** On the test folds of the Esker-28k dataset, we compare the predictions of a RNN classifier with a feedforward network baseline for the *IssuerLevelSplitting* setting.

Unsurprisingly, the extraction performance is lower for the *IssuerLevelSplitting* configuration (Table 3.2) than for *DocumentLevelSplitting* (Table 3.1) with respective micro F1 scores of 0.821 and 0.934 for the RNN model. However, the difference of scores between known and unknown layouts is small compared to the performances gaps seen for methods relying on class-specific knowledge. For example, Esser et al. (2014) notice a drop of 0.66 of F1 score when switching from known to unknown layouts in its test set. Our gap is similar to the 0.051 difference of micro F1 score from the recurrent network of Palm et al. (2017b) when extracting non-recurring fields. These combined results show that, whatever be the targeted fields, the deep token classifiers are able to learn extraction logic that is not specific to the document layouts seen at training time but that generalizes reasonably well to new layouts.

Finally, we notice that there is a significant difference of performance between the two fields, especially for unknown layouts: the RNN model has its ID number F1 score which is 0.142 lower than for the quantity. One reasonable explanation is the higher level of noise in the word labels for the ID number, even if we have access to the position of the field instances. Indeed, on the purchase orders, there are commonly two product references for a single item, one from the issuing company and one from the recipient side. The labels being generated by Esker's solution users from many distinct companies, one ID number instance or another or both may be marked as ground truth by a particular user. This choice depends on the further integration of the extracted data in their Enterprise Resource Planning (ERP) system. Moreover, as illustrated in Figure 1.3a, the ID number might not have a dedicated physical column, thus often appearing within the description column without clear delimitation of its instances. This behaviour makes the correct extraction of ID Number challenging without additional business context such as databases of product references.

3.4.2 Ablation studies on the token representations

On the Esker-28k dataset, we also conduct experiments to assess the importance of the main components of the word representations. In Table 3.3, we first gauge to what extent

both the textual and layout modalities are useful for extracting information from Visually Rich Documents (VRDs) like purchase orders. In Table 3.4, we more specifically study the textual modality by comparing the impact on performance of the word embeddings, the character-level representations and a combination of both. In all these experiments, we only evaluate the word RNN classifier for the *IssuerLevelSplitting* configuration, i.e. the most challenging setting. For the sake of readability, we only report the micro averaged metrics and omit the individual ID Number and quantity results.

First, we estimate the importance of the textual and layout modalities for IE tasks by successively discarding each modality from the word representations and comparing the resulting performance to the full model (Table 3.3). Specifically, we alternate between dropping the 64 dimensional word level embeddings and the spatial coordinates of their bounding boxes. In both situations, the case features as well as the length of the words are conserved in the feature vectors in order to provide basic information about the words.

Word features	Precision	Recall	F1 score
Word Embedding + Layout	**0.834**	**0.810**	**0.821**
Word Embedding only	0.826	0.780	0.802
Layout only	0.833	0.740	0.784

Table 3.3: **Ablation of the textual and layout modalities from the word representations.** We report the micro averaged metrics obtained by the word RNN classifier on the test folds of the Esker-28k dataset (*IssuerLevelSplitting* configuration).

In line with the works proposing multi-modal models (detailed in the section 2.3.2.1), we note that both the textual and layout modalities are crucial for efficiently extracting information from real-world purchases orders. The biggest drop of performance is reached when we remove the text embeddings, with a loss of 0.037 of micro F1 score compared to a loss of 0.019 when discarding the word coordinates.

During the inspection of the RNN predictions for a representative portion of the test documents, we noticed that information types whose textual content were similar to our targeted fields were regularly confused with them. For example, the product due dates were sometimes predicted as ID numbers since they often share the same word categories (e.g. *ContainsDigitAndDash* or *ContainsDigitAndSlash*) and thus the same word embeddings. The word categories appearing to be too broad, we hypothesised that finer-grained and learned representations of the text content would be beneficial. As detailed in the section 3.2.1.1, we chose to construct such representations by resorting to a BLSTM network iterating over the sequences of characters. In Table 3.4, we judge the benefit of character-level text representations of words – that were so far not included in the evaluated models —. To that end, we compare the IE performance of the RNN classifier using either only 64 dimensional word embeddings, only 64 dimensional character-level representations or a concatenation of both levels, each of size 32. In all cases, we compute the character-level

representations with a single BLSTM layer that relies on randomly initialized character embeddings of size 16. We consider all the characters that are observed on the training set for constituting the vocabulary.

Textual representation of words	Precision	Recall	F1 score
Word level only	0.834	0.810	0.821
Character level only	**0.860**	0.835	0.847
Word + Character level	0.859	**0.842**	**0.850**

Table 3.4: **Comparison of word and character level representations of the words.** We report the micro averaged metrics obtained by the word RNN classifier on the test folds of the Esker-28k dataset (*IssuerLevelSplitting* configuration).

We remark that the character-level representations significantly improve the extraction performance of the RNN with a micro F1 score boost of 0.026 compared to the word embedding baseline. When combining the word and character levels, we reach a F1 score of 0.850 which constitutes the best result among the three settings. Based on these findings, alongside the word embeddings, we include character-level representations in the feature vectors for the rest of the models of this chapter.

3.4.3 Deriving the token labels without the position of fields

In these experiments, we are interested in assessing the evolution of the extraction performance when the position of the field instances is missing and thus cannot be leveraged to create high-quality token level annotations for the classifiers. To that end, we compare the results of the word RNN based model when its training labels are derived *with* and *without* the position of the information within the documents. We conduct experiments on both the private Esker-47k and public SROIE datasets.

Experiment settings We first review the setting values adopted for the Esker-47k dataset. To derive the word and character level textual representations of words, we follow the hyper-parameters retained for the Esker-28k dataset (cf. sections 3.4.1 and 3.4.2). On this dataset, this results in character and word vocabularies of respectively 5,592 and 25,677 elements. We fix the number of BLSTM cells in each encoder layer to 256. For all BLSTM layers, each direction has $n/2$ LSTM cells and their output are concatenated to form n-dimensional vectors. With this parametrization, the whole classifier contains 1,515,733 trainable parameters. The loss is minimized with the Adam optimizer. The learning rate is fixed to 0.001 the first 2 epochs and then exponentially decreases by a factor of 0.8. The batch size is equal to 8.

Unlike the Esker-28k dataset on which the extraction models are evaluated at the document word level, the evaluation of Esker-47k is performed at an higher-level, i.e. by comparing the predicted and ground truth extraction schemas. Therefore, we apply

post-processing to the word RNN classifier to formulate the predictions at the extraction schema level. We follow the business heuristics evoked in the section 3.2.2. For pairing the ID number and quantity instances to form the ordered products, we choose the quantity as the anchor field since this is the field that had the highest word classifier performance in the previous experiments (Table 3.1 and 3.2). Since the metric used to evaluate the IE system is not computed at the document word level, predicting the class of a word by directly choosing the highest classifier confidence (equation 3.2) may be suboptimal. Rather, we apply a threshold strategy for predicting the word classes. For a given document word, if the predicted probability for a class is over its threshold, we attribute the corresponding class. If none of the class confidence is higher that its threshold, we return the `Other` class. The class specific thresholds are determined based on the final performance on the validation set. For Esker-47k, all the classes have been attributed 0.5 threshold values for the two differently annotated versions of the dataset.

Due to the limited size of the validation set (only 26 receipts), we reuse the hyperparameter values of Esker-47k for the experiments on the SROIE dataset. The exceptions are the word vocabulary which is not restricted in size since it contains only 1,160 entries as well as the decision thresholds for the total amount field — for both its `B` and `I` classes — which are fixed to 0.05 when the training instances have been randomly matched to the document candidates.

Results We first show in Table 3.5 the results of the RNN based model on the Esker-47k test set for both annotated versions of the training dataset. We report the post-processing gains as defined in the equation 3.6 for both the ID number and quantity fields and their micro average. We also provide the proportion of the test documents that were perfectly processed by the IE system.

Method for deriving the word training labels	ID number	Quantity	Micro avg.	% Docs Perfect
Using the position of fields	**0.689**	**0.805**	**0.747**	**59.1**
Without the position of fields	0.612	0.708	0.660	44.4

Table 3.5: **Impact of the knowledge of information position on the Esker-47k performance.** For both training label sets, we report the post-processing gains of the RNN based model when extracting the ordered products from the test documents. *% Perfect* column indicates the percentage of documents perfectly processed by the IE model.

We note that for both labeling methods the post-processing gains are positive, meaning that it is more efficient to correct the errors of models than manually perform the extraction from scratch. When using the position of the fields to label the training set, the RNN based system reaches a micro average gain of 0.747 on the test set. This corresponds to a reduction of 75 % of the number of human actions to correctly extract the

information from a Esker-47k document. Moreover, for roughly 6 out of 10 documents, the users do not have any correction to do since the IE system perfectly extracts their ordered products. Such automation rates are really appreciable, especially when knowing how tedious the IE task is if manually performed.

By comparing the two rows of the Table 3.5, we remark that there is a significant decline of extraction performance when the token labels are derived without knowing the position of the fields within the training documents. The difference is more noticeable for the quantity than for the ID number, with a drop of the post-processing gain of respectively 9.7 % and 7.7 %. This divergence is not surprising: as discussed in the section 3.3.2, the two targeted fields are not equally affected by the issue of multiple document occurrences of the field instances. Globally, labeling the training documents without the position of the information leads to a reduction of 15% of documents that are perfectly processed.

Secondly, we report in Table 3.6 the results of the RNN based model on the SROIE test set for both annotated versions of the training dataset. We report the F1 scores for the micro average over the 4 targeted fields as well as for the total amount which is the only field whose the training labels differ between the two labeling methods.

Method for deriving the word training labels	Total amount	All 4 fields (Micro avg.)
Using the position of fields	**0.903**	**0.852**
Without the position of fields	0.886	0.837

Table 3.6: **Impact of the knowledge of information position on the SROIE performance.** For both training label sets, we report the F1 scores of the RNN based model when extracting the total amount and all the targeted fields from the test receipts.

We first note that we reach a micro average F1 score of 0.852 with the best quality of labels. Our extraction method was proposed around the same time as the ICDAR 2019 competition introducing the SROIE dataset took place. If we had participated in this competition, we would have been ranked 5$^{\text{th}}$ out of the 18 received submissions (Huang et al., 2019). The first submission achieved a F1 score of 0.905 while the 10$^{\text{th}}$ submission obtained a F1 score of 0.756. A fine comparison between the proposed methods is still delicate since some of the submissions have corrected the ground truth OCR results of the test set which contain a non negligible amount of errors (see section 3.3.3). However, these results on a public IE dataset validate that our deep token classifier based approach is relevant for extracting information from business documents.

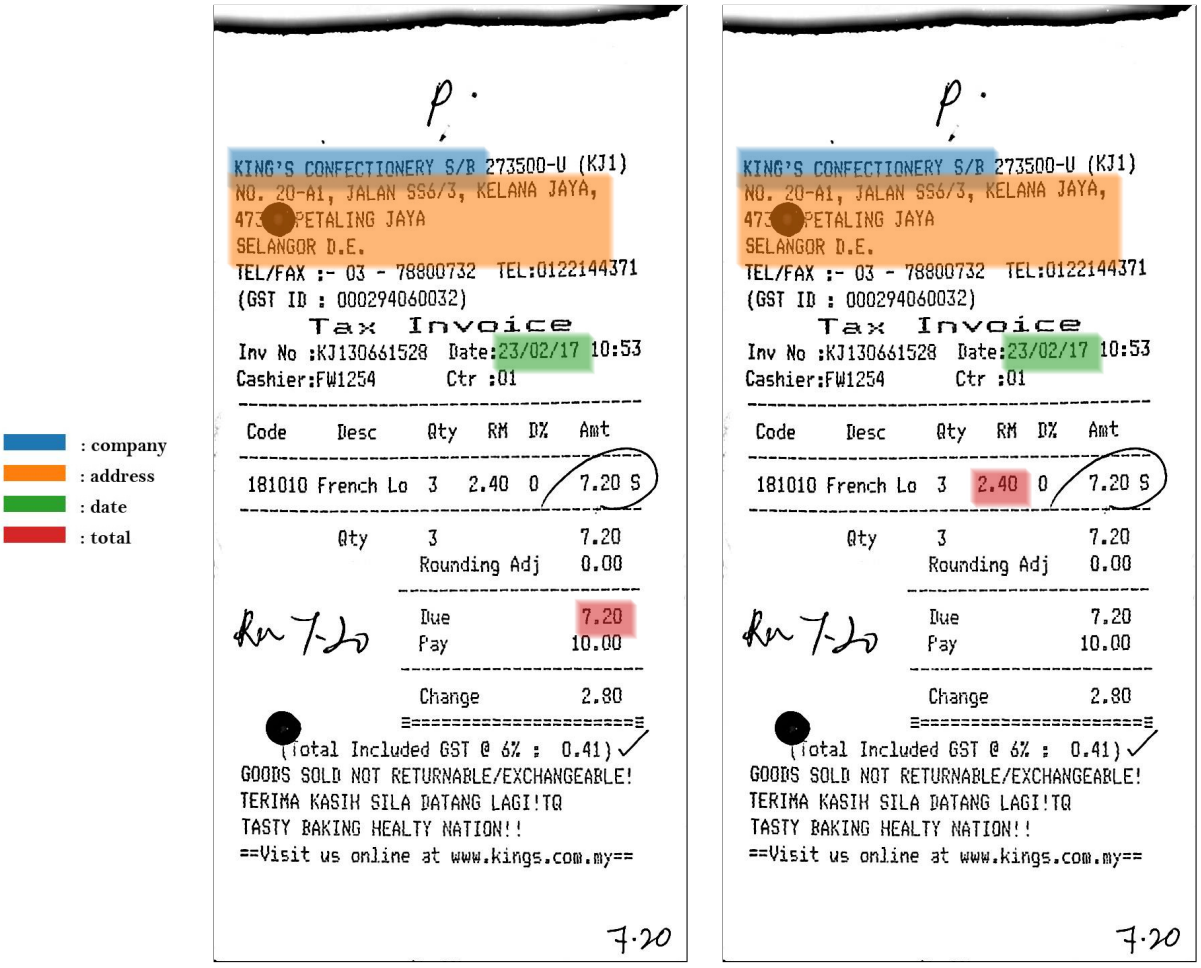
Unlike for Esker-47k, we then remark that the difference of performance between the two labeling methods is not significant on the SROIE dataset. Indeed, the F1 score of the total amount is reduced by 1.7 points when not resorting to the position of field instances to derive token labels while the variation for all four fields is of minus 1.5 points. We

see two main reasons for not observing a larger gap of extraction performance for the total amount between the two ways to construct its token labels. First, as detailed in the section 3.3.3, we in fact do not have the position of the total amount instances in the ground truth but rather estimate their position using strong *a priori* knowledge about the position of this field, i.e. we look for the bottom-most document candidate that is aligned with a keyphrase which contains the `total` word. While simple and powerful, this labeling heuristic is nevertheless not perfect based on our observations of some training receipts with their constructed labels. For example, the total instances are sometimes aligned with other keyphrases such as `Net Amount` or `Due`. These labeling inconsistencies may impact the final extraction performance in an unknown extent. To remedy this, although time consuming, we could manually label the training set for the total amount. The second reason is the structure of the extraction schema, which is simple for the SROIE dataset compared to Esker-47k. Indeed, for all targeted fields of SROIE, we know how many values are expected to be present in the filled schemas — only one instance —. This hypothesis seriously helps to post process the token classifier predictions even in the case of a classifier which is less reliable since it has learned from noisier labels. On the contrary, the extraction schema of Esker-47k contains two fields whose the number of values for a specific document is not known *a priori*. In this case, the fragmentation of the token classifier confidences when exposed to noisy labels is therefore harder to recover from in the post-processing stage.

In the Figure 3.8, we visualize a sample from the SROIE test set where the predictions of the IE system differ between the two labeling methods. Unsurprisingly, we note that the predictions are identical for the company name, address and the date, i.e. the three fields whose the training labels do not vary across the two configurations. On the contrary, there is a discrepancy for the total amount. Its expected value (7.20) was correctly retrieved when the training labels were derived with the help of the position of total amounts (Figure 3.8a) — the correct occurrence within the receipt is aligned with the `Due` keyword —. However, the IE system learning from labels that were produced without the help of information position predicts the value 2.40 which corresponds to a product unit price. This confusion can be explained by the fact that for receipts with a single invoiced product, their unit price may share the value of the total amount. Therefore, during the learning phase, some product unit prices have been labeled as carrying the total amount information. Since the deep neural networks tend to capture the noise in the training data (Arpit et al., 2017), this also affects the inference phase as illustrated in this example.

3.5 Conclusion

In this chapter, we adopted the conventional approach for extracting information from business documents which is based on deep token classifiers (Palm et al., 2017b; Katti et al., 2018; Lohani et al., 2018). Compared to these related works, we performed additional experiments with such models to better understand their behaviour in complementary IE settings. To that end, we created two in-house datasets of real-world documents by



(a) Labeled *with* the position (b) Labeled *without* the position

Figure 3.8: **Predictions of the word RNN classifier based model for a SROIE receipt.** We compare the predictions of the IE system when the training labels for the total amount have been derived with and without the position of its field instances.

collecting and filtering purchase orders from the Esker's commercial document automation solutions. On these two datasets, we studied the performance of token classifiers when extracting complex structures of information, namely tabular entities. Corroborating the results for single-value fields (Palm et al., 2017b), these models were able to extract structured information from document layouts that were not seen at training time. On the most challenging dataset, by resorting to the predictions of a RNN based model, we observed a reduction of 75% of the human effort needed to correctly extract information from unknown document issuers. We showed that both the textual — especially the fine-grained character-level representations of words — and layout modalities were primordial for achieving such results. We finally validated the efficiency of the RNN token classifier based approach by evaluating its performance on a publicly available IE dataset (Huang et al., 2019).

Considering that we reached a satisfying performance level, the RNN based model has been deployed on the Esker infrastructure to allow its customers to reap the benefits of automated information extraction. Originally proposed for extracting products from purchase orders, the approach was rapidly extended to process more fields and other document types like invoices. Currently, this approach is used to extract information from millions of documents per year for the worldwide Esker's customers.

Nevertheless, the token classifiers cannot learn to extract information directly from the ground truth provided by an IE task. Indeed, we have at our disposal filled extraction schemas while such IE models expect supervision at the document token level. The token labels can still be derived from the extraction schemas. But, in the general case — notably when the position of the field instances within the training documents is missing —, the labeling process is not trivial and prone to errors. In the last experiments of this chapter, we demonstrated that such noise introduced in the token labels can lower the extraction performance, especially when processing real-world documents. Therefore, in the next chapter, we plan to explore deep learning methods that bypass token level supervision and are able to learn directly from the filled extraction schemas, i.e. in an end-to-end manner.

Chapter 4

Sequence-to-sequence models for end-to-end extraction

Chapter abstract

The predominant approaches for extracting information from business documents resort to classifiers predicting the information type of each token of the documents. However, the token labels used at training time are not naturally produced by the extraction task and thus may be expensive to obtain or likely to contain errors that degrade the extraction performance.

In this chapter, we aim at learning to extract information directly from the raw extraction schemas rather than from a token level supervision. We introduce a new end-to-end method that, unlike prior IE work, is able to retrieve any arbitrarily structured information such as tabular entities. To achieve this, we adapt an attention-based sequence-to-sequence model called the Pointer-Generator Network (PGN) in order to alternately copy the document words carrying relevant information and generate the XML tags structuring the output. Experiments on both the public SROIE and real-world Esker-47k datasets show that the PGN can outperform a token classifier based system whose training labels are derived only from the extraction schemas and is rivaling with the same system when provided with perfect yet potentially costly labels. Hence, our works confirm that the end-to-end methods are an effective alternative to the traditional IE approaches learning from token labels.

The work in this chapter has led to the publication of a workshop paper:

- Clément Sage, Alex Aussem, Véronique Eglin, Haytham Elghazel, and Jérémy Espinas. End-to-end extraction of structured information from business documents with pointer-generator networks. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 43–52, Online, November 2020. Association for Computational Linguistics.

Contents

4.1 Introduction

In the general case, a supervision at the document token level is not provided for a given IE task, thus preventing from directly training a token classifier to extract information. For sure, the token labels can be automatically guessed from the available ground truth, i.e. the filled extraction schemas. But, as shown by the experiments of the last chapter, this process produces lower quality labels that may significantly reduce the extraction performance.

In this chapter, we investigate models that are able to directly learning IE from the naturally-produced extraction ground truth. By comparison with the token classifier based approaches, such models perform *end-to-end* information extraction (Palm et al., 2019). Besides their potential to avoid decline of performance, end-to-end methods also have two more advantages over the token classifiers. First, they eliminate the need to implement post-processing operations to predict the extraction schemas (Chua and Duffy, 2021). Secondly, end-to-end methods do not rely on a text serializer for making predictions whereas aggregating the IOB2 tags generated by a token classifier imposes to correctly organize the tokens of the document into a unidimensional sequence (Hwang et al., 2020; Hong et al., 2021). Yet, the serialization of business documents is notoriously known to be difficult and even ambiguous since such documents often exhibit a multi-column layout and image distortions due to the scanning process.

To the best of our knowledge, Palm et al. (2017a, 2019) were the firsts to propose deep end-to-end methods for extracting information from documents. They followed the *sequence-to-sequence* (seq2seq) paradigm (Sutskever et al., 2014) which is now ubiquitous within the NLP community. Specifically, they leveraged encoder-decoder architectures to map the input sequence of the document tokens which is fed to the encoder to the output sequence of extracted information generated by the decoder. However, they introduced methods that were only able to extract independent fields, which is severely restricting for

SUNBURST

SALES ORDER: 165813

9/3/2018

SUNBURST, INC.
3565 Sunny Blvd.
San Diego, CA 92115
P 619-870-9876
F 619-870-9865

TO:
IDES US INC.
1230 Lincoln Avenue
New York, NY 10019
USA

SHIP TO:
SUNBURST, INC.
3565 Sunny Blvd.
San Diego, CA 92115
USA

SALES REP.	JOB	SHIP DATE	SHIP VIA	PAYMENT TERMS
Jeanne S.	165813	10/3/2018	FED EX	Due on receipt

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
2	R-1141 PAQ Monitor, 20", Color	300.00	600.00
2	R-1002 Maxitec R 3133 Personal computer	1392.40	2784.80
2	R-5002 Processor Pentium	530.00	1060.00
1	M-11 Flatscreen MS 1785P	1491.22	1491.22
			5936.02

(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 4.1: **End-to-end IE task using the XML format for the output.** Illustrating the Esker-47k task, we aim to retrieve the ordered products which are contained in the main table of the purchase order. Two fields are recognized for each product: their ID number and their quantity.

industrial applications. In our work, we investigate end-to-end approaches that can also process arbitrarily structured information, including but not limited to tabular entities.

For convenience reasons, we do not train our end-to-end IE methods to generate the extraction schemas in the JSON format as in the Figure 1.3b. Indeed, this format would impose to produce many syntactic tokens such as {, : or } (Hwang et al., 2021). Rather, we choose the XML syntax for structuring the output extraction schemas, each XML tag corresponding to a target information type. In Figure 4.1, we illustrate such an end-to-end IE task on a fictive purchase order from the Esker-47k dataset.

To efficiently output such XML representations of the extracted information, we adapt the Pointer-Generator Network (PGN) from See et al. (2017) which is a sequence-to-sequence model originally proposed for text summarization (section 4.3). In the section 4.4, we compare the extraction performance of the PGN to the word classifier approaches studied in the last chapter. On both the SROIE and Esker-47k datasets, we show that the PGN outperforms the word classifier when its training labels were not derived

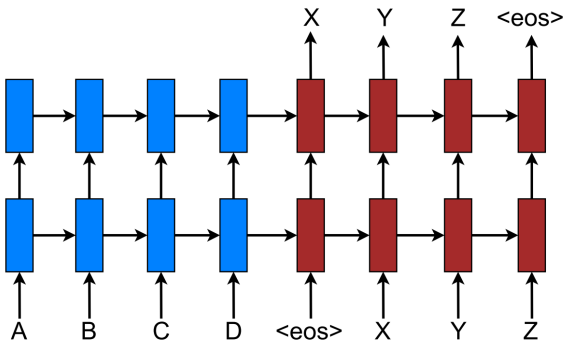


Figure 4.2: **A vanilla sequence-to-sequence model.** This neural architecture allows to transform a source sequence `A B C D` into a target sequence `X Y Z`. Here, unidirectional recurrent connections are utilized for both the encoder and decoder, respectively in blue and brown. `<eos>` denotes the end of both the source and target sequences. Image reproduced from Luong et al. (2015).

with the help of the field position. Moreover, the end-to-end method is demonstrated to perform competitively when high-quality word level supervision is available. We start the chapter by reviewing in the next section the prior work in sequence-to-sequence models.

4.2 Related works

We first discuss the seq2seq framework from a broad perspective and then consider its application to information extraction tasks.

4.2.1 Background in sequence-to-sequence models

4.2.1.1 Vanilla models

Neural sequence-to-sequence models were first introduced to tackle machine translation tasks, e.g. translating an English sentence to its French counterpart (Kalchbrenner and Blunsom, 2013; Cho et al., 2014b; Sutskever et al., 2014). These works proposed neural architectures that map an input sequence of words in the source language to an output sequence in the target language without knowing beforehand its length and its alignment with the input sequence. Such model property is particularly appreciable for performing translation since there are often no one-to-one and monotonic correspondences between the words of the source and target sequences. For example, the English phrase `The quickest brown fox` is likely to be translated in French as `Le renard brun le plus rapide`, a phrase which contains two more words with a different ordering of the adjectives. Sutskever et al. (2014) described an end-to-end model constituted of an encoder which reads the input sequence to produce a fixed-length vector — large enough to encode the whole sequence — and then a decoder to extract the output sequence from that vector, one timestep at a time. We illustrate this neural architecture in Figure 4.2.

Formally, the goal of a sequence-to-sequence model is to estimate the conditional probability $p(\boldsymbol{y}_1, ..., \boldsymbol{y}_{T'} \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_T)$ where $\boldsymbol{x} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}_{T \in \mathbb{N}}$ is the input sequence of length T and $\boldsymbol{y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_{T'}\}_{T' \in \mathbb{N}}$ is its corresponding output sequence of length T' which may differ from T. The encoder-decoder computes this conditional probability by first producing the fixed-size representation $\boldsymbol{v} \in \mathbb{R}^{d_m}$ of the input sequence $\boldsymbol{x}$ and then calculating the probability of $\boldsymbol{y}$ with a language model which is initialized by the vector $\boldsymbol{v}$:

$$p(\boldsymbol{y}_1, ..., \boldsymbol{y}_{T'} \mid \boldsymbol{x}_1, ..., \boldsymbol{x}_T) = \prod_{t=1}^{T'} p(\boldsymbol{y}_t \mid \boldsymbol{v}, \boldsymbol{y}_1, ..., \boldsymbol{y}_{t-1}) \tag{4.1}$$

In this equation, each $p(\boldsymbol{y}_t \mid \boldsymbol{v}, \boldsymbol{y}_1, ..., \boldsymbol{y}_{t-1})$ distribution is represented with a softmax over all the words in the vocabulary of the target language. The vocabulary includes a special end-of-sequence token — usually denoted `<EOS>` — which enables the model to define a distribution over sequences of all possible lengths. The decoder is autoregressive in the sense that it emits one token at a time conditioned on all the source sentence and the previously generated tokens. On each step t, the decoder input $\boldsymbol{x}_t^d$ is the word embedding of the previous word. While training, this is the previous ground truth word whereas at inference time, it is the previous word emitted by the decoder.

Both the encoder and decoder networks are simultaneously trained by minimizing the cross-entropy loss of the correct translations $\{\boldsymbol{y}\}$ given the source sentences $\{\boldsymbol{x}\}$ averaged over all the time steps:

$$-\frac{1}{|S|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in S} \frac{1}{T'} \sum_{t=1}^{T'} \log p(\boldsymbol{y}_t \mid \boldsymbol{v}, \boldsymbol{y}_1, ..., \boldsymbol{y}_{t-1}) \tag{4.2}$$

where S is a training set of parallel sentences. Once the training is complete, the model produces a translation $\hat{\boldsymbol{y}}$ for a new source sentence $\boldsymbol{x}$ by finding the most likely translation:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) \tag{4.3}$$

Searching for the optimal translation over the full output space is often intractable since its complexity is exponential with the output sequence length (Vijayakumar et al., 2016). Approximate search algorithms are thus commonly employed to speed up the decoding process while still reaching a high conditional probability solution. *Beam search* is one of the most prominent inference algorithms to decode output sequences from sequence-to-sequence models. It explores the search space in a greedy left-to-right fashion retaining, at each time step, only the top-B candidates according to the conditional probability. The hyper-parameter B, called the beam width, bounds the memory required to perform the search. With an infinite beam width, we perform exact search of the optimal translation.

A natural parametrization of the encoder and decoder is using Recurrent Neural Networks (RNNs), especially its LSTM or GRU variants. Although the encoder of Figure 4.2 is browsing the input sequence only from left-to-right, the encoder may also resort to bidirectional recurrent connections. We denote by $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T\}$ the hidden states produced by the (bidirectional) RNN encoder for the input sequence $\boldsymbol{x}$. With such parametrization,

the vector representing the input sequence is generally the last hidden state of the RNN encoder, i.e. $\boldsymbol{v} = \boldsymbol{h}_T$, and is used to initialize the hidden state of the RNN decoder. However, the encoder-decoder approach is in no way limited to recurrent architectures and can also successfully leverage the convolution (Gehring et al., 2017) and self-attention (Vaswani et al., 2017) mechanisms which we extensively covered in the section 2.2.

While globally more powerful than traditional statistical translation models, basic sequence-to-sequence systems were still showing troubles when dealing with long sentences to translate (Cho et al., 2014a). To remedy this, Sutskever et al. (2014) suggested to reverse the order of the words in all the source sentences before encoding them with a LSTM network. For most language pairs, this simple trick makes the first words of the output sequence closer than their counterpart in the input sequence. Easing the optimization problem, the reversing indeed improved the encoder-decoder's performance markedly.

4.2.1.2 Enhanced with attention mechanisms

Bahdanau et al. (2015) argued that compressing all the necessary information of a source sentence into a single fixed-length vector $\boldsymbol{v}$ is the main reason explaining that the sequence-to-sequence models had issues when translating long sentences. In turn, they proposed to perform, at each decoder time step, a soft-search for positions in the source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vector $\boldsymbol{h}_t^*$ associated with these source positions and all the previously generated target words $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\}$. Such an extension of the vanilla encoder-decoder model is referred to as an *attention mechanism*. An overview of the mechanism is given in Figure 4.3a when using RNN layers for both the encoder and decoder. Formally, Bahdanau et al. (2015) defined the conditional probability of equation 4.1 by:

$$p\left(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}\right) = g\left(\boldsymbol{y}_{t-1}, \boldsymbol{s}_t, \boldsymbol{h}_t^*\right) \tag{4.4}$$

where g is a non-linear, potentially multi-layered, function that outputs the probability of $\boldsymbol{y}_t$ and s_t is the RNN hidden state for time step t, computed by:

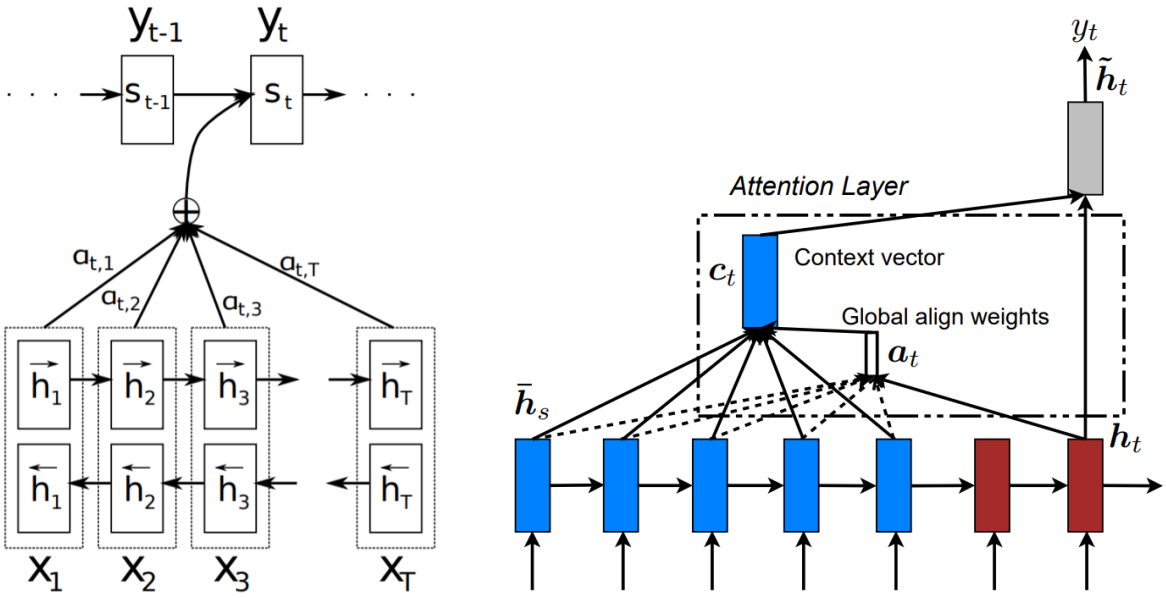
$$\boldsymbol{s}_t = f\left(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{h}_t^*\right) \tag{4.5}$$

where f is the GRU cell function from Cho et al. (2014b) extended for including $\boldsymbol{h}_t^*$. The context vector $\boldsymbol{h}_t^*$ is computed as a weighted sum of the encoder outputs $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T\}$:

$$\boldsymbol{h}_t^* = \sum_{j=1}^{T} \alpha_{tj} \boldsymbol{h}_j \tag{4.6}$$

For each step $t = 1, \ldots, T'$, the weight α_{tj} of each encoder output $\boldsymbol{h}_j$ into $\boldsymbol{h}_t^*$ is calculated with a softmax function:

$$\alpha_{tj} = \frac{\exp\left(e_{tj}\right)}{\sum_{k=1}^{T} \exp\left(e_{tk}\right)} \tag{4.7}$$



(a) Bahdanau's attention

(b) Luong's attention

Figure 4.3: **Two flavours of attention mechanisms.** Both mechanisms allow the RNN decoder to adaptively focus on different parts of the encoded input sequence at each step t. Besides the calculation of the alignment scores between the source and target elements, the main difference is when the attention is computed: before (4.3a) or after (4.3b) updating the hidden state of the decoder. Image reproduced from Bahdanau et al. (2015); Luong et al. (2015)

where e_{tj} is the alignment score between the input sequence at position j and the output sequence at position t. In their work, they implemented the scoring function with:

$$e_{tj} = \boldsymbol{v}_a{}^\top \tanh\left(\boldsymbol{W}_a \left[\boldsymbol{s}_{t-1}, h_j\right] + \boldsymbol{b}_a\right) \tag{4.8}$$

where $\boldsymbol{v}_a, \boldsymbol{b}_a \in \mathbb{R}^{d_a}$ and $\boldsymbol{W}_a \in \mathbb{R}^{d_a \times 2d_m}$ are trainable vector and matrix weights related to the attention mechanism — d_a is the attention size and d_m the dimension of both the encoder and decoder —. The alignment model directly computes a soft differentiable alignment between the source and target sentences. This permits the gradient of the cost function to be backpropagated through, thus allowing the attention module to be learned with the rest of the model.

Bahdanau et al. (2015) showed that enhancing a encoder-decoder with such an attention mechanism conducts to significantly better translation performances when dealing with long sentences, typically for sentences with several dozens of words.

A few later, Luong et al. (2015) simplified and generalized the attention mechanism while still maintaining its efficiency. As shown in Figure 4.3b, they introduced a simpler computation path which now involves the attention mechanism on top of the decoder ($\boldsymbol{s}_t \to \boldsymbol{\alpha}_t \to \boldsymbol{h}_t^* \to \tilde{\boldsymbol{s}}_t$) and not within the decoder anymore ($\boldsymbol{s}_{t-1} \to \boldsymbol{\alpha}_t \to \boldsymbol{h}_t^* \to \boldsymbol{s}_t$),

where $\tilde{\boldsymbol{s}}_t$ is the attention state used to predict the target word $\boldsymbol{y}_t$ and $\boldsymbol{\alpha}_t = [\boldsymbol{\alpha}_{t1}, \ldots, \boldsymbol{\alpha}_{tT}]$ is the attention distribution at the decoder step t. They also proposed another alignment function which performs slightly better than the score function of Bahdanau et al. (2015), replacing the equation 4.8 by:

$$e_{tj} = \boldsymbol{s}_t{}^\top \boldsymbol{W}_a \boldsymbol{h}_j \tag{4.9}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{d_m \times d_m}$ is a matrix of weights.

Due to its effectiveness for machine translation, the encoder-decoder models, particularly when enhanced with an attention mechanism, have then been employed for many other NLP generative tasks such as text summarization, conditional text generation and information extraction (Otter et al., 2020).

4.2.1.3 Copying via attention

For generating output sequences whose elements may correspond to elements of the input sequence, Vinyals et al. (2015) proposed Pointer Networks which use attention as a pointer selecting input elements as the output. These encoder-decoder networks were successfully employed for solving various combinatorial optimization problems such as the Travelling Salesman Problem. A number of follow-up works in text summarization have then mixed these copying abilities with the natural generation mode of the decoder (Gu et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016; See et al., 2017). The pointing part was leveraged to help the neural network to accurately reproduce the prominent information of the text to summarize by efficiently copying the rare or out-of-vocabulary words of the input sequence, e.g. proper nouns. In the following, we describe in more detail the approach of See et al. (2017) which is currently one of the most optimized sequence-to-sequence model combining attention-based copying and classical generation (Xu et al., 2020a).

The end-to-end model proposed by See et al. (2017) is the Pointer-Generator Network, which we abbreviate PGN for readability. As many other sequence-to-sequence models mentioned earlier, the PGN is made of a bidirectional LSTM encoder and a unidirectional LSTM decoder on top of which is placed an attention mechanism using the Bahdanau's alignment function. At each time step $t = 1 \ldots T'$, the generator part of the PGN computes the distribution P_t^{vocab} over the fixed-size output vocabulary V_{fixed} from the concatenation of the decoder hidden state $\boldsymbol{s}_t$ and the context vector $\boldsymbol{h}_t^*$:

$$P_t^{\text{vocab}} = \text{softmax}\left(\boldsymbol{V}'\left(\boldsymbol{V}\left[\boldsymbol{s}_t, \boldsymbol{h}_t^*\right] + \boldsymbol{b}\right) + \boldsymbol{b}'\right) \tag{4.10}$$

where $\boldsymbol{V}$, $\boldsymbol{V}'$ matrices and $\boldsymbol{b}$, $\boldsymbol{b}'$ vectors are learnable parameters of two fully-connected layers. The probability p_t^{gen} of sampling from the generator distribution P_t^{vocab} is computed from the context vector $\boldsymbol{h}_t^*$, the decoder state $\boldsymbol{s}_t$ and the decoder input $\boldsymbol{x}_t^d$:

$$p_t^{\text{gen}} = \sigma\left(\boldsymbol{w}_{h^*}{}^\top \boldsymbol{h}_t^* + \boldsymbol{w}_s{}^\top \boldsymbol{s}_t + \boldsymbol{w}_x{}^\top \boldsymbol{x}_t^d + b_{\text{ptr}}\right) \tag{4.11}$$

where the vectors $\boldsymbol{w}_{h^*}$, $\boldsymbol{w}_s$, $\boldsymbol{w}_x$ and the scalar b_{ptr} are learnable parameters and σ is the sigmoid function ensuring that $p_t^{\text{gen}} \in [0, 1]$. Next, the copying part of the PGN computes

the probability $P_t^{copy}(w)$ of copying a unique word $w \in U_{\boldsymbol{x}}$ from the input sequence $\boldsymbol{x} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}_{T \in \mathbb{N}}$ by summing all its activation weights at the current step t:

$$P_t^{copy}(w) = \sum_{\substack{j=1 \\ \boldsymbol{x}_j = w}}^{T} \alpha_{tj} \tag{4.12}$$

For each input sequence $\boldsymbol{x}$, the final output distribution $p(w \mid \boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ of the PGN is defined over the extended vocabulary V_x which is the union of the fixed-size vocabulary V_{fixed} and the unique words $U_{\boldsymbol{x}}$ of $\boldsymbol{x}$. The distribution is obtained by mixing the generator and pointing distributions through the generation probability p_t^{gen} that operates as a soft switch between the two modes:

$$p(w \mid \boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) = p_t^{\text{gen}} P_t^{\text{vocab}}(w) + (1 - p_t^{\text{gen}})\, P_t^{copy}(w) \tag{4.13}$$

where $P_t^{\text{vocab}}(w) = 0$ for the source words $w \in U_{\boldsymbol{x}}$ which are out-of-vocabulary and $P_t^{copy}(w) = 0$ for the vocabulary words $w \in V_{fixed}$ which do not appear in the input sequence $\boldsymbol{x}$.

4.2.1.4 Coverage mechanism

Repetition is a common problem observed for sequence-to-sequence models, especially when dealing with long output sequences. See et al. (2017) also included a coverage mechanism to mitigate this issue. In this mechanism, a *coverage vector* $\boldsymbol{c}_t \in \mathbb{R}^T$ is maintained to keep track of the source tokens that have already received attention in the previous time steps. This vector is computed by simply summing the attention distributions up to time step t:

$$\boldsymbol{c}_t = \sum_{t'=1}^{t-1} \alpha_{t'} \tag{4.14}$$

with $\boldsymbol{c}_1$ being a zero vector of length T. To ensure that the attention mechanism's current decision is informed by its previous decisions, the coverage vector is used as an extra input to the Bahdanau's attention, changing equation 4.8 to:

$$e_{tj} = \boldsymbol{v}_a{}^{\top} \tanh\left(\boldsymbol{W}_a \left[\boldsymbol{s}_{t-1}, \boldsymbol{h}_j\right] + \boldsymbol{w}_c c_{ti} + \boldsymbol{b}_a\right) \tag{4.15}$$

where $\boldsymbol{w}_c \in \mathbb{R}^{d_a}$ is a learnable vector of the same length as $\boldsymbol{v}_a$. In order to be effective, See et al. (2017) found necessary to define a *coverage loss* that explicitly penalizes the model for repeatedly paying attention to the same source tokens, thus preventing repetitions of the copying mechanism. For the time step t, the coverage loss is equal to:

$$\sum_{j=1}^{T} \min\left(\alpha_{tj}, c_{tj}\right) \tag{4.16}$$

Finally, the coverage loss is reweighted by the hyperparameter $\lambda \in \mathbb{R}$ and added to the primary loss function defined in equation 4.2:

$$-\frac{1}{|S|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in S} \frac{1}{T'} \sum_{t=1}^{T'} \left[\log p(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_1, ..., \boldsymbol{y}_{t-1}) + \lambda \sum_{j=1}^{T} \min\left(\alpha_{tj}, c_{tj}\right)\right] \tag{4.17}$$

4.2.1.5 Structured generation

A number of works have proved that attention-based encoder-decoders can produce well-formed and well-typed sequences in a structured language without supplying an explicit grammar of the language.

Some authors transformed 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 used a forward RNN 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 their formal SQL or Prolog representations (Dong and Lapata, 2016; Rabinovich et al., 2017). Since for this task the text is the modality of both the input and output, Jia and Liang (2016); Zhong et al. (2017) included attention-based copying abilities in their seq2seq model to efficiently generate the rare or out-of-vocabulary words.

4.2.2 Sequence-to-sequence models applied to information extraction

The sequence-to-sequence models were utilized for extracting information from documents in the objective of performing the task in an end-to-end manner. As far as we are aware, Palm et al. (2017a) is the first work which have leveraged such approaches to extract information. They utilized the Pointer Networks (Vinyals et al., 2015) which were presented in the last section in order to retrieve specific fields from short natural language requests in the airline booking, restaurant and movie domains. Their approach is illustrated in the Figure 4.4. They employed a single BLSTM network to encode the sequences of words constituting the requests and as many LSTM decoders as there are target fields. Each decoder is equipped with its own attention mechanism which is tasked to point to the input words to extract the field values. On three public IE datasets, their end-to-end method showed competitive extraction results with strong baselines learning from token-level supervision. Yet, each field specific decoder is independently trained, preventing to learn joint extraction logic and potentially leading to conflicts between the extracted field values at inference time. More importantly, their approach is not able to extract structured information such as entities grouping multiple fields.

In a later work (Palm et al., 2019), the same authors proposed a new sequence-to-sequence approach dubbed Attend, Copy, Parse in order to extract main information from invoices. They mostly focused on designing a multi-modal architecture that efficiently encode such visually rich documents and on providing parsing abilities to the model to not only copy the words of the invoice but also transform them to output normalized field values, e.g. for date and amount fields. However, as in Palm et al. (2017a), their approach is only able to process unstructured information and thus omit the extraction

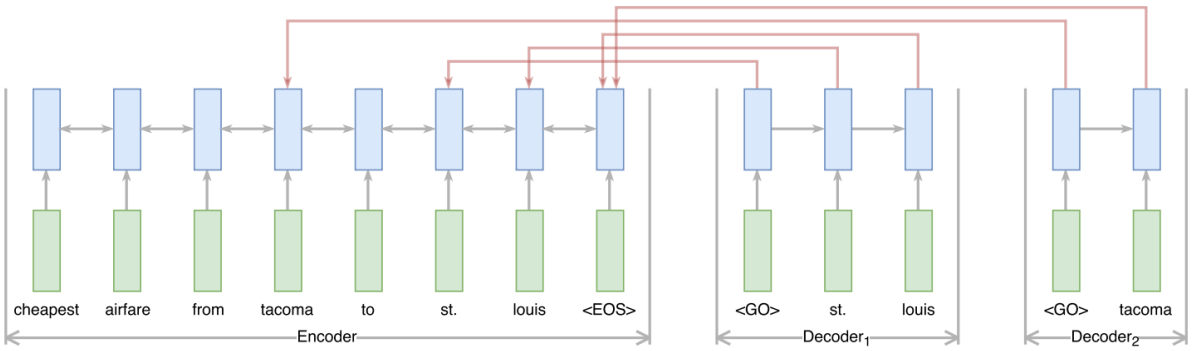


Figure 4.4: **End-to-end extraction through Pointer Networks.** This IE model proposed by Palm et al. (2017a) encodes an input sequence with a BLSTM layer and then uses attention mechanisms on top of its LSTM decoders to copy the input words containing the expected information. The solid red lines denote the strongest attention weights for each decoder step. In this example, the seq2seq model is tasked to retrieve the starting and ending locations from an airline booking request. Image reproduced from Palm et al. (2017a).

of the invoiced products for example.

Since the publication of our work (Sage et al., 2020), two more IE works have been tackling end-to-end extraction of structured information. Like our approach, Hwang et al. (2021) employ an encoder-decoder equipped with an attention-based copying mechanism for extracting information from their in-house name cards and receipts. However, they do not rely on RNN models but rather on the Transformer architecture. They have also chosen another language than XML for structuring the extracted information: their seq2seq model learns to produce sequences that mix the extracted field values and Abstract Syntax Trees (AST) actions. The actions are later interpreted to convert the predictions under the JSON format. As Palm et al. (2017a), their end-to-end model achieves comparable performance with document word classifiers when using the same volume of annotated documents. Moreover, when pre-trained on abundant weakly labeled data that the word classifiers cannot leverage, their sequence-to-sequence model reaches a higher extraction performance than the baseline.

On its side, Chua and Duffy (2021) use a structured prediction approach by resorting to a Context Free Grammar (CFG) which models the structure of information to be extracted. They leverage deep neural encoders to provide the conditional probabilities for each production rule of the CFG, resulting in a tractable end-to-end system for extracting the products from their private invoices.

4.3 Models

In our work, we leverage the sequence-to-sequence framework for directly producing the extraction schemas of the incoming documents following the XML syntax (Figure 4.1). To that end, we serialize both the set of document words and the XML output. For the input sequence, we organize the words from the top-left element to the bottom-right element as for the word classifiers while for the output sequence, we perform a depth-first search of the XML. In the vocabulary of the decoder, we include all the XML tag pairs corresponding to the target fields, i.e. `<Product>`, `</Product>`, `<IDNumber>`, `</IDNumber>`, `<Quantity>` and `</Quantity>` for the Esker-47k task. We also add the token `</Extraction>` into the vocabulary for marking the end of the produced sequences. At each time step, the decoder emits either one of the XML tag or a piece of a field value.

Since the field values contained in the business documents are often specific to a document or a issuer, the extracted information cannot be generated by the decoder from a fixed-size vocabulary of words. At best, the field values would be hard to generate for the rare words. In the worst case where the words are OOV, the extraction would not be possible. Our initial idea was then to use an encoder-decoder which would generate the field values at the *character level* since the character vocabulary is a finite set. Yet, even when enhanced with the attention mechanism from Luong et al. (2015), our preliminary experiments on the Esker-47k dataset showed that the character-level decoder was struggling to extract the actual field values of the documents. This finding was confirmed both by the negative values of the post-processing gain metrics — meaning that it involves more human effort to correct the predictions than manually doing the IE task from scratch — and by the inspection of the attention mechanism which was not able to focus on the document words carrying the expected information. Another drawback of the generation at the character level is the length of the output sequence which reached a maximum of 2,907 characters on the validation set. With auto-regressive decoders, this translates into slow model training and inference (Sun et al., 2019). On the bright side, the seq2seq model was able to produce a well-formed and non empty XML for more than 97 % of the test documents, often predicting a correct number of products and respecting the data type of the fields, i.e. integers for the quantity and alphanumeric sequences for the ID number.

Based on these preliminary results, this is inevitable to resort to the copying abilities of the encoder-decoders in order to efficiently extract information from business documents.

4.3.1 Pointer-generator network

For copying the words of the document carrying relevant information, we adapt the Pointer-Generator Network (PGN) from See et al. (2017) to our extraction needs. In this section, we highlight the keys differences with the original model that we described in details in the section 4.2.1.3. An overview of our approach is given in the Figure 4.5. The responsibilities are split between the generator and the pointer components: the former is only tasked to produce the XML tags structuring the output while the latter is fully in

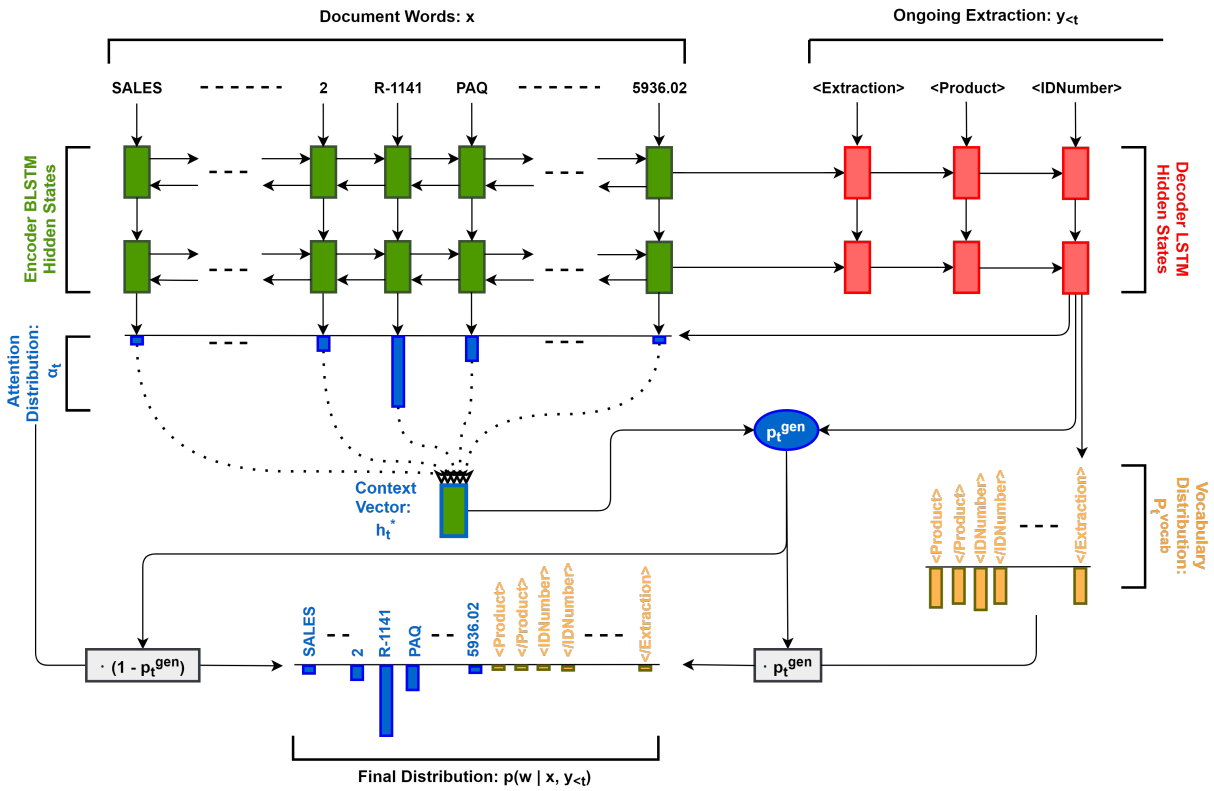


Figure 4.5: **Our Pointer-Generator Network for extracting structured information.** For each decoder step t, a generation probability $p_t^{\text{gen}} \in [0, 1]$ is calculated, which weights the probability of generating XML tags from the vocabulary versus copying words from the document carrying information. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution. For the illustrated step, the model mainly points to the word `R-1141`, i.e. the ID number of the first product from the document in Figure 4.1.

charge of extracting the information from the documents.

4.3.1.1 Word representations

See et al. (2017) only use word level embeddings for describing the elements of the input sequence. In our case, we reemploy the method used in our word classifiers for constructing the representations of the k words of the document (see section 3.2.1.1) since they were shown to be effective in the past experiments. To recap, the word representations $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k\}_{k \in \mathbb{N}}$ are mainly constituted of learned character and word level descriptions of their text content as well as their spatial 2D coordinates for embedding the document layout.

4.3.1.2 Encoder

Like See et al. (2017), the word representations $\{\boldsymbol{r}_j\}_{j=1...k}$ are then fed to a BLSTM encoder to obtain their contextualized representations $\{\boldsymbol{h}_j\}_{j=1...k}$. We use a two-layer network instead of a single layer in the original PGN.

4.3.1.3 Decoder

In accordance with the encoder choices, the decoder is a two-layer forward LSTM, producing a hidden state $\boldsymbol{s}_t$ at each step t. An attention mechanism is added on top of the decoder to compute the attention distribution $\boldsymbol{\alpha}_t$ over the document words and the context vector $\boldsymbol{h}_t^* = \sum_{j=1}^{k} \boldsymbol{\alpha}_{tj} \boldsymbol{h}_j$. While See et al. (2017) use the alignment function of Bahdanau et al. (2015), we choose the form of Luong et al. (2015) since this is computationally less expensive while showing similar performances. As we resort to the coverage mechanism from See et al. (2017) to reduce the repetitions in the output sequences (section 4.2.1.4), we adapt the Luong's alignment function to involve the coverage vector $\boldsymbol{c}_t$. This changes the equation 4.9 to:

$$e_{tj} = \boldsymbol{s}_t{}^\top (\boldsymbol{W}_a \boldsymbol{h}_j + \boldsymbol{w}_c c_{tj}) \tag{4.18}$$

where $\boldsymbol{w}_c$ is a vector of trainable parameters like in the original coverage mechanism (equation 4.15).

To further help the model keeping track of words already copied, we also apply the input-feeding approach from Luong et al. (2015) by concatenating the previous context vector $\boldsymbol{h}_{t-1}^*$ with the decoder input representation $\boldsymbol{x}_t^d$ before applying the first decoder LSTM layer at step t. During training, the decoder input is the previous token of the ground truth sequence, while in inference mode, the previous token emitted by the PGN is used. Its representation $\boldsymbol{x}_t^d$ is either a dense embedding learned from scratch if the token is a XML tag or the textual feature set $\boldsymbol{r}_j^t$ of the corresponding words $\{w_j\}$ if the token is copied from the document.

We also simplify the computation of the distribution P_t^{vocab} since the generator is only in charge of producing the XML tags and thus has a vocabulary of limited size in our setting. We apply a single fully-connected layer instead of two and do not involve the context vector $\boldsymbol{h}_t^*$ in the expression of P_t^{vocab}, changing equation 4.10 to:

$$P_t^{\text{vocab}} = \text{softmax}(\boldsymbol{V}\boldsymbol{s}_t + \boldsymbol{b}) \tag{4.19}$$

where $\boldsymbol{V}$ and $\boldsymbol{b}$ are learnable parameters.

The rest of the PGN is unchanged: we compute the generation probability p_t^{gen} as in equation 4.11 for switching between the generating and pointing modes; the pointing distribution P_t^{copy} is estimated with the equation 4.12 and mixed with the generator's distribution as in equation 4.13 to form the final distribution to sample from. Lastly, the whole model is learned by minimizing the training loss from the equation 4.17 that combines the primary cross entropy loss over the final distribution and the coverage loss.

4.4 Comparison with token classifier approaches

We compare the end-to-end extraction performance of our Pointer-Generator Network (PGN) with the RNN word classifier based method evaluated in the last chapter. We conduct the comparison on both the public SROIE and private Esker-47k datasets which were respectively introduced in the sections 3.3.3 and 3.3.2. In Table 4.1 and 4.2, we recap their main characteristics and give additional statistics related to end-to-end IE.

	# documents
Training	600
Validation	26
Test	347
Words per document	124 ± 34
Pages per document	1.00 ± 0.00
Tokens in output sequence	27.70 ± 3.22
Words per company name instance	4.35 ± 1.26
Words per address instance	11.18 ± 2.86
Words per date instance	1.17 ± 0.56
Words per total instance	1.03 ± 0.17

Table 4.1: **Main statistics of the SROIE dataset** (Huang et al., 2019).

	# documents	# issuers
Training	33,153	11,051
Validation	4,716	1,572
Test	9,846	3,282
Words per document		398 ± 856
Pages per document		1.59 ± 8.69
Products per document		3.81 ± 9.36
Tokens in output sequence		34.35 ± 84.03
Words per ID number instance		1.28 ± 1.13
Words per quantity instance		1.00 ± 0.00

Table 4.2: **Main statistics of the Esker-47k dataset.**

4.4.1 Experiment settings

For the word classifier based approaches, we reuse the parametrization given in the section 3.4.3 for the Esker-47k dataset. For the PGN, we derive the document word representations as the word classifiers. For a fair comparison, we ensure similar numbers of

trainable parameters between models by lowering the size of each BLSTM encoder layer of the PGN to 128 instead of 256. Its two LSTM decoder layers also have a size of 128 and are initialized by the last states of the encoding BLSTM layers. This results in 1,400,908 and 1,515,733 trainable parameters for the PGN and the word classifiers.

Following the recommendations of 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 cross-entropy loss increase. The batch size is 8 if the model fits on the GPU memory, 4 otherwise.

At inference time, we decode with a beam search of width 3. We set the maximum length of the output sequence to the maximum length observed on the validation set for Esker-47k and on both the training and validation sets for SROIE since the latter set only encompasses 26 receipts. A 10 % upper tolerance is also applied. This corresponds to output sequences of at most 49 and 1,885 tokens for respectively the SROIE and Esker-47k datasets.

The experiments are still carried out on a single TITAN X GPU. We keep not training the models on documents with more than 1,800 words, which amounts to 2.8 % of the Esker-47k training set being put aside. Yet, we evaluate the models on all documents of the validation and test sets. The implementation of the PGN is based on the seq2seq subpackage of TensorFlow Addons (Luong et al., 2017).

4.4.2 SROIE results

First, we report in Table 4.3 the results of the PGN and the word classifier based models on the SROIE test set (Huang et al., 2019). As in the last chapter, we include the two versions for deriving the training word labels, the version using the position of field instances and the version only using their textual values (Table 3.6). For SROIE, we remember that these labeling differences only impact the total amount field. We report the F1 scores for this field as well as the micro average over the four targeted fields.

		Total amount	All 4 fields (Micro avg.)
Word classifier	w/ position of fields	0.903	**0.852**
	wo/ position of fields	0.886	0.837
Pointer-**G**enerator **N**etwork		**0.908**	0.839

Table 4.3: **Comparison of the PGN with word classifiers for SROIE receipts.** For all models, we report the F1 scores when extracting the total amount and all the targeted fields from the test receipts.

We note that the PGN reaches similar extraction performance compared to the two word classifiers based approaches, with a micro F1 score of 0.839 against 0.852 and 0.837 for the baselines. Although not significant, the PGN even slightly outperforms the best performing word classifier for the total amount field.

We also have manually inspected the prediction errors and have not noticed any important difference of behaviour between the two families of models. Besides the brittle OCR errors in the provided test set which inevitably disturb the evaluation of the IE task, both the PGN and word classifier approaches tend to miss or wrongly add a word in their predictions for the long field values. This is especially pronounced for the company name and address which are fields containing on average 4.65 and 11.18 words (Table 4.1). Since the evaluation methodology is exact string matching with the ground truth, these slight prediction errors highly affect the F1 score values. Both methods are also inclined to confusions with candidates of the same type, e.g. the paid amount, the total before tax, discount or rounding, the unit and total prices of products are sometimes mistakenly interpreted as the total amount since they are all floats.

Overall, these results validate that end-to-end extraction is a viable alternative to the approaches using costly and error prone word-level supervision for training the extraction models.

4.4.3 Esker-47k results

We then conduct experiments on the Esker-47k task which aims to extract structured tabular information from incoming documents instead of independent fields for the SROIE receipts.

4.4.3.1 Main results

We start the analysis of the Esker-47k results by giving in Table 4.4 the post-processing gains of the PGN and the two word classifiers methods when extracting the products from the purchase orders of the whole test set. The gains of both constituting fields as well as their micro average are displayed alongside the percentage of the test documents for which no insertions, modifications or deletions of predicted field instances are required for achieving perfect extraction.

		ID number	Quantity	Micro avg.	% Docs Perfect
Word classifier	w/ position of fields	**0.689**	**0.805**	**0.747**	59.1
	wo/ position of fields	0.612	0.708	0.660	44.4
Pointer-**G**enerator **N**etwork		0.643	0.777	0.710	**62.2**

Table 4.4: **Comparison of the PGN with word classifiers for Esker-47k documents.** For all models, we report the post-processing gains when extracting the ordered products from the test documents. *% Perfect* column indicates the percentage of documents perfectly processed by the IE model.

We first remark that, like the word classifier based approaches, the PGN have positive post-processing gain values for both the ID number and quantity fields, meaning that

the end-to-end model actually reduces the human efforts for extracting structured information. When compared to the word classifier which learns IE from the same training ground truth, i.e. only from the raw extraction schemas, the PGN greatly outperforms the baseline for all evaluation criteria. The PGN achieves a 7.6 % higher micro average gain (0.710 against 0.660) while it perfectly extracts information from a document 40 % more often than the word classifier using the same level of supervision (62.2 % against 44.4%).

When compared to the word classifier trained with the help of the position of the field instances, all the field level metrics are a little behind for the PGN, its micro post-processing gain being 0.037 lower than the baseline. Yet, the PGN process a whole document without any error slightly more often than the best performing word classifier. These later results shows that end-to-end models are competitive with the traditional IE approaches for extracting structured information at the benefit of demanding fewer efforts for supervising the models.

4.4.3.2 Visual inspection of the pointing mechanism

The quantitative comparison with the word classifiers have shown that the PGN has successfully learned to extract information through its attention-based pointing mechanism. In this section, we aim to get insights into how this pointing module actually proceeds to copy the document words carrying useful information. We particularly investigate the case where there are multiple occurrences of a field value within the document. In Figure 4.6, we visualize the attention distribution $\boldsymbol{\alpha}_t$ while the PGN is extracting the products from a sample purchase order. At the decoder step $t = 6$ that we have chosen, the end-to-end model is recognizing the first product contained in the table of products: it has already retrieved its ID number, i.e. `THX-63972D`, and is about to extract its quantity since the previously emitted token was `<Quantity>`.

We observe that the PGN correctly localizes the expected quantity within the document. Indeed, the PGN massively focuses its attention on the word `1` in the table row of the first product in order to include this word in the output sequence. On the contrary, the model ignores the occurrences of `1` which correspond to street and page numbers as well as other product quantities. This behaviour is noteworthy since the PGN has not been taught to copy the field occurrences which are semantically correct while learning to extract information. Indeed, the PGN does *not* leverage information position knowledge at training time. Rather, the neural network is left in charge of performing the disambiguation between all the occurrences of the training field values. This task is not trivial since the attention weights of all occurrences are summed to form the pointing distribution (equation 4.12), the irrelevant occurrences thus bringing noise in the learning process.

This example thus provides a visual confirmation that the PGN has learned to properly focus its attention in order to copy the relevant document words.

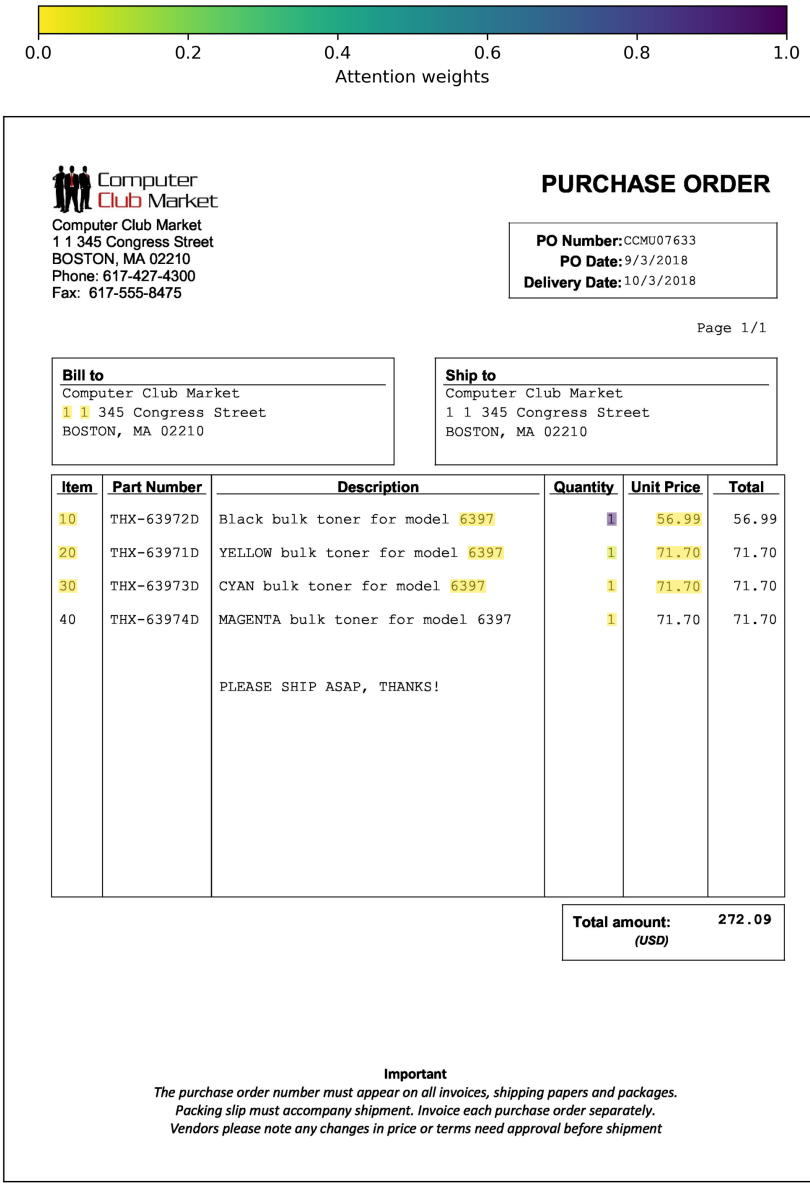


Figure 4.6: **Visualization of the attention-based copying mechanism on a sample purchase order.** We show the top-15 attention weights for the 6^{th} time step of the Pointer-Generator Network (PGN), after having outputted the tokens `<Product>`, `<IDNumber>`, `THX-63972D`, `</IDNumber>` and `<Quantity>`. The model rightly points to the word `1` to extract the quantity of the first product.

4.4.3.3 Troubles for outputting long extraction schemas

Besides the difficulties already encountered by the word classifier based approaches (see section 3.4.1), the PGN also faces a specific issue: it tends to repeat itself by duplicating some field instances and skipping others in its generated sequences. This is especially observed for documents with a large number of products, therefore a large output sequence.

This behaviour is not surprising since prior works using sequence-to-sequence models have already noticed phenomena of repetitions for machine translation and text summarization (Tu et al., 2016; See et al., 2017).

To measure the impact of these repetitions for the information extraction task, we split the Esker-47k test set into three subsets according to the number of products contained in the document: no more than 3, between 4 and 14 and at least 15 products. We recompute the metrics for each test subset and report in Table 4.5 the micro average gains from both the word classifiers and the PGN. For the latter model, we also perform an ablation of the coverage mechanism proposed by See et al. (2017) for mitigating the repetition issue.

		$N \leq 3$	$3 < N < 15$	$N \geq 15$
Documents		7,439	1,977	430
Products		10,404	13,695	13,258
Word classifier	w/ position of fields	0.711	0.768	**0.753**
	wo/ position of fields	0.545	0.685	0.724
Pointer-**G**enerator **N**etwork	w/ coverage	0.734	**0.794**	0.605
	wo/ coverage	**0.802**	0.754	0.438

Table 4.5: **Model performance according to the volume of information to extract.** For both the PGN and the word classifiers, we report the micro average gains over three subsets of the Esker-47k test set. Each subset gathers the documents containing a similar number N of products to retrieve.

We note that the performance is rather stable for the word classifiers whatever the number of products within the document — surprisingly, the micro post-processing gain is even improving when facing a lot of information to extract for the classifier whose training labels were obtained without the help of the information position. On the opposite, the micro average gain of the PGN is higher than both word classifier versions for documents with a reasonable amount of information to recognize ($N < 15$) but its extraction performance greatly declines when facing larger numbers of products. These comparative results indicate that the PGN is more affected by the repetitions than the word level baselines. They also explain the paradoxical fact observed in the global results (Table 4.4) that the PGN perfectly extracts information from documents more often than the word classifiers even if its field level metrics may be lower. Indeed, the purchase orders containing few products constitute the vast majority of the Esker-47k test set and the PGN is better than the word classifiers at extracting information from those documents.

We also remark that the coverage mechanism of the PGN alleviates the repetitions in the output sequences since its removal from the model makes the micro post-processing gain decrease from 0.605 to 0.438 for the longest extraction schemas. Yet, the mechanism still does not permit to reach the levels of performance observed for smaller output se-

quences. The coverage mechanism may be not as effective as in See et al. (2017) since it was originally assessed on expected output sequences of at most 120 tokens while the 430 documents from our third test subset contain on average 31 products, which correspond on average to more than 250 output tokens. Instead of the coverage mechanism from See et al. (2017), we tried to use the temporal attention from Paulus et al. (2018) in order to avoid copying the same words multiple times and thus further improve the performance on long extraction schemas but this was not more successful.

Finally, we verified that the poor test performance of the PGN for the long extraction schemas was not due to the under-representation of such documents in the training set after the discarding, for GPU memory reasons, of the 942 training documents containing more than 1,800 words. Indeed, only 64 out of the 942 removed training documents (6.8 %) were containing more than 15 products while this proportion is only slightly lower in the whole test set (4.4 %).

4.5 Conclusion

In this chapter, we proposed a novel end-to-end method for extracting information from business documents, i.e. that learns directly from the extraction schemas instead of from document token level supervision. Unlike the prior end-to-end approaches which were only able to extract unstructured information (Palm et al., 2017a, 2019), our method can process any arbitrarily structured information since it learns to output the whole extraction schema in a single pass. To achieve this, we adapted the attention-based sequence-to-sequence model developed by See et al. (2017), called the Pointer-Generator Network (PGN), in order to alternately copy the document words carrying relevant information and generate the XML tags structuring the output. We first showed that our PGN is competitive with the word classifiers introduced in the previous chapter for extracting independent fields from receipts of the public SROIE dataset. When tasked to retrieve tabular information from the real-world Esker-47k documents, our PGN greatly outperforms the word classifier using the same level of knowledge for the training ground truth while it rivals with the word classifier resorting to the supplementary knowledge of information position for deriving its training labels. We found that the PGN is lagging behind the best performing word classifier only when there is a lot of information to retrieve due to repetitions in its copying mechanism. Overall, these experiment results confirmed that the end-to-end methods are an effective alternative to the traditional word classifier based approaches, particularly when having only the extraction schemas at our disposal for learning the IE task.

Nevertheless, our end-to-end method is only able to extract fields whose values appear verbatim in the document, either because the fields do not actually require normalization like the ID numbers and quantities in the Esker-47k dataset or their values are deliberately not normalized in the extraction schemas like in the SROIE dataset (Huang et al., 2019). Similarly to Palm et al. (2019), it would be interesting to add parsing abilities into our sequence-to-sequence model in order to simultaneously learn to normalize the extracted

information. Handling normalization would allow to directly learn the IE task from any ground truth extraction schema, e.g. for the Kleister dataset (Graliński et al., 2020), and thus constitutes the last step to fully achieve extraction of structured information in an end-to-end manner.

Chapter 5

Data-efficient extraction with pre-trained language models

Chapter abstract

Like for many text understanding and generation tasks, pre-trained languages models have emerged as a powerful approach for extracting information from business documents. However, their performance has not been properly studied in data-constrained settings which are often encountered in industrial applications. In this chapter, we show that LayoutLM, a pre-trained model recently proposed for encoding 2D documents, reveals a high sample-efficiency when fine-tuned on public and real-world Information Extraction (IE) datasets. Indeed, LayoutLM reaches more than 80% of its full performance with as few as 32 documents for fine-tuning. When compared with a strong baseline learning IE from scratch, the pre-trained model needs between 4 to 30 times fewer annotated documents in the toughest data conditions. Finally, LayoutLM performs better on the real-world dataset when having been beforehand fine-tuned on the full public dataset, thus indicating valuable knowledge transfer abilities. When tackling practical extraction problems, we therefore advocate the use of pre-trained language models for decreasing the annotation efforts.

The work in this chapter has led to the publication of a workshop paper:

- Clément Sage, Thibault Douzon, Alexandre Aussem, Véronique Eglin, Haytham Elghazel, Stefan Duffner, Christophe Garcia, and Jérémy Espinas. Data-efficient information extraction from documents with pre-trained language models. In *Proceedings of the First Workshop on Document Images and Language (DIL)*, pages 455-469, Lausanne, September 2021. Springer International Publishing.

This work has also been presented at a national conference:

- Clément Sage, Thibault Douzon, Alexandre Aussem, Véronique Eglin, Haytham Elghazel, Stefan Duffner, Christophe Garcia, and Jérémy Espinas. Data-efficient information extraction from documents with pre-trained language models. In *Conférence Francophone sur l'Apprentissage Automatique (CAp)*, Online, June 2021.

Contents

5.1 Introduction

In the previous chapter, we investigated sequence-to-sequence models for extracting information from business documents in an end-to-end manner. By learning directly from the extraction schemas and not from document token labels as in Chapter 3, these approaches can effectively reduce the amount of supervision efforts dedicated to a training document and, thus, the human efforts for training an extraction model.

A complementary option to decrease the global supervision efforts would be to minimize the volume of training documents required for reaching the desired performance. One promising way would be to not learn the IE task from scratch as done in our previous approaches and the IE works we have mentioned so far. Indeed, all the trainable parameters of the extraction model, except perhaps the token embeddings (Lohani et al., 2018; Denk and Reisswig, 2019), were learned in a fully supervised task-specific way. The parameters were attributed attributed random values at the beginning of the training stage and their values were then updated by directly minimizing the loss on the target IE dataset. While being successful for most IE tasks, this results in a costly process since a massive amount of weights need to be learned without prior knowledge. On the opposite, starting from weights that have been purposefully pre-trained could allow to learn the IE task from fewer training documents without sacrificing the extraction performance.

For a few years now, a pre-training strategy has gained much traction within the NLP community. The pre-training is performed with language modeling objectives that allow to learn deep representations of tokens from unlabeled text, e.g. BERT (Devlin et al., 2018) or GPT models (Brown et al., 2020). As the language models are self-supervised, one can easily leverage an unbounded amount of text in order to learn powerful and

generic representations of the language. These token representations can then be *fine-tuned* to tackle a wide range of text understanding and generation tasks. Initially geared towards plain text, the language models have then been extended to process the text of visually rich documents (Xu et al., 2020c; Pramanik et al., 2020; Xu et al., 2020b; Hong et al., 2021). These later works have pre-trained language models on large collections of unlabeled documents to be able to understand and reason about the content and structure of business documents. The learned representations have then been adapted for solving several document analysis tasks such as information extraction but also document-level classification and visual question answering. Their evaluation have shown that the pre-trained models considerably outperform the previous state-of-the art models that were trained from scratch, whether they are evaluated on benchmarks with large-scale (Harley et al., 2015) or relatively restrained (Guillaume Jaume, 2019; Huang et al., 2019; Park et al., 2019) annotated sets for training.

However, the study of pre-trained models performance and their comparison to fully supervised models have not been conducted in even more data-constrained settings that are often encountered in practical document analysis applications. In this chapter, we thus aim to quantify to what extent the pre-trained models are sample-efficient when extracting information from documents. Specifically, we compare LayoutLM, a pre-trained language model recently introduced by Xu et al. (2020c) for encoding two-dimensional documents, with two deep models not benefiting from any pre-training.

We here present three main findings that we experimentally validated using the public SROIE benchmark (Huang et al., 2019) as well as a new in-house dataset of purchase orders dubbed PO-51k:

- The pre-trained LayoutLM exhibits remarkable few-shot learning capabilities for IE, reaching more than 80% of its full performance with as few as 32 documents for fine-tuning.

- This model is significantly more data-efficient than a strong non-pretrained baseline in the lowest data regimes, hitting the same levels of extraction performance with around 30 times fewer samples for the real-world dataset.

- Finally, the pre-trained model displays helpful knowledge transfer between IE tasks since learning beforehand to extract information on the full SROIE dataset improves the performance of up to 10 % when fine-tuning the model on the PO-51k dataset.

Our results corroborate the data efficiency of pre-trained language models already observed for classical NLP tasks (Howard and Ruder, 2018; Chen et al., 2020; Brown et al., 2020) and show that using such models dramatically reduces the supervision efforts required for achieving a satisfying IE performance.

5.2 Related works

1D language models As discussed in the section 2.2.2.2, language modeling techniques have lately been massively employed to learn powerful representations of tokens from large corpora of unlabeled text. Originally, the pre-trained tokens representations such as Word2Vec (Mikolov et al., 2013a) or Glove (Pennington et al., 2014) embeddings were then inserted into task-specific architectures for tackling the problems at hand. Nowadays, the language models producing the token representations are directly fine-tuned with minimal alterations to their architecture in order to solve the downstream tasks. Since the introduction of this neural network in Vaswani et al. (2017), the language models are *de facto* based on the Transformer architecture, taking the form of encoder (e.g. BERT (Devlin et al., 2018)), decoder (e.g. the GPT family (Radford et al., 2018; Brown et al., 2020)) or encoder-decoder (e.g. T5 (Raffel et al., 2020)) models.

Naturally, as they were reaching compelling performance for a wide range of classical NLP tasks, the Transformer-based language models have then been leveraged for extracting information from business documents. Following the traditional sequence labeling approach for solving IE, the first works were adding a final fully-connected classification layer on top of a pre-trained encoder and then fine-tuned the parameters of the whole model on the target IE dataset. Nguyen et al. (2019); Zhang et al. (2020b) both showed that the general-purpose language knowledge encoded in BERT transfers well to the highly specialized domains of their business documents for extracting information. To unlock the true potential of language models, Denk and Reisswig (2019) further specified BERT to the target domain by pursuing its pre-training on a large corpus of unlabeled invoices before using its specialized token representations on a much smaller IE dataset.

2D language models Although improving the extraction performance, the traditional language models such as BERT operate on serialized text inputs and thus cannot properly encode the layout and image modalities of business documents. Starting with LayoutLM (Xu et al., 2020c), the language models have been extended to process spatially distributed text from visually rich documents (VRDs), e.g. text blocks and tables.

As illustrated in Figure 5.1, LayoutLM builds upon the BERT or RoBERTa (Liu et al., 2019b) models. Besides the original 1D positional encodings and textual embeddings, the input token representations are enhanced with two-dimensional positional encodings to include the document layout during the pre-training stage. These supplementary positional encodings relate to the absolute 2D coordinates of the token bounding boxes within the page. To take into account the variability of page dimensions, each XY coordinate is normalized to an integer value from 0 to 1,000 to which is associated an embedding vector learned from scratch. Image embeddings are then added to the pre-trained language representations by applying the Faster R-CNN model (Ren et al., 2015) whose Regions of Interest (ROI) correspond to the token bounding boxes.

LayoutLM has been pre-trained over millions of document pages from the IIT-CDIP Test Collection (Lewis et al., 2006) which gathers a wide range of business document types. The pre-training is mainly performed with the Masked Visual Language Modeling

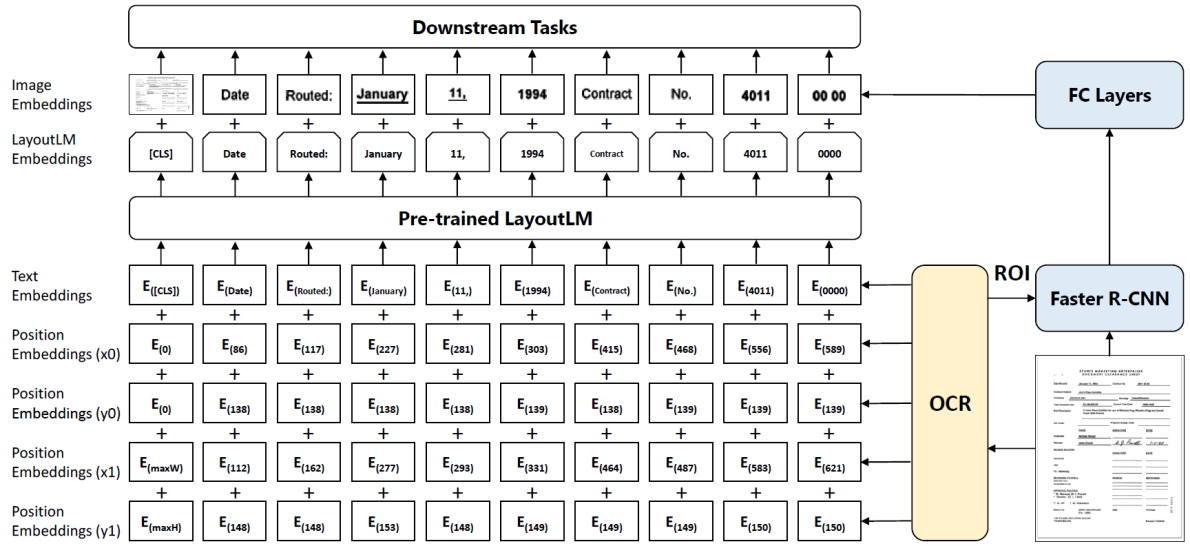


Figure 5.1: **LayoutLM encoding the text of a visually rich document.** Learned two-dimensional positional encodings and image embeddings from Faster R-CNN are integrated into the original BERT architecture to consider the layout and visual modalities of documents. The image and pre-trained LayoutLM embeddings work together for solving downstream document analysis tasks. Image reproduced from Xu et al. (2020c).

(MVLM) loss which naturally expands the MLM objective of BERT that we illustrated in Figure 2.15. This self-supervised task randomly masks the textual embeddings of 15 % the document tokens keeping their 1D and 2D positional encodings and asks LayoutLM to predict what are the masked tokens based on their context.

Once pre-trained, LayoutLM has been fine-tuned on three document understanding tasks by passing the final representations to a fully-connected layer equipped with a softmax activation. For document-level classification such as prediction of the document type in the RVL-CDIP dataset (Harley et al., 2015), the representation of the global $[CLS]$ token is employed. On the SROIE and FUNSD benchmarks (Huang et al., 2019; Guillaume Jaume, 2019), the classification is performed at the token level to predict the information type carried by each token of the document. Xu et al. (2020c) have showed that LayoutLM outperforms the BERT and RoBERTA baselines on these downstream tasks, thus demonstrating the need of combining the text, layout and image modalities for accurately understanding business documents.

The release of LayoutLM has inspired follow-up works aiming to further improve language models dedicated to visually rich documents. While the image modality was introduced only at the fine-tuning stage in LayoutLM, later models include visual descriptors from convolutional layers directly into the token representations used for pre-training (Pramanik et al., 2020; Hong et al., 2021; Xu et al., 2020b). These recent works also focus on adding new pre-training objectives complementing the MVLM loss to more effectively

mix the text, layout and image modalities when learning the document representations. In a second version of LayoutLM, Xu et al. (2020b) introduce the text-image alignment and matching tasks. Topic-modeling and document shuffling tasks (Pramanik et al., 2020), a Sequence Positional Relationship Classification (SPRC) objective (Wei et al., 2020) and a 2D area-masking strategy (Hong et al., 2021) have also been proposed for pre-training language models that process VRDs. Yet, there is lacking a proper comparison in order to conclude which of these self-supervised tasks leads to the best pre-trained token representations and therefore to the highest performance across downstream tasks.

Instead of only relying on the absolute positions supplied in the Transformer input, Xu et al. (2020b); Powalski et al. (2021) both modify the computation of the self-attention scores in order to better model local invariance in the document layout. To that end, they explicitly provide the relative positions of tokens as bias terms added in the self-attention operations.

In the first Transformer-based language models such as BERT, the self-attention is computed as described in the original paper from Vaswani et al. (2017), see equation 2.28. It implies that, in all the self-attention layers, each token attends to all the tokens of the sequence, resulting in a quadratic complexity over the sequence length. With commonly available current hardware and model sizes, this formulation typically limits the input sequence to roughly 512 tokens. It thus prevents Transformers from extracting information without splitting a long document into multiple independently processed sub-sequences. To remedy this, Pramanik et al. (2020) resort to the efficient Longformer's self-attention (Beltagy et al., 2020) that scales linearly with the sequence length in order to process long business documents. They also include page index embeddings into the initial token representations in order to handle multi-page documents.

All the language models dedicated to VRDs that we mentioned so far have been pre-trained and evaluated on English only documents. This is severely restricting since nearly 40% of the digital documents

All these pre-trained language models have largely surpassed fully supervised models that do not benefit from pre-training. They have established state-of-the-art performance on multiple document understanding benchmarks, including common information extraction datasets (Guillaume Jaume, 2019; Huang et al., 2019; Park et al., 2019). Yet, all the experiments have been performed with the full training set of the downstream tasks for fine-tuning, thus not studying the potential of pre-trained models to learn IE from few annotated documents compared to models without such pre-training. Our contribution consists here in showing to what extent the usage of pre-trained language models leads to performance gains on low-resource IE tasks.

5.3 Models

Like most of the works introducing languages models tailored for VRDs, we follow the sequence labeling approach for extracting information from documents. As detailed in Chapter 3, the IE models are thus composed of an encoder delivering contextualized

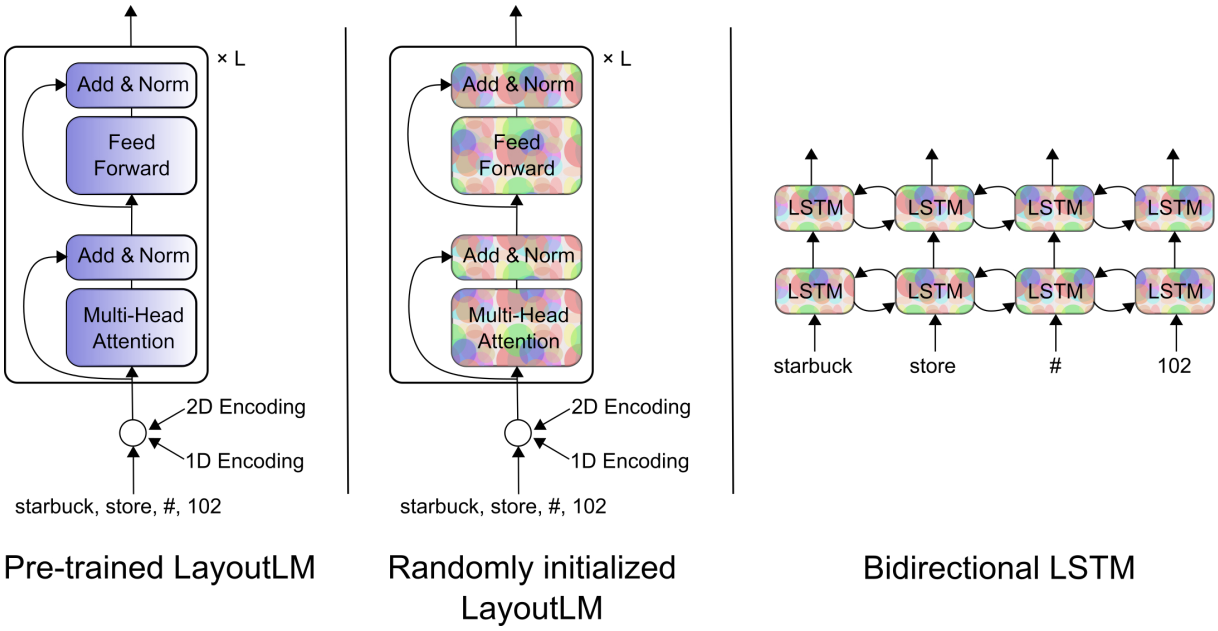


Figure 5.2: **The different architectures used for encoding our documents.** From left to right: Transformer-based LayoutLM (Xu et al., 2020c) with pre-trained weights, LayoutLM with random initialization and a 2-layer bidirectional LSTM also randomly initialized.

representations of the tokens and a linear classifier that decodes this sequence of representations to extract information. We experiment with three models that only differ by the choice of their encoder.

5.3.1 Encoders

As shown in Figure 5.2, we use three different networks for encoding the business documents. We compare a pre-trained encoder with two fully supervised encoders.

5.3.1.1 Pre-trained model

As a pre-trained model, we use LayoutLM (Xu et al., 2020c) because, at the time of our experiments, this was the only 2D language model whose pre-trained parameters have been publicly released[1]. We use its smallest version ($\text{LayoutLM}_{\text{BASE}}$) which is built upon the BERT base-uncased model. It consists of a 12-layer Transformer with a hidden size of 768 and 12 attention heads per layer, resulting in 113 millions weights. Although proposed in their paper for the fine-tuning stage, we do not leverage the image modality since it brings marginal improvements for IE. We thus solely rely on the text and its layout for constructing the final token representations.

[1] https://github.com/microsoft/unilm/tree/master/layoutlm. Since then, the pre-trained weights of LayoutLMv2 (Xu et al., 2020b) have also been published in the same GitHub repository.

5.3.1.2 Fully supervised models

For the fully supervised models, we use 2 encoders that are trained from scratch on the IE tasks. First, we reuse the LayoutLM model but we discard pre-training and randomly initialize all its parameters. However, as confirmed by our early experiments, this encoder version performs poorly in low-resource settings due to its massive amount of parameters to learn from scratch. Secondly, we propose a smaller fully supervised baseline that has shown success in the past chapters. This is a 2-layer bidirectional LSTM network with a 128 hidden size. We reuse the same sub-word tokenizer as LayoutLM and employ only textual embeddings for the tokens. The resulting model contains 8.5 millions parameters, i.e. an order of magnitude fewer parameters than LayoutLM. This makes the BLSTM more suitable to directly learn the IE task in data-constrained regimes.

Following standard practises, the Transformer and embedding layers are respectively initialized with a truncated normal and Gaussian distributions. The BLSTM layers resort to Glorot initialization (Glorot and Bengio, 2010).

5.3.2 Decoder

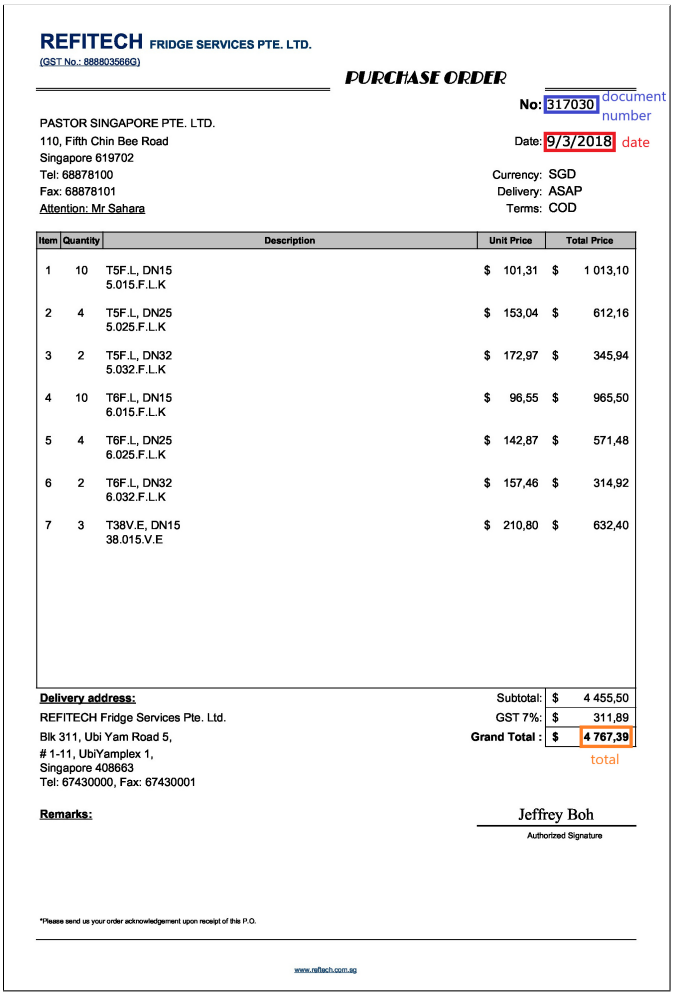
On top of each of these 3 encoders, we add a dense softmax layer to predict the information type carried by each document token. Since the fields to extract can be spread over multiple tokens, the `IOBES` labeling scheme (Ramshaw and Marcus, 1999) is utilized to denote the beginning (`B`), continuation (`I`) and end (`E`) of a field value while `S` classes stand for single token values (see Figure 2.16). This results in 4 output classes per field, with the additional class `O` for tokens not conveying any relevant information. At inference time, we determine the class of a token by getting its highest probability and reduce the resulting list of `IOBES` classes to obtain the field level predictions. If a document has more than 512 tokens, its text is split in multiple sequences that are independently processed by the extraction model.

5.4 Experiments

We train and evaluate the IE models on two datasets: the public **S**canned **R**eceipts **O**CR and **I**nformation **E**xtraction (SROIE) benchmark (Huang et al., 2019) and an in-house dataset of real-world purchase orders dubbed PO-51k.

SROIE The SROIE dataset was covered extensively in the sections 3.3.3 and 4.4 — a sample is given in Figure 3.6 —. In this chapter, as Xu et al. (2020c), we manually fix most of the OCR errors appearing in the official test set[2]. Apart from that, we employ all the pre and post processing operations described in the previous chapters, including the business specific heuristic to derive better token labels for the total amount field.

[2]We provide the corrected dataset at: https://drive.google.com/drive/folders/1T4E7HLOGhLZmEq2cTzXoYRR365qLfANw



REFITECH FRIDGE SERVICES PTE. LTD.
(GST No.: 888803568Q)

PURCHASE ORDER

No: 317030 document number

PASTOR SINGAPORE PTE. LTD.
110, Fifth Chin Bee Road
Singapore 619702
Tel: 68878100
Fax: 68878101
Attention: Mr Sahara

Date: 9/3/2018 date

Currency: SGD
Delivery: ASAP
Terms: COD

Item	Quantity	Description	Unit Price	Total Price
1	10	T5F.L, DN15 5.015.F.L.K	$ 101,31	$ 1 013,10
2	4	T5F.L, DN25 5.025.F.L.K	$ 153,04	$ 612,16
3	2	T5F.L, DN32 5.032.F.L.K	$ 172,97	$ 345,94
4	10	T6F.L, DN15 6.015.F.L.K	$ 96,55	$ 965,50
5	4	T6F.L, DN25 6.025.F.L.K	$ 142,87	$ 571,48
6	2	T6F.L, DN32 6.032.F.L.K	$ 157,46	$ 314,92
7	3	T38V.E, DN15 38.015.V.E	$ 210,80	$ 632,40

Delivery address:
REFITECH Fridge Services Pte. Ltd.
Blk 311, Ubi Yam Road 5,
1-11, UbiYamplex 1,
Singapore 408663
Tel: 67430000, Fax: 67430001

Subtotal:	$ 4 455,50
GST 7%:	$ 311,89
Grand Total :	**$ 4 767,39** total

Remarks:

Jeffrey Boh
Authorized Signature

*Please send us your order acknowledgement upon receipt of this P.O.

www.refitech.com.sg

Figure 5.3: **A fictive purchase order illustrating the PO-51k task.** The goal is to retrieve the document number, the date and the total amount of each incoming purchase order.

PO-51k To prove the efficiency of IE models, we also conduct experiments on documents originating from the Esker's document automation solution. For convenience reasons, the dataset is slightly different than the Esker-28k and Esker-47k datasets that were respectively introduced in the sections 3.3.1 and 3.3.2. Since LayoutLM has been pre-trained on English only documents, we also restrict to English purchase orders. The dataset is composed of 51,000 samples that we split in 40k, 1k and 10k documents for training, validation and test sets. Unlike SROIE, these three subsets contain different document issuers, respectively 6200, 870 and 1700 issuers. This induces that for a large portion of the test set, the layout and content organization of documents have not been seen at training time.

As illustrated in Figure 5.3, we aim to extract 3 different fields among the PO-51k purchase orders: the document number, the date and the total amount. The ground truth for these fields is provided at the word level by the end users of the Esker's automation

software, ensuring high-quality annotations. We employ the same methodology as in SROIE for evaluating the models. This leads to precisions, recalls and F1 scores averaged over the three targeted fields.

Since LayoutLM is not designed for handling multi-page documents, we only consider the first page of documents. Because of this limitation, there may be no value to predict for a target field. In practice, roughly 25% of the documents miss a total amount on the first page while only 10% of the documents are affected for the two other fields.

5.4.1 Experiment settings

We use the following settings in all our experiments. To evaluate data efficiency, we restrict the training set to 8, 16, 32, 64, 128, 256 and 600 randomly selected documents for both datasets. For PO-51k, we additionally study the extraction performance when training with 2k, 8k and 40k samples. We repeat each experiment 5 times, each time with different random seeds and thus different selected training documents. We plot the average μ of the 5 F1 scores as well as the shaded region $[\mu - \sigma, \mu + \sigma]$ for representing the standard deviation σ. We use a log scale over the number of training documents to better visualize the lowest-resource regimes.

As Xu et al. (2020c), we use the Adam optimizer with an initial learning rate of 5e-5, linearly decreasing it to 0 as we reach the maximum number of training steps. For the BLSTM model, we employ a higher initial learning rate of 5e-3 since the former value was not giving a good convergence. For each run, we set the maximum number of training steps to 1k for the pre-trained LayoutLM and 2k for models without pre-training. We proceed to early stopping on the validation set to choose the model checkpoint to evaluate or use for a further training run. We employ a batch size of 8 for all runs in SROIE. For PO-51k, we set the batch size to 16 for all runs, except for 8 and 40k training docs where we fix it to respectively 8 and 32 in order to see at least once each training document. Following the results of language models fine-tuning in low-resource settings (Howard and Ruder, 2018), we update the entire model in all runs.

All training runs are performed on a single 12 Go TITAN XP GPU. We have released the code for reproducing the experiments on the SROIE dataset[3].

5.4.2 Few-shot learning

For both datasets, we first study the performance when the models independently learn the IE task from few annotated samples. After initializing them from scratch or from pre-trained weights, we fine-tune the models for variable numbers of training documents. We report below their results on the whole test set.

5.4.2.1 SROIE

We show F1 scores for the SROIE dataset in the Figure 5.4. We first notice that we get

[3] https://github.com/clemsage/unilm

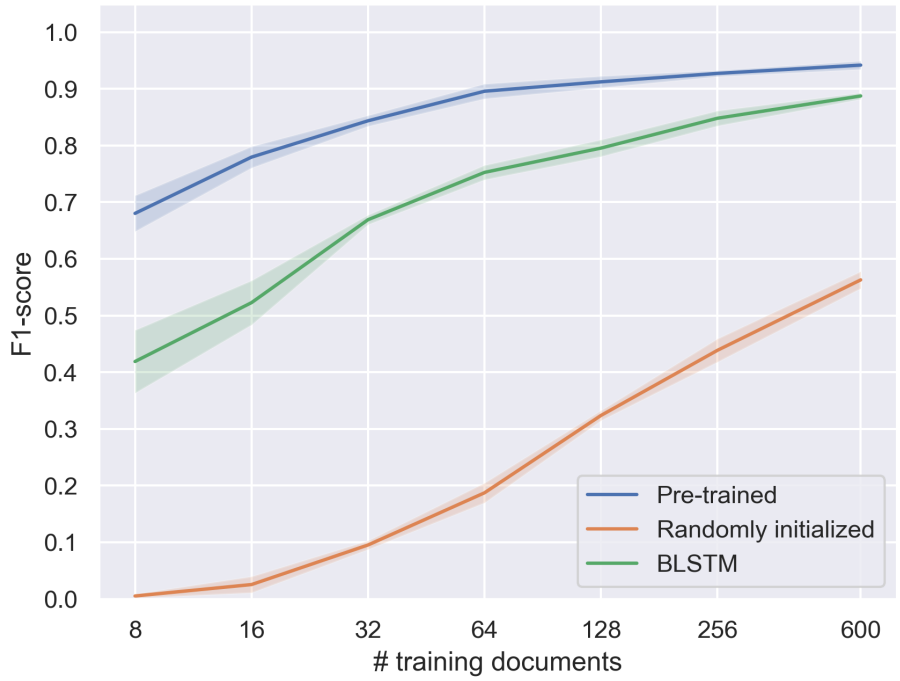


Figure 5.4: **Few-shot extraction performance on the SROIE dataset.** We report the test F1 scores as a function of the number of receipts available for training. We compare the pre-trained LayoutLM (Xu et al., 2020c) against two non pre-trained models: its randomly initialized version and a BLSTM network.

to an average F1 score of 0.9417 when the pre-trained LayoutLM is fine-tuned on 600 receipts. This is in accordance with the 0.9438 F1 score reported in Xu et al. (2020c) when considering the 626 documents of the original training set. The model convergence is really fast, hitting 90% of its full performance with only 32 documents, i.e. a 18 times smaller training set.

Unsurprisingly, we observe that the pre-trained LayoutLM achieves significantly better performance than fully supervised models whatever the number of training documents. Yet, the fewer training documents we make use of, the larger is the difference of F1 score between these two classes of models. For instance, even if the BLSTM network reaches a near similar level of performance with 600 documents (0.8874 against 0.9417), it performs significantly worse than LayoutLM in more data-constrained regimes: the gap of F1 score attains 0.2612 for 8 training receipts. This is even more noticeable for the randomly initialized LayoutLM which completely fails to extract the fields when trained with 8 documents. When offered the full training set, the model does not even outperform its pre-trained counterpart that makes use of only 8 documents.

As expected (Zhang et al., 2021), the performance variance is greater in the lowest data regimes. Yet, the pre-training effectively reduces the variance, making pre-trained models less dependent on the choice of fine-tuning documents.

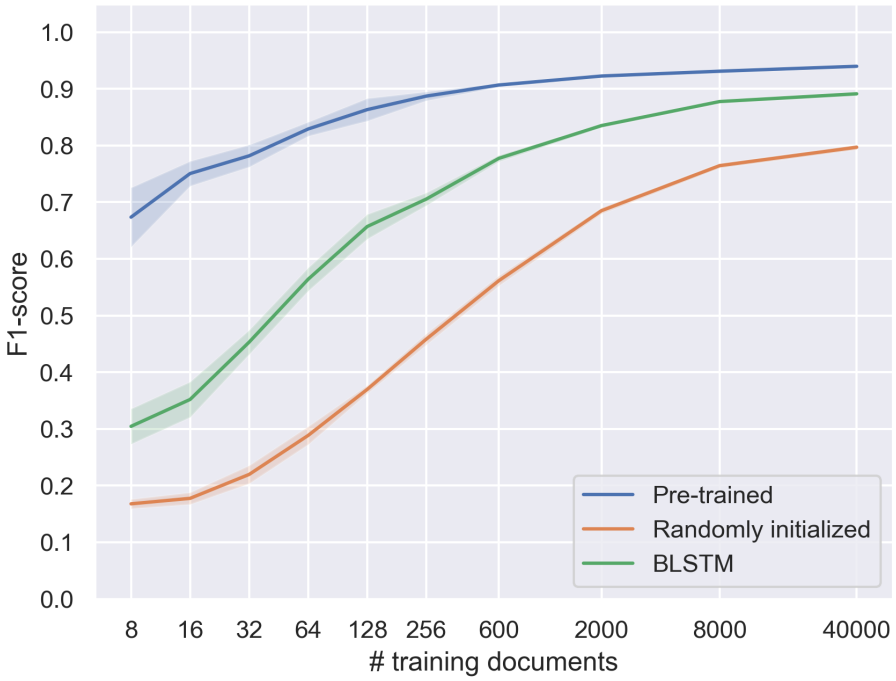


Figure 5.5: **Few-shot extraction performance on the PO-51k dataset.** We report the test F1 scores as a function of the number of documents available for training. We compare the pre-trained LayoutLM (Xu et al., 2020c) against two non pre-trained models: its randomly initialized version and a BLSTM network.

5.4.2.2 PO-51k

We show F1 scores for the PO-51k dataset in the Figure 5.5. We observe similar learning curves for all models, including the pre-trained model that hits 92% of its maximal performance with only 128 samples, i.e. 312 times fewer training documents. In the lowest data regimes, the gap between LayoutLM and the fully supervised baselines is even wider than for SROIE. Indeed, the difference with the BLSTM model is on average of 0.37 F1 score until 32 documents while it was on average of 0.23 points for SROIE. LayoutLM fine-tuned on only 32 documents performs on par with the BLSTM trained with 600 documents, i.e. a order of magnitude more annotations. We also note that this real-world dataset is notoriously more complex than SROIE since a few hundreds documents are not enough to achieve full convergence of the F1 scores. We finally underline the sample inefficiency of LayoutLM trained from scratch with a F1 score at 40k training documents that still lags behind both its pre-trained counterpart and the BLSTM.

On both datasets, we have confirmed that the pre-training stage extensively reduces the amount of annotations needed to reach specific performance for downstream IE tasks.

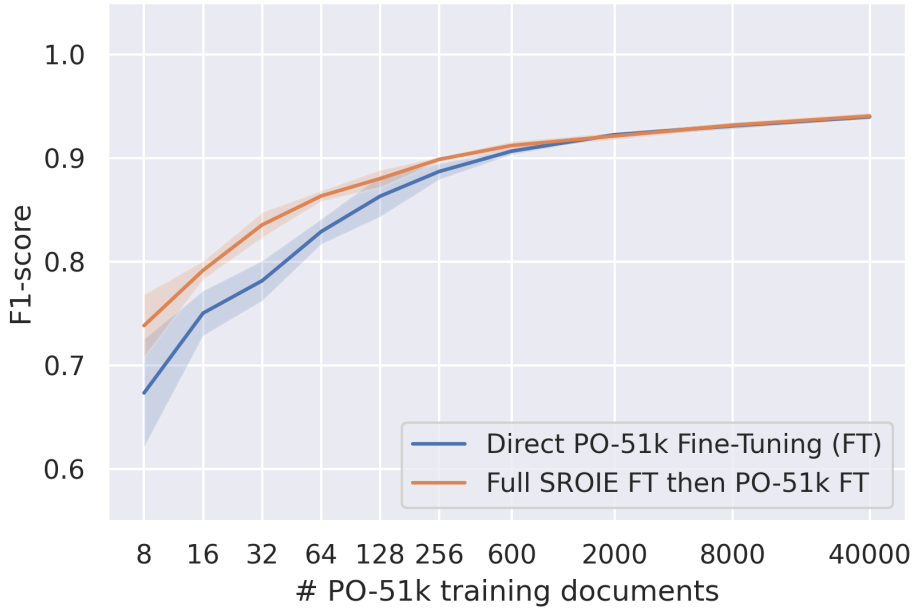


Figure 5.6: **Knowledge transfer between IE tasks with a pre-trained model.** We report the F1 scores of the pre-trained LayoutLM on the PO-51k test set. We either directly fine-tune LayoutLM on the PO-51k documents at our disposal or fine-tune beforehand on the full SROIE training set.

5.4.3 Intermediate learning

In these experiments, we analyze to what extent learning to extract information from given documents decreases the annotation efforts for later performing IE on another document distribution. Specifically, we first fine-tune the pre-trained LayoutLM on the SROIE task using its full training set and then transfer the resulting model on the PO-51k dataset and study its few-shot performance. This simulates an actual use case where a practitioner leverages publicly available data to later tackle IE in more challenging industrial environments.

Since the fields to extract are not identical between the SROIE and PO-51k tasks, we remove the final classifier layer on top of LayoutLM after the fine-tuning on SROIE. We replace it with a randomly initialized layer that matches the number of fields in PO-51k. Even if this imposes to learn again the decoder parameters from scratch between the two IE tasks, there are only a few thousands compared to the million weights of the encoder. We therefore hope that LayoutLM can still transfer some knowledge from SROIE to PO-51k tasks.

5.4.3.1 SROIE to PO-51k

We compare the few-shot performance on PO-51k when having firstly fine-tuned on SROIE with the results obtained when directly employing the pre-trained LayoutLM weights. We show the results of these intermediate learning experiences in Figure 5.6.

We note that the fine-tuning on SROIE considerably improves the extraction for few PO-51k examples with a boost of 0.065 (+10%) F1 score for 8 documents. For 600 examples or more, the effect of intermediate learning disappears with a performance indistinguishable from directly fine-tuning on PO-51k. Fine-tuning beforehand on the SROIE dataset also helps to reduce the variance when it is significant: between 8 to 32 PO-51k documents, the mean standard deviation decreases from 0.031 to 0.017 (-45%) when resorting to intermediate learning.

Therefore, if the amount of annotated documents at their disposal is limited, we encourage IE practitioners not to directly fine-tune the pre-trained models on their task but first use publicly available IE datasets to enhance performance.

5.5 Conclusion

In this chapter, we confirmed that the pre-trained language models are highly beneficial for extracting information from few annotated documents. On a public dataset as well as on a more demanding industrial application, such a pre-trained approach consistently outperformed two fully supervised models that learn from scratch the IE task. We finally demonstrated that the pre-training brings additional improvements when transferring knowledge from an IE task to another.

In the future, it would be pertinent to further investigate the potential of language models for transferring the knowledge learned during its self-supervised and supervised pre-training stages. Indeed, under the current sequence labeling paradigm, the decoder still needs to be learned from scratch for each IE task since the fields to extract are usually specific to a task. This presumably hinders the transferability of extraction knowledge between the downstream tasks. We hypothesize that resorting to decoders with reusable weights may help to better leverage the knowledge learned from the pre-training and intermediate IE tasks. We have particularly in mind the question answering format (Gardner et al., 2019) which has already shown success for zero-shot relation extraction (Levy et al., 2017), i.e. extraction of relations that were unknown at training time.

Chapter 6

Conclusion

Contents

6.1 Summary of contributions

In this manuscript, we covered multiple aspects related to the learning of models extracting information from business documents.

Learning from document token labels In Chapter 3, we embraced the sequence labeling approach which is the historical and still predominant approach for training models to extract information from documents. On both a public dataset and in-house collections of real-world documents, we taught deep models to classify each token of the incoming documents according to the information type it carries. We confirmed that capturing long-range dependencies between the tokens through recurrent connections is helpful for extracting recurring tabular fields. We also verified that the deep token classifiers were able to retrieve such complex information structures from document issuers that were unknown at training time. Third, we demonstrated that combining the textual and layout modalities in the token features is primordial to extract information from visually rich documents such as purchase orders and receipts. Particularly, the word level embeddings were proved to be not powerful enough for representing the textual modality of business documents and should be completed with finer-grained representations.

Although reaching high automation rates, the token classifiers cannot directly learn from the filled extraction schemas which are naturally produced by the IE task. In the general case, where the token level supervision used for learning is not available, it can be derived from these extraction schemas. Yet, we showed that this labeling process is prone to errors and that the noise introduced in the token labels may drastically reduce the extraction performance of sequence labeling approaches.

End-to-end learning To get rid of the dependence on document token level supervision, we proposed in Chapter 4 a novel IE method that learns directly from the extraction

schemas. Unlike the prior end-to-end approaches which were only able to extract unstructured information, our method was designed to process any arbitrarily structured information since it learns to output the whole extraction schema in a single pass. To achieve this, we adapted an attention-based sequence-to-sequence model dubbed Pointer Generator Network (PGN) in order to alternately copy the document tokens carrying relevant information and generate the XML tags structuring the output schema. By comparing the extraction performance of the PGN with the token classifiers studied in Chapter 3, we showed that end-to-end methods are competitive with sequence labeling approaches and can greatly outperform them when their token labels must be deduced from the extraction schemas of the training documents. Yet, we highlighted that end-to-end extraction comes with his own challenges: besides being slower than token classifiers, such methods have troubles when there is a lot of information to retrieve within the incoming document.

Learning from fewer samples In Chapter 5, we studied to what extent learning the extraction task with a pre-trained model reduces the needs for annotated documents. We leveraged an existing language model whose parameter values had been pre-trained in an self-supervised manner on a large collection of business documents. When fine-tuned on an IE task through sequence labeling, the language model was proved to require very few training documents for attaining close to maximal extraction performance. We also confirmed that the pre-trained model is significantly more data-efficient than token classifiers that learn the extraction task from scratch. Finally, we demonstrated that the few-shot performance for a given IE task can be further improved by fine-tuning beforehand the language model on another task even if its types of documents and targeted information differ from the initial task.

6.2 Perspectives for future work

In the following, we highlight some possible future research directions related to the work performed during this thesis.

Normalization of extracted information In this manuscript, we only introduced models able to extract information as it appear in the source documents and thus we set aside the normalization of field values. It would be interesting to adjust our models in order to handle normalization of extracted information and evaluate them on standardized IE benchmarks such as the Kleister datasets (Graliński et al., 2020).

For sequence labeling based IE, the normalization should be introduced in the post-processing stage by adapting the text of field values predicted by the token classifier. It could be manually performed by explicitly exploiting business specific knowledge. Yet, some data types like the amounts and dates exhibit a huge variability of formats within the incoming documents that makes the development of parsing heuristics a tedious task. In such cases, the normalization could also be learned by training sequence-to-sequence models operating at the character level to transform each group of labeled document

tokens to its standardized field value. A Seq2Seq model could be employed for processing all the targeted fields or only a certain data type as in Palm et al. (2019).

For end-to-end extraction, the sequence-to-sequence models could be applied to the whole document text as input in order to simultaneously localize the information in the document and normalize the extracted values. As underlined by Townsend et al. (2021), a particular attention must be paid to the tokenization of the document text. Naturally, we could not tokenize the input sequence at the word level like in the Chapter 4 since the document words could not be directly copied into the extraction schemas. On the other hand, tokenizing the document at each character would result in significantly longer input sequences, and thus prohibitive training and inference times and GPU memory usages for long documents. A sensible solution would be to tokenize the input sequence at an intermediate level, i.e. into subwords. Yet, Townsend et al. (2021) have recently showed that the tokenizer inherited from the T5 model (Raffel et al., 2020) produces a sequence of document subwords that may not align with the the tokenization of the extraction schema. Such token misalignment makes the normalization task challenging to be learned in the same time as the extraction and highlights that future work on tokenization is required beyond directly employing the tokenizers from generic language models. Finally, we would like to improve the attention-based pointing mechanism in order to copy a whole span of tokens from the document at a single timestep (Zhou et al., 2018) instead of letting the decoder successively copy each token like for Hwang et al. (2021) and our PGN. We assume that it would make the learning signal thicker for the pointing mechanism and thus would boost the extraction performance of end-to-end models learning from normalized entries.

Further reduction of supervision efforts In Chapter 5, we investigated pre-trained language models for learning to extract information from fewer annotated documents. We have explored a pre-trained encoder — LayoutLM (Xu et al., 2020c) — that we later fine-tuned for IE tasks following the sequence labeling paradigm. However, if token labels are unavailable, one could also initialize a sequence-to-sequence model with pre-trained parameter values in order to more efficiently pursue end-to-end extraction of information. Encoder-decoder language models specialized for business documents do exist but their pre-trained weights have not been publicly released yet (Powalski et al., 2021; Hwang et al., 2021; Townsend et al., 2021). To remedy this, one solution would be to pre-train its own encoder-decoder, by starting from a language model operating on serialized text, e.g. T5 (Raffel et al., 2020) or BART (Lewis et al., 2020), then inserting 2D positional encodings into the encoder input to process spatially distributed text and finally continuing the pre-training on business document collections with one or several of the self-supervised objectives described in section 5.2 for binding the text and layout modalities.

To further reduce the volume of documents annotated for an IE task, it would also be interesting to develop more advanced strategies for leveraging the knowledge learned by the language models during the pre-training stage and through some intermediate supervised tasks. As evoked in section 5.5, we would particularly like to explore the question answering format (Gardner et al., 2019). This format would allow a language

model to learn to learn to extract information before being finally fine-tuned according to our precise extraction objectives. To that end, extraction instructions such as `Extract the document date from:` could be concatenated to the document tokens and act as questions. The method to answer them would depend to the ground truth at our disposal. If token level supervision is available, we could use a pre-trained encoder like LayoutLM and follow the method for adapting BERT to the SQuAD datasets (Devlin et al., 2018). To achieve this, a generic classifier layer would be added on top of the encoder in order to predict the probability of each document token to be the start and the end of the answer spans. If only end-to-end ground truth is available, a text-generation approach involving a pre-trained encoder-decoder could be employed to answer the questions (Raffel et al., 2020). As intermediate tasks, we could resort to schema-based extraction tasks, e.g. not only the SROIE dataset used in section 5.4.3 but also the CORD (Park et al., 2019) and Kleister (Graliński et al., 2020) benchmarks. We could also leverage open-ended extraction tasks such as Visual Question Answering on document images (Mathew et al., 2021) and question-to-answer linking within forms (Guillaume Jaume, 2019).

Extraction from document images In this thesis, we assumed that the text of the documents is immediately accessible, either because the documents are born-digital or we had already applied an OCR engine for the scanned documents. For the latter, the errors introduced by the OCR step might become the dominant source of extraction errors when facing noisy document images. In this case, it might be relevant to discard the pipeline approach where the text recognition and information extraction steps are decoupled. Instead, as shown by recent works (Zhang et al., 2020a; Wang et al., 2021; Klaiman and Lehne, 2021), the information could be extracted directly from the document images with fully learnable encoder-decoder architectures. To further improve the end-to-end extraction, we could also denoise the document images beforehand through unsupervised methods (Gangeh et al., 2021).

Bibliography

Lincoln A. Mullen, Kenneth Benoit, Os Keyes, Dmitry Selivanov, and Jeffrey Arnold. Fast, consistent tokenization of natural language text. *Journal of Open Source Software*, 3 (23):655, 2018. (Cited on page 34.)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. (Cited on pages 26 and 69.)

ABBYY. Capturing semi-structured forms and documents: challenges and available technologies, 2004. https://cdn.ymaws.com/www.bfma.org/resource/resmgr/articles/05_04.pdf. (Cited on page 17.)

Alireza Alaei, Partha Pratim Roy, and Umapada Pal. Logo and seal based administrative document image retrieval: a survey. *Computer Science Review*, 22:47–63, 2016. (Cited on page 15.)

Jay Alammar. The illustrated Transformer, 2018a. URL http://jalammar.github.io/illustrated-transformer. (Cited on page 33.)

Jay Alammar. The illustrated BERT, ELMo, and co. (how NLP cracked transfer learning), 2018b. URL http://jalammar.github.io/illustrated-bert/. (Cited on page 42.)

Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020. (Cited on pages 19, 20, 21 and 24.)

Enrico Appiani, Francesca Cesarini, Anna Maria Colla, Michelangelo Diligenti, Marco Gori, Simone Marinai, and Giovanni Soda. Automatic document classification and indexing in high-volume applications. *International Journal on Document Analysis and Recognition*, 4(2):69–83, 2001. (Cited on page 16.)

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017. (Cited on page 77.)

Enes Aslan, Tugrul Karakaya, Ethem Unver, and Yusuf Sinan Akgul. A part based modeling approach for invoice parsing. In *International Conference on Computer Vision Theory and Applications*, volume 4, pages 390–397. SCITEPRESS, 2016. (Cited on pages 18 and 44.)

Martin Atzmueller, Peter Kluegl, and Frank Puppe. Rule-based information extraction for structured data acquisition using textmarker. In *LWA*, volume 8, pages 1–7, 2008. (Cited on page 14.)

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. (Cited on page 33.)

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473. (Cited on pages 32, 86, 87, 88 and 94.)

Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzębski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*, 2017. (Cited on page 37.)

Evgeniy Bart and Prateek Sarkar. Information extraction by finding repeated structure. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 175–182, 2010. (Cited on page 18.)

Alberto Bartoli, Giorgio Davanzo, Eric Medvet, and Enrico Sorio. Improving features extraction for supervised invoice classification. In *Artificial Intelligence and Applications*. MH Hamza, 2010. (Cited on pages 6 and 16.)

Yolande Belaïd and Abdel Belaïd. Morphological tagging approach in document analysis of invoices. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 469–472. IEEE, 2004. (Cited on page 14.)

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. (Cited on page 108.)

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. (Cited on page 30.)

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003. (Cited on pages 37, 38 and 58.)

Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017. (Cited on pages 23 and 24.)

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012. (Cited on page 26.)

H. S. Bhatt, S. Roy, L. Bhatnagar, C. Lohani, and V. Jain. Digital auditor: A framework for matching duplicate invoices. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 434–441, 2019. doi: 10.1109/ICDAR.2019.00076. (Cited on page 14.)

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. (Cited on pages 20 and 22.)

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. (Cited on page 39.)

Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991. (Cited on page 25.)

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075. (Cited on page 41.)

Falk Brauer, Robert Rieger, Adrian Mocan, and Wojciech M Barczynski. Enabling information extraction by inference of regular expressions from sample entities. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1285–1294, 2011. (Cited on pages 13 and 14.)

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. (Cited on pages 104, 105 and 106.)

Lukáš Bureš, Petr Neduchal, and Luděk Müller. Automatic information extraction from scanned documents. In *International Conference on Speech and Computer*, pages 87–96. Springer, 2020. (Cited on pages 13 and 15.)

Richard Casey, David Ferguson, K Mohiuddin, and Eugene Walach. Intelligent forms processing system. *Machine Vision and Applications*, 5(3):143–155, 1992. (Cited on page 17.)

João Filipe Mendes Castilho. *Intelligent Document Validation Using Computer Vision and Natural Language Processing*. PhD thesis, Universidade de Coimbra, 2020. (Cited on page 49.)

William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175. Citeseer, 1994. (Cited on page 36.)

Francesca Cesarini, Enrico Francesconi, Marco Gori, Simone Marinai, JQ Sheng, and Giovanni Soda. A neural-based architecture for spot-noisy logo recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 1, pages 175–179. IEEE, 1997. (Cited on page 15.)

Francesca Cesarini, Marco Gori, Simone Marinai, and Giovanni Soda. Informys: A flexible invoice-like form-reader system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):730–745, 1998. (Cited on page 17.)

Francesca Cesarini, Enrico Francesconi, Marco Gori, and Giovanni Soda. Analysis and understanding of multi-class invoices. *Document Analysis and Recognition*, 6(2):102–114, 2003. (Cited on page 18.)

Man-Kit Chang, Waiman Cheung, Chun-Hung Cheng, and Jeff HY Yeung. Understanding ERP system adoption from the user's perspective. *International Journal of production economics*, 113(2):928–942, 2008. (Cited on page 2.)

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013. (Cited on page 38.)

Jiaze Chen, Liangcai Gao, and Zhi Tang. Information extraction from resume documents in pdf format. *Electronic Imaging*, 2016(17):1–8, 2016. (Cited on pages 5 and 45.)

Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Wang. How large a vocabulary does text classification need? a variational approach to vocabulary selection. *arXiv preprint arXiv:1902.10339*, 2019. (Cited on page 37.)

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.18. URL https://www.aclweb.org/anthology/2020.acl-main.18. (Cited on page 105.)

Mengli Cheng, Minghui Qiu, Xing Shi, Jun Huang, and Wei Lin. One-shot text field labeling using attention and belief propagation for structure information extraction. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 340–348, 2020. (Cited on page 43.)

Laura Chiticariu, Yunyao Li, and Frederick Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 827–832, 2013. (Cited on pages 8, 11, 15 and 19.)

Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016. (Cited on page 59.)

Han-Cheol Cho, Naoaki Okazaki, Makoto Miwa, and Jun'ichi Tsujii. Named entity recognition with multiple segment representations. *Information Processing & Management*, 49(4):954–965, 2013. (Cited on page 43.)

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014a. (Cited on pages 31 and 86.)

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014b. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179. (Cited on pages 32, 84 and 86.)

Freddy C Chua and Nigel P Duffy. DeepCPCFG: Deep learning and context free grammars for end-to-end information extraction. *arXiv preprint arXiv:2103.05908*, 2021. (Cited on pages 51, 55, 82 and 91.)

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*, 2014. (Cited on page 31.)

F. Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *IJCAI*, 2001. (Cited on page 14.)

Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015. (Cited on page 26.)

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.07289. (Cited on page 29.)

Bob Cohen and Matthew York. Ardent partners' accounts payable metrics that matter in 2020. Technical report, Ardent Partners, 2020. http://ardentpartners.com/2020/ArdentPartners-AP-MTM2020-FINAL.pdf. (Cited on pages 1 and 3.)

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008. (Cited on pages 37 and 39.)

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011. (Cited on page 37.)

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009. (Cited on page 21.)

Matteo Cristani, Andrea Bertolaso, Simone Scannapieco, and Claudio Tomazzoli. Future paradigms of automated processing of business documents. *International Journal of Information Management*, 40:67–75, 2018. (Cited on page 3.)

Vincent Poulain d'Andecy, Emmanuel Hartmann, and Marçal Rusinol. 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, 2018. (Cited on page 18.)

Tuan Anh Nguyen Dang and Dat Nguyen Thanh. End-to-end information extraction by character-level embedding and multi-stage attentional u-net. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2019. (Cited on pages 48 and 49.)

Sumit Das, Aritra Dey, Akash Pal, and Nabamita Roy. Applications of artificial intelligence in machine learning: review and prospect. *International Journal of Computer Applications*, 115(9), 2015. (Cited on page 20.)

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/17e23e50bedc63b4095e3d8204ce063b-Paper.pdf. (Cited on page 26.)

Florian Deckert, Benjamin Seidler, Markus Ebbecke, and Michael Gillmann. Table content understanding in smartfix. In *2011 International Conference on Document Analysis and Recognition*, pages 488–492. IEEE, 2011. (Cited on page 14.)

Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020. (Cited on page 20.)

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. (Cited on page 21.)

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 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, 2017. (Cited on page 90.)

Yuntian Deng, David Rosenberg, and Gideon Mann. Challenges in end-to-end neural scientific table recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 894–901. IEEE, 2019. (Cited on page 90.)

Andreas R Dengel. Making documents work: Challenges for document understanding. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 1026–1035. IEEE, 2003. (Cited on page 6.)

Timo I Denk and Christian Reisswig. Bertgrid: Contextualized embedding for 2d document representation and understanding. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019. (Cited on pages 48, 49, 104 and 106.)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. (Cited on pages 41, 104, 106 and 120.)

Pranjal Dhakal, Manish Munikar, and Bikram Dahal. 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, 2019. (Cited on pages 17 and 18.)

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon, 2004. (Cited on page 12.)

William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. (Cited on page 41.)

Li Dong and Mirella Lapata. 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, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1004. URL `https://www.aclweb.org/anthology/P16-1004`. (Cited on page 90.)

Timothy Dozat. Incorporating nesterov momentum into adam. In *ICLR 2016 workshop*, 2016. (Cited on page 26.)

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011. (Cited on page 26.)

Pradheep Elango. Coreference resolution: A survey. *University of Wisconsin, Madison, WI*, 2005. (Cited on page 12.)

Daniel Esser, Daniel Schuster, Klemens Muthmann, Michael Berger, and Alexander Schill. Automatic indexing of scanned documents: a layout-based approach. In *Document recognition and retrieval XIX*, volume 8297, page 82970H. International Society for Optics and Photonics, 2012. (Cited on pages 6 and 16.)

Daniel Esser, Daniel Schuster, Klemens Muthmann, and Alexander Schill. Few-exemplar information extraction for business documents. In *ICEIS (1)*, pages 293–298, 2014. (Cited on pages 18, 19 and 72.)

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957. (Cited on page 38.)

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013. (Cited on page 66.)

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001. (Cited on page 19.)

Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. (Cited on page 27.)

Robert Gaizauskas and Yorick Wilks. Information extraction: Beyond document retrieval. *Journal of documentation*, 54(1):70–105, 1998. (Cited on page 12.)

Rinon Gal, Shai Ardazi, and Roy Shilkrot. Cardinal graph convolution framework for document information extraction. In *Proceedings of the ACM Symposium on Document Engineering 2020*, pages 1–11, 2020. (Cited on pages 48 and 49.)

Mehrdad J Gangeh, Marcin Plata, Hamid Motahari, and Nigel P Duffy. End-to-end unsupervised document image blind denoising. *arXiv preprint arXiv:2105.09437*, 2021. (Cited on page 120.)

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. Question answering is a format; when is it useful? *arXiv preprint arXiv:1909.11291*, 2019. (Cited on pages 116 and 119.)

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017. (Cited on page 86.)

Felix A Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 189–194. IEEE, 2000. (Cited on page 31.)

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. (Cited on page 110.)

Filip Graliński, Tomasz Stanisławek, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. Kleister: A novel task for information extraction involving long documents with complex layout. *arXiv preprint arXiv:2003.02356*, 2020. (Cited on pages 51, 55, 102, 118 and 120.)

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005. (Cited on page 32.)

Gregory Grefenstette. Tokenization. In *Syntactic Wordclass Tagging*, pages 117–133. Springer, 1999. (Cited on page 34.)

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. (Cited on page 31.)

Ralph Grishman and Beth Sundheim. Message Understanding Conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996. URL https://www.aclweb.org/anthology/C96-1079. (Cited on page 12.)

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, 2016. (Cited on page 88.)

Jean-Philippe Thiran Guillaume Jaume, Hazim Kemal Ekenel. Funsd: A dataset for form understanding in noisy scanned documents. In *Accepted to ICDAR-OST*, 2019. (Cited on pages 105, 107, 108 and 120.)

Hui Han, C Lee Giles, Eren Manavoglu, Hongyuan Zha, Zhenyue Zhang, and Edward A Fox. Automatic document metadata extraction using support vector machines. In *2003 Joint Conference on Digital Libraries, 2003. Proceedings.*, pages 37–48. IEEE, 2003. (Cited on page 44.)

Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE, 2015. (Cited on pages 51, 105 and 107.)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. (Cited on pages 24 and 32.)

Jerry R Hobbs. The generic information extraction system. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*, 1993. (Cited on page 12.)

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. (Cited on page 30.)

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008. (Cited on page 24.)

Martin Holeček, Antonín Hoskovec, Petr Baudiš, and Pavel Klinger. Line-items and table understanding in structured documents. *arXiv preprint arXiv:1904.12577*, 2019. (Cited on page 48.)

Xavier Holt and Andrew Chisholm. Extracting structured data from invoices. In *Proceedings of the Australasian Language Technology Association Workshop 2018*, pages 53–59, 2018. (Cited on page 44.)

Teakgyu Hong, DongHyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. BROS: A pre-trained language model for understanding texts in document, 2021. URL https://openreview.net/forum?id=punMXQEsPr0. (Cited on pages 48, 82, 105, 107 and 108.)

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. (Cited on page 27.)

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031. (Cited on pages 40, 105 and 112.)

Su Su Htay and Khin Thidar Lynn. Extracting product features and opinion words using pattern knowledge in customer reviews. *The Scientific World Journal*, 2013, 2013. (Cited on page 44.)

Jianying Hu, Ramanujan Kashi, and Gordon Wilfong. Comparison and classification of documents based on layout similarity. *Information Retrieval*, 2(2-3):227–243, 2000. (Cited on page 16.)

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE, 2019. (Cited on pages 51, 55, 56, 62, 66, 67, 68, 76, 78, 95, 96, 101, 105, 107, 108 and 110.)

Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Sohee Yang, and Minjoon Seo. Spatial dependency parsing for 2d document understanding. *arXiv preprint arXiv:2005.00642*, 2020. (Cited on page 82.)

Wonseok Hwang, Hyunji Lee, Jinyeong Yim, Geewook Kim, and Minjoon Seo. Cost-effective end-to-end information extraction for semi-structured document images. *arXiv preprint arXiv:2104.08041*, 2021. (Cited on pages 83, 91 and 119.)

iPayables. Why automation matters: A survey study of the modern accounts payable department. Technical report, iPayables, 2016. https://pdf4pro.com/cdn/a-survey-study-of-the-modern-accounts-payable-department-565c5d.pdf. (Cited on page 1.)

Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901. (Cited on page 17.)

Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017. (Cited on page 22.)

Robin Jia and Percy Liang. 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, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1002. URL https://www.aclweb.org/anthology/P16-1002. (Cited on page 90.)

Zhaohui Jiang, Zheng Huang, Yunrui Lian, Jie Guo, and Weidong Qiu. Integrating coordinates with context for information extraction in document images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 363–368. IEEE, 2019. (Cited on pages 47 and 48.)

Anjali Ganesh Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011. (Cited on page 36.)

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1700–1709, 2013. (Cited on page 84.)

Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. Chargrid: Towards understanding 2d documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4459–4469, 2018. (Cited on pages 37, 48, 49, 50, 55, 65 and 77.)

Mohamed Kerroumi, Othmane Sayem, and Aymen Shabou. Visualwordgrid: Information extraction from scanned documents using a multimodal approach. *arXiv preprint arXiv:2010.02358*, 2020. (Cited on page 50.)

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference*

Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980. (Cited on pages 26 and 71.)

Shachar Klaiman and Marius Lehne. Docreader: Bounding-box free training of a document information extraction model, 2021. (Cited on page 120.)

Bertin Klein, Andreas R Dengel, and Andreas Fordan. smartfix: An adaptive system for document analysis and understanding. In *Reading and Learning*, pages 166–186. Springer, 2004. (Cited on pages 14, 15 and 18.)

Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980. (Cited on page 17.)

John R Koza, Forrest H Bennett, David Andre, and Martin A Keane. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In *Artificial Intelligence in Design'96*, pages 151–170. Springer, 1996. (Cited on page 19.)

Felix Krieger, Paul Drews, Burkhardt Funk, and Till Wobbe. Information extraction from invoices: A graph neural network approach for datasets with high layout variety. In *Proceedings of WI 2021*, 2021. (Cited on page 48.)

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://www.aclweb.org/anthology/P18-1007. (Cited on page 36.)

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://www.aclweb.org/anthology/D18-2012. (Cited on page 36.)

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. (Cited on page 61.)

Andrey Kurenkov. A brief history of neural nets and deep learning. *Skynet Today*, 2020. (Cited on page 26.)

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages

282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813. (Cited on page 45.)

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://www.aclweb.org/anthology/D17-1082. (Cited on pages 40 and 41.)

Kevin J Lang. A time-delay neural network architecture for speech recognition. *Technical Report*, 1988. (Cited on page 27.)

Yann Le Cun, Leon Bottou, and Yoshua Bengio. Reading checks with multilayer graph transformer networks. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 151–154. IEEE, 1997. (Cited on page 27.)

Rémi Lebret and Ronan Collobert. Word embeddings through hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-1051. URL https://www.aclweb.org/anthology/E14-1051. (Cited on page 37.)

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. (Cited on page 27.)

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, 2017. (Cited on page 116.)

David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, D Grossman, and Jefferson Heard. Building a test collection for complex document information processing. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 665–666, 2006. (Cited on page 106.)

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703. (Cited on page 119.)

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and H. V. Jagadish. Regular expression learning for information extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 21–30, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D08-1003. (Cited on page 14.)

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luis Marujo, and Tiago Luís. 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, 2015. (Cited on page 59.)

Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize f1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–239. Springer, 2014. (Cited on page 64.)

Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 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, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-2005. URL https://www.aclweb.org/anthology/N19-2005. (Cited on pages 6, 7, 48 and 49.)

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b. (Cited on pages 42 and 106.)

Vasily Loginov, Aleksandr Valiukov, Stanislav Semenov, and Ivan Zagaynov. Document data extraction system based on visual words codebook. In *Document Analysis Systems: 14th IAPR International Workshop, DAS 2020, Wuhan, China, July 26–29, 2020, Proceedings*, page 356. Springer Nature, 2020. (Cited on page 4.)

Devashish Lohani, Abdel Belaïd, and Yolande Belaïd. An invoice reading system using a graph convolutional network. In *Asian Conference on Computer Vision*, pages 144–158. Springer, 2018. (Cited on pages 48, 77 and 104.)

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017. (Cited on page 96.)

Thang Luong, Hieu Pham, and Christopher D. Manning. 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, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/

D15-1166. URL https://www.aclweb.org/anthology/D15-1166. (Cited on pages 84, 87, 92 and 94.)

Steven L Lytinen and Anatole Gershman. Atrans automatic processing of money transfer messages. In *AAAI*, volume 86, pages 1089–1093, 1986. (Cited on pages 11 and 15.)

Bodhisattwa Prasad Majumder, Navneet Potti, Sandeep Tata, James Bradley Wendt, Qi Zhao, and Marc Najork. Representation learning for information extraction from form-like documents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6495–6504, 2020. (Cited on pages 46, 61 and 68.)

Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to information retrieval*. Cambridge university press, 2008. (Cited on page 12.)

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209, 2021. (Cited on page 120.)

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598, 2000. (Cited on page 45.)

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/20c86a628232a67e7bd46f76fba7ce12-Paper.pdf. (Cited on page 40.)

Eric Medvet, Alberto Bartoli, and Giorgio Davanzo. A probabilistic approach to printed document understanding. *International Journal on Document Analysis and Recognition (IJDAR)*, 14(4):335–347, 2011. (Cited on page 18.)

Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, 2016. (Cited on page 88.)

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781. (Cited on pages 39 and 106.)

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b. (Cited on page 39.)

Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969. (Cited on page 27.)

T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN 9780071154673. URL https://books.google.fr/books?id=EoYBngEACAAJ. (Cited on page 19.)

Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088. Citeseer, 2009. (Cited on page 37.)

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer, 2005. (Cited on page 37.)

Hamid Motahari, Nigel Duffy, Paul Bennett, and Tania Bedrax-Weiss. A report on the first workshop on document intelligence (di) at neurips 2019. *ACM SIGKDD Explorations Newsletter*, 22(2):8–11, 2021. (Cited on page 50.)

Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989. (Cited on page 29.)

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007. (Cited on page 12.)

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016. (Cited on page 88.)

Minh-Tien Nguyen, Viet-Anh Phan, Nguyen Hong Son, Miku Hirano, Hajime Hotta, et al. Transfer learning for information extraction with limited data. In *International Conference of the Pacific Association for Computational Linguistics*, pages 469–482. Springer, 2019. (Cited on page 106.)

Christopher Olah. Understanding lstm networks, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/. (Cited on pages 30 and 31.)

Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. (Cited on page 88.)

Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019. (Cited on page 45.)

Rasmus Berg Palm. *End-to-end information extraction from business documents*. PhD thesis, Technical University of Denmark, 2019. (Cited on page 15.)

Rasmus Berg Palm, Dirk Hovy, Florian Laws, and Ole Winther. End-to-end information extraction without token-level supervision. In *Proceedings of the Workshop on Speech-Centric Natural Language Processing*, pages 48–52, 2017a. (Cited on pages 82, 90, 91 and 101.)

Rasmus Berg Palm, Ole Winther, and Florian Laws. Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 406–413. IEEE, 2017b. (Cited on pages 44, 47, 48, 54, 55, 56, 59, 61, 63, 72, 77 and 78.)

Rasmus Berg Palm, Florian Laws, and Ole Winther. Attend, copy, parse end-to-end information extraction from documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 329–336. IEEE, 2019. (Cited on pages 7, 82, 90, 101 and 119.)

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. Cord: A consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019. (Cited on pages 51, 105, 108 and 120.)

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013. (Cited on page 69.)

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf. (Cited on page 26.)

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkAClQgA-. (Cited on page 101.)

Sachin Pawar, Girish K Palshikar, and Pushpak Bhattacharyya. Relation extraction: A survey. *arXiv preprint arXiv:1712.05191*, 2017. (Cited on page 12.)

Fuchun Peng and Andrew McCallum. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4):963–979, 2006. (Cited on pages 44 and 45.)

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. (Cited on pages 39, 58 and 106.)

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1161. URL https://www.aclweb.org/anthology/P17-1161. (Cited on page 40.)

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202. (Cited on page 40.)

Rafał Powalski, Łukasz Borchmann, Dawid Jurkiewicz, Tomasz Dwojak, Michał Pietruszka, and Gabriela Pałka. Going full-tilt boogie on document understanding with text-image-layout transformer. *arXiv preprint arXiv:2102.09550*, 2021. (Cited on pages 108 and 119.)

Subhojeet Pramanik, Shashank Mujumdar, and Hima Patel. Towards a multi-modal, multi-task learning based pre-training framework for document representation learning. *arXiv preprint arXiv:2009.14457*, 2020. (Cited on pages 105, 107 and 108.)

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999. (Cited on page 26.)

Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. Graphie: A graph-based framework for information extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 751–761, 2019. (Cited on pages 5 and 48.)

M Atif Qureshi and Derek Greene. Eve: explainable vector based embedding technique using wikipedia. *Journal of Intelligent Information Systems*, 53(1):137–165, 2019. (Cited on page 37.)

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 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, July 2017. Association for Computational Linguistics. doi: 10.

18653/v1/P17-1105. URL https://www.aclweb.org/anthology/P17-1105. (Cited on page 90.)

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *arxiv*, 2018. (Cited on pages 40, 41 and 106.)

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21: 1–67, 2020. (Cited on pages 106, 119 and 120.)

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264. (Cited on page 41.)

Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999. (Cited on pages 43 and 110.)

Rizlene Raoui-Outach, Cecile Million-Rousseau, Alexandre Benoit, and Patrick Lambert. Deep learning for automatic sale receipt understanding. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 2017. (Cited on page 15.)

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 91–99, 2015. (Cited on page 106.)

AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge, MA, 1987. (Cited on page 29.)

Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton*. Cornell Aeronautical Laboratory, 1957. (Cited on pages 26 and 28.)

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. (Cited on pages 24 and 26.)

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. (Cited on pages 26 and 27.)

Marçal Rusinol, Tayeb Benkhelfallah, and Vincent Poulain dAndecy. Field extraction from administrative documents by incremental structural templates. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1100–1104. IEEE, 2013. (Cited on pages 15, 17 and 18.)

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. ISBN 0136042597. (Cited on page 22.)

S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991. (Cited on page 24.)

Clément Sage, Alex Aussem, Véronique Eglin, Haytham Elghazel, and Jérémy Espinas. End-to-end extraction of structured information from business documents with pointer-generator networks. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 43–52, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.spnlp-1.6. (Cited on page 91.)

Naomi Sager. *Natural Language Information Processing: A Computer Grammmar of English and Its Applications*. Addison-Wesley Longman Publishing Co., Inc., 1981. (Cited on page 11.)

Sunita Sarawagi. Efficient inference on sequence segmentation models. In *Proceedings of the 23rd international conference on Machine learning*, pages 793–800, 2006. (Cited on page 44.)

Sunita Sarawagi. Information extraction. *Foundations and Trends® in Databases*, 1(3): 261–377, 2008. ISSN 1931-7883. doi: 10.1561/1900000003. URL http://dx.doi.org/10.1561/1900000003. (Cited on pages 11, 13, 14, 43, 44 and 45.)

Ritesh Sarkhel and Arnab Nandi. Improving information extraction from visually rich documents using visual span representations. *Proceedings of the VLDB Endowment*, 14 (5):822–834, 2021. (Cited on page 50.)

Roger C Schank and Kenneth Mark Colby. *Computer models of thought and language*. WH Freeman San Francisco, 1973. (Cited on page 11.)

Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 1162–1167. IEEE, 2017. (Cited on page 4.)

Daniel Schuster, Marcel Hanke, Klemens Muthmann, and Daniel Esser. Rule-based versus training-based extraction of index terms from business documents: how to combine the results. In *Document Recognition and Retrieval XX*, volume 8658, page 865813. International Society for Optics and Photonics, 2013a. (Cited on page 44.)

Daniel Schuster, Klemens Muthmann, Daniel Esser, Alexander Schill, Michael Berger, Christoph Weidling, Kamil Aliyev, and Andreas Hofmeier. Intellix–end-user trained information extraction for document archiving. In *2013 12th International Conference on Document Analysis and Recognition*, pages 101–105. IEEE, 2013b. (Cited on page 17.)

Daniel Schuster, Daniel Esser, Klemens Muthmann, and Alexander Schill. Modelspace-cooperative document information extraction in flexible hierarchies. In *ICEIS (1)*, pages 321–329, 2015. (Cited on page 18.)

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE, 2012. (Cited on page 36.)

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. (Cited on pages 31 and 32.)

Abigail See, Peter J Liu, and Christopher D Manning. 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, 2017. (Cited on pages 83, 88, 89, 92, 93, 94, 96, 100 and 101.)

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162. (Cited on page 36.)

Kristie Seymore, Andrew McCallum, Roni Rosenfeld, et al. Learning hidden markov model structure for information extraction. In *AAAI-99 workshop on machine learning for information extraction*, pages 37–42, 1999. (Cited on page 45.)

Younghak Shin and Ilangko Balasingham. Comparison of hand-craft feature based svm and cnn based deep learning framework for automatic polyp classification. In *2017 39th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 3277–3280. IEEE, 2017. (Cited on page 45.)

Chirag Shivalker. How Automating Your Invoice Processing helps You Grow Your Business, 2020. https://datafloq.com/read/how-automating-your-invoice-processing-helps-you-g/7452. (Cited on page 2.)

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/74563ba21a90da13dacf2a73e3ddefa7-Paper.pdf. (Cited on page 92.)

Beth M. Sundheim. Overview of the third Message Understanding Evaluation and Conference. In *Third Message Uunderstanding Conference (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*, 1991. URL https://www.aclweb.org/anthology/M91-1001. (Cited on page 12.)

Beth M Sundheim. Tipster/muc-5 information extraction system evaluation. Technical report, NAVAL COMMAND CONTROL AND OCEAN SURVEILLANCE CENTER RDT AND E DIV SAN DIEGO CA, 1993. (Cited on page 12.)

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3104–3112, 2014. (Cited on pages 82, 84 and 86.)

Gábor Szegedi, Diána Bajdikné Veres, Imre Lendák, and Tomáš Horváth. Context-based information classification on hungarian invoices. In *ITAT*, 2020. (Cited on page 45.)

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJcKhk-Ab. (Cited on page 30.)

Yuan Yan Tang, Ching Y Suen, Chang De Yan, and Mohamed Cheriet. Financial document processing based on staff line and description language. *IEEE transactions on systems, man, and cybernetics*, 25(5):738–754, 1995. (Cited on page 17.)

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher D Manning. Max-margin parsing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, 2004. (Cited on page 45.)

Sandeep Tata, Navneet Potti, James B. Wendt, Lauro Beltrao Costa, Marc Najork, and Beliz Gunel. Glean: Structured extractions from templatic documents. In *Proceedings of the VLDB Endowment*, 2021. (Cited on page 56.)

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020. (Cited on page 34.)

Untapped Editorial Team. State of Technology in the AP Department | Accounts Payable Automation, 2020. https://www.mediusflow.com/en/untapped/articles/process/state-of-technology-ap-department. (Cited on page 2.)

Virginia Teller. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. *Computational Linguistics*, 26(4):638–641, 2000. (Cited on page 37.)

Ken Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968. (Cited on page 13.)

Benjamin Townsend, Eamon Ito-Fisher, Lily Zhang, and Madison May. Doc2dict: Information extraction as text generation. *arXiv preprint arXiv:2105.07510*, 2021. (Cited on page 119.)

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, 2016. (Cited on page 100.)

Jordi Turmo, Alicia Ageno, and Neus Català. Adaptive information extraction. *ACM Computing Surveys (CSUR)*, 38(2):4–es, 2006. (Cited on page 14.)

George Tzanis, Ioannis Katakis, Ioannis Partalas, and Ioannis Vlahavas. Modern applications of machine learning. In *Proceedings of the 1st Annual SEERC Doctoral Student Conference–DSC*, volume 1, pages 1–10, 2006. (Cited on page 20.)

Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112, 2014. (Cited on page 36.)

Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009. (Cited on page 20.)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. (Cited on pages 32, 33, 34, 86, 106 and 108.)

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016. (Cited on page 85.)

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700, 2015. (Cited on pages 88 and 90.)

Jiapeng Wang, Chongyu Liu, Lianwen Jin, Guozhi Tang, Jiaxin Zhang, Shuaitao Zhang, Qianying Wang, Yaqiang Wu, and Mingxiang Cai. Towards robust visual information extraction in real world: New dataset and novel solution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2738–2745, 2021. (Cited on pages 4 and 120.)

Mengxi Wei, Yifan He, and Qiong Zhang. Robust layout-aware IE for visually rich documents with pre-trained language models. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2367–2376. ACM, 2020. doi:

10.1145/3397271.3401442. URL https://doi.org/10.1145/3397271.3401442. (Cited on pages 6 and 108.)

Ralph Weischedel and Elizabeth Boschee. What can be accomplished with the state of the art in information extraction? a personal view. *Computational Linguistics*, 44(4): 651–658, 2018. (Cited on page 12.)

Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988. (Cited on page 29.)

Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Image-to-markup generation via paired adversarial learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 18–34. Springer, 2018. (Cited on page 90.)

Patrick Xia, Shijie Wu, and Benjamin Van Durme. Which *BERT? A survey organizing contextualized encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7516–7533, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.608. URL https://www.aclweb.org/anthology/2020.emnlp-main.608. (Cited on page 42.)

Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362, 2020a. (Cited on page 88.)

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:2012.14740*, 2020b. (Cited on pages 42, 105, 107, 108 and 109.)

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. LayoutLM: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200, 2020c. (Cited on pages 19, 105, 106, 107, 109, 110, 112, 113, 114 and 119.)

Zhi-qin John Xu, Tao Luo, Zheng Ma, and Yaoyu Zhang. Suggested notation for machine learning. https://github.com/Mayuyu/suggested-notation-for-machine-learning, 2020d. (Cited on page xv.)

Vikas Yadav and Steven Bethard. 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, 2018. (Cited on page 47.)

Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. World Scientific, 2009. (Cited on page 22.)

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5315–5324, 2017. (Cited on page 48.)

Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. PICK: processing key information extraction from documents using improved graph learning-convolutional networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4363–4370. IEEE, 2021. (Cited on page 48.)

Peng Zhang, Yunlu Xu, Zhanzhan Cheng, Shiliang Pu, Jing Lu, Liang Qiao, Yi Niu, and Fei Wu. TRIE: End-to-end text reading and information extraction for document understanding. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1413–1422, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450379885. doi: 10.1145/3394171.3413900. URL https://doi.org/10.1145/3394171.3413900. (Cited on pages 4 and 120.)

Ruixue Zhang, Wei Yang, Luyun Lin, Zhengkai Tu, Yuqing Xie, Zihang Fu, Yuhao Xie, Luchen Tan, Kun Xiong, and Jimmy Lin. Rapid adaptation of bert for information extraction on domain-specific business documents. *arXiv preprint arXiv:2002.01861*, 2020b. (Cited on page 106.)

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. Revisiting few-sample BERT fine-tuning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=cO1IH43yUF. (Cited on page 113.)

Xiaohui Zhao, Zhuo Wu, and Xiaoguang Wang. CUTIE: Learning to understand documents with convolutional universal text information extractor. *arXiv preprint arXiv:1903.12363*, 2019. (Cited on pages 48, 49 and 50.)

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017. (Cited on page 90.)

Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: data, model, and evaluation. *arXiv preprint arXiv:1911.10683*, 2019. (Cited on page 90.)

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. Sequential copying networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. (Cited on page 119.)

Guangyu Zhu, Timothy J Bethea, and Vikas Krishna. Extracting relevant named entities for automated expense reimbursement. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1004–1012, 2007. (Cited on pages 44 and 45.)

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. (Cited on page 38.)

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015. (Cited on pages 38, 40 and 41.)