



HAL
open science

Language Models for Document Understanding

Thibault Douzon

► **To cite this version:**

Thibault Douzon. Language Models for Document Understanding. Artificial Intelligence [cs.AI]. INSA LYON, 2023. English. NNT: . tel-04459378

HAL Id: tel-04459378

<https://hal.science/tel-04459378>

Submitted on 15 Feb 2024

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

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



N°d'ordre NNT : 2023ISAL0075

**THÈSE de DOCTORAT DE L'INSA LYON,
membre de l'Université de Lyon**

**École Doctorale n°512
InfoMaths**

**Spécialité de doctorat :
Informatique**

Soutenue publiquement le 24/10/2023, par :
Thibault Douzon

**Language Models for
Document Understanding**

Devant le jury composé de :

Lemaitre Aurélie	Maître de Conférences HDR, Université Rennes 2	Rapporteuse
Paquet Thierry	Professeur des Universités, Université de Rouen Normandie	Rapporteur
Tabbone Salvatore-Antoine	Professeur des Universités, Université de Lorraine	Examineur
Ogier Jean-Marc	Professeur des Universités, La Rochelle Université	Président du jury & Examineur
Garcia Christophe	Professeur des Universités, INSA Lyon	Directeur de thèse
Duffner Stefan	Maître de Conférences HDR, INSA Lyon	Co-directeur de thèse
Espinas Jérémy Bérard Jean-Jacques	Docteur-Ingénieur R&D, Esker Directeur R&D, Esker	Invité & Encadrant Invité

Référence : TH1015_DOUZON Thibault

L'INSA Lyon a mis en place une procédure de contrôle systématique via un outil de détection de similitudes (logiciel Compilatio). Après le dépôt du manuscrit de thèse, celui-ci est analysé par l'outil. Pour tout taux de similarité supérieur à 10%, le manuscrit est vérifié par l'équipe de FEDORA. Il s'agit notamment d'exclure les auto-citations, à condition qu'elles soient correctement référencées avec citation expresse dans le manuscrit.

Par ce document, il est attesté que ce manuscrit, dans la forme communiquée par la personne doctorante à l'INSA Lyon, satisfait aux exigences de l'Établissement concernant le taux maximal de similitude admissible.

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	CHIMIE DE LYON https://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	M. Stéphane DANIELE C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne directeur@edchimie-lyon.fr
E.E.A.	ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE https://edeea.universite-lyon.fr Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	M. Philippe DELACHARTRE INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 philippe.delachartre@insa-lyon.fr
E2M2	ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION http://e2m2.universite-lyon.fr Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	Mme Sandrine CHARLES Université Claude Bernard Lyon 1 UFR Biosciences Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69622 Villeurbanne CEDEX sandrine.charles@univ-lyon1.fr
EDISS	INTERDISCIPLINAIRE SCIENCES-SANTÉ http://ediss.universite-lyon.fr Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	Mme Sylvie RICARD-BLUM Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
INFOMATHS	INFORMATIQUE ET MATHÉMATIQUES http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	M. Hamamache KHEDDOUCI Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 hamamache.kheddouci@univ-lyon1.fr
Matériaux	MATÉRIAUX DE LYON http://ed34.universite-lyon.fr Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	M. Stéphane BENAYOUN Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 stephane.benayoun@ec-lyon.fr
MEGA	MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE http://edmega.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	M. Jocelyn BONJOUR INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX jocelyn.bonjour@insa-lyon.fr
ScSo	ScSo* https://edsciencessociales.universite-lyon.fr Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	M. Bruno MILLY Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 bruno.milly@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Acknowledgments

First, I would like to thank Esker for giving me the opportunity to pursue my Ph.D. thesis and for funding my work for the last four years. Thanks to Jean-Jacques Bérard and Jérémy Espinas for putting their trust in me while I was an intern at Esker. I would more particularly thank Jérémy for his benevolent supervision and all the insightful talks. During hard times, he knew how to change perspective and go forward. I also thank Clément Sage with whom I co-authored my first paper and traveled to Lausanne.

My thanks also go to Christophe Garcia and Stefan Duffner for their supervision and their advice related to academia. Christophe first convinced me to pursue my scholarship in this thesis and I cannot thank him enough for that. I cannot forget Véronique Églin, which gave me several opportunities to show my work.

I also thank Salvatore-Antoine Tabbone, Jean-Marc Ogier, Thierry Paquet and Aurélie Lemaitre for accepting to sit on my defense jury. And a special one for Thierry and Aurélie for being rapporteurs of my manuscript.

Finally, I would like to thank my family and my girlfriend Mathilde for their foolproof support. A sincere thanks to all my friends who were always here for much-needed breaks and a special one to Victor who introduced me to the Kaffee Berlin. My final thank goes to my dog, Sahel. He truly is the bestest good boy.

Résumé

Chaque jour, d'innombrables quantités de documents sont réceptionnés et traités dans les entreprises du monde entier. Afin de réduire les coûts de traitement associés à chaque document, les plus grandes entreprises se sont résolues à automatiser leur traitement. Dans le meilleur des mondes, le traitement d'un document s'effectue sans aucune intervention humaine : le document est lu par la machine qui en extrait les informations importantes et les transmet au service adéquat. L'état de l'art a rapidement évolué ces dernières décennies, les premiers algorithmes à base de règles écrites manuellement étant devenus des modèles statistiques. Cette thèse se concentre sur les modèles d'apprentissage machine pour l'extraction d'information dans les documents.

De récents progrès en architecture de modèle pour la compréhension du langage naturel ont montré l'importance des mécanismes d'attention. Les transformeurs ont révolutionné le domaine en démocratisant l'utilisation d'attention et en améliorant les pré-entraînements auto-supervisés. Dans une première partie, nous confirmons qu'avec un pré-entraînement sur des documents, les transformeurs sont capables de réaliser des tâches de compréhension de documents avec grande précision. Nous montrons également que, lorsqu'ils sont utilisés pour de l'extraction d'information par classification de mots, les transformeurs apprennent plus efficacement que les modèles utilisant des réseaux récurrents. Les transformeurs n'ont besoin que d'une petite portion du dataset d'entraînement pour atteindre des performances proches du maximum. Ces résultats soulignent l'importance du pré-entraînement auto-supervisé pour l'apprentissage de la tâche finale.

Dans la partie suivante, nous présentons des tâches de pré-entraînement spécifiquement conçues pour mieux préparer le modèle à une distribution de données ciblée tels que les documents d'entreprise. En remarquant certaines de leurs spécificités telles que leur structure tabulaire et la présence d'un grand nombre de valeurs numériques, il est possible de cibler certains savoir-faires utiles pour la compréhension des documents par le modèle. Nous montrons que ces nouvelles tâches de pré-entraînement améliorent les performances, même avec de petits modèles. Ces derniers atteignent des niveaux de performance similaires à ceux de modèles significativement plus gros. Ce type de pré-entraînement spécifique est donc une piste sérieuse pour réduire la taille des modèles utilisés, sans coût supplémentaire lors de l'entraînement supervisé ou de l'inférence.

Enfin, dans la dernière partie, nous portons notre attention sur un des principaux défauts de l'architecture transformeur qui est le coût d'évaluation quand le modèle est utilisé sur de longues séquences. Nous montrons que des architectures efficaces dérivées des transformeurs nécessitent moins de ressources et obtiennent de meilleurs résultats sur de longues séquences. Cependant, à cause de la manière dont le calcul d'attention est modifié, ces modèles plus efficaces souffrent d'une légère perte de performance sur de courtes séquences comparé au transformeur clas-

sique. Ces résultats montrent les avantages d'un système utilisant plusieurs modèles en discriminant sur la taille des séquences à traiter. De plus, ces architectures efficaces ouvrent la possibilité de concaténer des séquences provenant de modalités différentes.

Abstract

Every day, an uncountable amount of documents are received and processed by companies worldwide. In an effort to reduce the cost of processing each document, the largest companies have resorted to document automation technologies. In an ideal world, a document can be automatically processed without any human intervention: its content is read, and information is extracted and forwarded to the relevant service. The state-of-the-art techniques have quickly evolved in the last decades, from rule-based algorithms to statistical models. This thesis focuses on machine learning models for document information extraction.

Recent advances in model architecture for natural language processing have shown the importance of the attention mechanism. Transformers have revolutionized the field by generalizing the use of attention and by pushing self-supervised pre-training to the next level. In the first part, we confirm that transformers with appropriate pre-training were able to perform document understanding tasks with high performance. We show that, when used as a token classifier for information extraction, transformers are able to exceptionally efficiently learn the task compared to recurrent networks. Transformers only need a small proportion of the training data to reach close to maximum performance. This highlights the importance of self-supervised pre-training for future fine-tuning.

In the following part, we design specialized pre-training tasks, to better prepare the model for specific data distributions such as business documents. By acknowledging the specificities of business documents such as their table structure and their over-representation of numeric figures, we can target specific skills useful for the model in its future tasks. We show that those new tasks improve the model's downstream performances, even with small models. Using this pre-training approach, we are able to reach the performances of significantly bigger models without any additional cost during finetuning or inference.

Finally, in the last part, we address one drawback of the transformer architecture which is its computational cost when used on long sequences. We show that efficient architectures derived from the classic transformer require fewer resources and perform better on long sequences. However, due to how they approximate the attention computation, efficient models suffer from a small but significant performance drop on short sequences compared to classical architectures. This incentivizes the use of different models depending on the input length and enables concatenating multimodal inputs into a single sequence.

Contents

Contents	ix
List of Figures	xvii
List of Tables	xix
List of Acronyms	xxi
List of Notations	xxiii
1 Introduction	1
1.1 Context	1
1.1.1 Business Documents	2
1.1.2 Enterprise Resource Planning	2
1.2 Understanding Business Documents	3
1.2.1 Textual information	4
1.2.2 Other information modalities	5
1.2.3 Structure	6
1.2.4 Complexity	7
1.3 Related tasks	8
1.3.1 Layout Analysis	9
1.3.2 Classification	9
1.3.3 Information Extraction	11
1.4 Outline	13
2 Related Work	15
2.1 Introduction to Machine Learning	16
2.1.1 Supervised Machine Learning	17
2.1.2 Artificial Neural Networks	25
2.2 Deep Learning	29
2.2.1 Convolutional Neural Networks	29
2.2.2 Recurrent Neural Networks	34
2.2.3 Transformers	42
3 Data-Efficient Information Extraction from Documents with Pre-Trained Language Models	57
3.1 Introduction	58
3.2 Related works on Information Extraction	59
3.3 Models	62
3.3.1 Encoder	62
3.3.2 Classifier	63

3.4	Datasets	63
3.4.1	Scanned Receipts OCR and Information Extraction (SROIE)	64
3.4.2	Real-world purchase orders (PO-51k)	64
3.5	Experiments	65
3.5.1	Experiment settings	65
3.5.2	Few-shot learning	65
3.5.3	Intermediate learning	68
3.6	Conclusion	69
4	Improving Information Extraction on Business Documents with Specific Pre-Training Tasks	71
4.1	Introduction	72
4.2	Related Work	73
4.2.1	Information Extraction	73
4.2.2	Pre-Training	74
4.3	Models	75
4.3.1	Architecture	75
4.3.2	ConfOpt Post-Processing	76
4.4	Pre-training	77
4.4.1	Numeric Ordering Task	77
4.4.2	Layout Inclusion Task	78
4.5	Datasets	79
4.5.1	Business Documents Collection	80
4.5.2	Business Documents Collection – Purchase Orders	81
4.5.3	ICDAR 2019 – Scanned Receipts	81
4.6	Experiments	81
4.6.1	Post-Processing	82
4.6.2	Business Document-Specific Pre-training	83
4.7	Conclusion	85
5	Long-Range Transformer Architectures for Document Understanding	87
5.1	Introduction	88
5.2	Related Work	89
5.2.1	From Natural Language Processing to Document Understanding	89
5.2.2	Long-Range Transformers	90
5.3	Datasets	92
5.3.1	Business Document Collection - Purchase Orders	92
5.3.2	DocBank	93
5.4	Models	93
5.4.1	LayoutLM	94
5.4.2	LayoutLinformer	95
5.4.3	LayoutCosformer	96
5.4.4	2D Relative attention	97

Contents	ix
<hr/>	
5.5 Experiments	99
5.5.1 Long-Range	99
5.5.2 Relative Attention	103
5.6 Conclusion	104
6 Conclusion	105
6.1 Contributions	105
6.2 Future Work and Perspectives	106
Bibliography	109

List of Figures

1.1	Example of a PO drawn from Esker sample documents. It contains all the necessary information to identify completely the buyer and the list of purchased goods. The bottom-left corner asks for the vendor to acknowledge the reception of this document by sending an order acknowledgment, another type of BD.	4
1.2	Extracted OCR with Google Cloud Vision API ¹ from Figure 1.1 with light preprocessing to improve readability and select valuable information. Each word is a string attached to some location specified by its bounding box. Words are grouped into paragraphs and blocks in a hierarchical manner.	5
1.3	A collection of synthetic purchase orders exemplifying the extensive range of template variations and lexical fields for goods. Despite the theoretical absence of correlation between templates and lexical fields, in practice, they tend to be interdependent due to companies reusing identical templates for their documents.	7
1.4	Two sample pages from DocBank dataset with annotation inserted as a colored overlay. The dataset provides both pixel-level and word-level annotations. Figures were handpicked from [Pfitzmann et al., 2022].	9
1.5	Sample images from RVL-CDIP dataset from 6 out of 16 classes. Both images and OCR extracted text are provided. A wide variety of business document types are represented in this dataset, which is why its documents are sometimes reused for other tasks in other datasets. Figures were handpicked from [Harley et al., 2015]	10
1.6	An example of annotation of Figure 1.1 for IE. 8 header fields and 4 table fields are displayed. Depending on the end-user application, the choice of annotated fields might differ. Because human annotation is costly, annotated fields are often limited to the strict minimum in practice.	12
2.1	A simplified machine learning application life cycle. Once data is collected, cleaned and annotated, models can be trained. Multiple trainings can be done to compare different models' architectures and evaluate variance. Once satisfied with the results, the chosen model is pushed to production, where it can be monitored closely. With time, training data may become old and different from live production data. This phenomenon is called data drift and necessitates new data collection.	16

-
- 2.2 On the left is represented a binary classification task. Each input \mathbf{x}_i is an image, the model is a function that maps any image to its label. On the right is a linear regression performed on a set of 10 sample data points. In this example, each $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}$ is mapped to its target $\mathbf{y}_i \in \mathcal{Y} \subset \mathbb{R}$. This particular model has 2 parameters: the slope and the intercept of the line which are computed based on the sample data. 18
- 2.3 A Perceptron [Rosenblatt, 1958] with 2 input neurons (Left) and a visual representation of its decision boundary once trained on a small dataset (Right). From current standards, it is comparable to a single neuron of an ANN. An input point \mathbf{x} is classified into the positive (resp. negative) class 1 (resp. 0) by looking at its position relative to the hyperplane described by \mathbf{w} and b : if \mathbf{x} is over (resp. under), it is classified positively (resp. negatively). The Heaviside function $\mathbb{I}: x \mapsto \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$ is used to discretize the model's output. As shown on the right, its decision boundary can only be a straight line. This limitation led to the further development of non-linear models. 19
- 2.4 Confusion matrices of 2 models on the same binary classification task. Both models have the same accuracy ($\frac{99+1}{99+1+1+9} = \frac{92+8}{92+8+8+2} = \frac{100}{110}$) but they greatly differ on their error repartition. A quick inspection reveals that model (a) on the left commits most of its errors on false negatives. Model (b) on the right is more balanced at the cost of worse performances for the negative class. Computing the F1-score of the positive class for both models reveals the difference in performance: $\text{F1-score}_a = \frac{1}{6} < \text{F1-score}_b = \frac{8}{13}$. This improvement is balanced by a relatively smaller performance metric on the negative class. This toy example illustrates the precision-recall tradeoff that takes place with unbalanced classification tasks. 21
- 2.5 Fitting 7th degree polynomials on a set of points. Training is achieved by performing ridge regression which regularizes the model's weights by penalizing their magnitude. The amount of regularization is controlled by a parameter λ . No regularization (Red dotted curve) results in an overfitted model that passes close to every point in the training data. However, the learned function does not match the actual structure of the data: it decreases sharply on the left side of the plot while the data points towards a steep increase. Too much regularization (Orange plain curve) constrains the model to a simple constant function. The best results are achieved with an intermediate amount (Green dashed curve) which forces the algorithm to learn a simple yet sufficient representation of the data. 23

- 2.6 Architecture of a MLP with 2 hidden layers and a binary classification head. Bold links between nodes denote a trainable parameter of the model with a total of 41 parameters. Typical implementations of MLPs use matrices of parameters such that a layer is a single matrix multiplication followed by the activation function σ_i 28
- 2.7 Illustration of a convolution operation between a 3×3 filter (Left) and a 5×8 image (Bottom right). Convolution can be represented as overlaying the filter over the image and computing the dot product between the filter and the part of the image underneath. By repeating this operation for every possible position where the filter fits by following the black dotted line, an output image (Top right) is computed where each output pixel is the result of one dot product. In this very example, the filter detects vertical edges in the input image by producing large outputs around those edges. Figure freely inspired of [Géron \[2017, Figures 14.3-14.4\]](#). 30
- 2.8 Different normalizations being performed on a tensor. Values in shades of orange are standardized to the same mean and variance which is computed by aggregating those same values. H and W stand for the height and width of the feature map, while C is the channel dimension and N is the batch size. Figure freely inspired of [Wu and He \[2018\]](#). 32
- 2.9 Same architecture as Figure 2.6 but with dropout [[Srivastava et al., 2014](#)] applied to each hidden layer. Neurons crossed out have been dropped out and are not used by the model during this training step. With a probability $P = \frac{1}{3}$ of dropping each neuron, the architecture of the model is completely different. And because the dropped-out neurons change at each training step, the model must be resilient to this perturbation. 33
- 2.10 The input of the network \mathbf{x} is a sequence $(\mathbf{x}_t)_{1 \leq t \leq N}$. The network can produce a sequence of outputs $(\hat{\mathbf{y}}_t)_{1 \leq t \leq N}$ aligned with the input. Some output of the well ϕ at step t is fed to the next cell at step $t+1$. The compact version (Left) showcases the recurrence in the model whereas the unrolled version (Right) better represents the sequence length dimension and the multiple copies of the recurrent cell. Figure freely inspired of [Olah \[2015\]](#). 34
- 2.11 A bidirectional RNN is composed of two independant RNNs: the forward and backward networks. Like in Figure 2.10, the forward network ϕ^{\rightarrow} treats the input sequence from the beginning to the end. In contrast, the backward network ϕ^{\leftarrow} processes the sequence in reverse order. By combining both forward and backward representations, a bidirectional RNN is able to capture both past and future context into its prediction. 36

- 2.12 Three short sentences (Left) being tokenized and embedded. The first sentence (Top) is tokenized into words while the second (Middle) is tokenized into characters and the third (Bottom) uses sub-word tokenization techniques. Colored squares represent embeddings of each corresponding token. At the cost of a slightly longer tokenized sequence, sub-word tokenization can tokenize unknown words like `embeddings` into known pieces (`embed` and `#dings`). The hash symbol `#` at the beginning of a token denotes the continuation of a word. . . . 38
- 2.13 An encoder-decoder architecture with RNNs. The encoder (Left, in green) processes a context sequence $(t_i)_{1 \leq i \leq N}$ and produces its representation. The decoder (Right, in red) receives the representation of the context sequence in its recurrent input and starts its prediction with a special begin-of-sequence [BOS] token. At each inference step j , the previous decoder's output \hat{z}_{j-1} is appended to its input for the next inference. Multiple strategies exist to select which output token is selected, given the model's probabilities distribution Graves [2012]. 41
- 2.14 An encoder-decoder with attention. The decoder (Right, in red) can retrieve the encoder's (Left, in green) outputs through an attention layer. For readability reasons, only the cell at step j of the decoder is represented. The scalar α_{ij} directly describes how much the decoder at step j pays attention to the encoder's i^{th} output. It is normalized with a softmax operator so that $\sum_i \alpha_{ij} = 1$ and $(\alpha_{ij})_i$ can be interpreted as a probability distribution. 42
- 2.15 Visualization of two attention maps on a translation from English to French task. Because English and French are grammatically close, words are mostly aligned with their direct translation. This alignment results in a bright diagonal almost everywhere. However, some nominal groups are formed backward in French like "European Economic Area" in Subfigure 2.15a where the attention locally takes the shape of an anti-diagonal matrix. Visualizing attention map reveals in Subfigure 2.15b how the model understands the definite pronoun "l'" and associates it to the correct noun "environment". Figures reproduced from Bahdanau et al. [2016]. 43
- 2.16 The original transformer architecture. Originally presented as an encoder-decoder architecture, the encoder (Left, in green) and the decoder (Right, in red) can be used independently. State-of-the-art models stack multiple transformer layers. A decoder-only transformer does not use the multi-head attention that takes the encoder's output. Figure freely inspired of Vaswani et al. [2017] 44
- 2.17 Venn diagram of efficient transformers implementations adapted to long sequences. Figure reproduced from Tay et al. [2022a]. 45

- 2.18 Common pre-training tasks for encoder and decoder architectures. Links in the model’s box mean the model can attend to other inputs to produce its output at a given position. Some links are grayed out as they are not used in this example to solve the task but are nevertheless allowed attention links. Decoders (Left) are trained with an autoregressive LM task in which the model learns to predict the next token based on all previous tokens. Encoders (Right) cannot be trained using the same task as they are aware of both past and future tokens. Instead, given a sequence with corrupted tokens (in purple), they learn to predict the original tokens. 49
- 2.19 Evolution of LLMs sizes in the number of parameters, number training tokens and computational operations involved. Since late 2018 [Devlin et al., 2019] (Left), models’ sizes grew exponentially up to hundreds of billions of parameters [Chowdhery et al., 2022]. The cost of training a model (Right) has also increased by several orders of magnitude in the same period, requiring clusters of GPUs for multiple months and millions of dollars. Data is publicly available on Wikipedia² 50
- 2.20 LayoutMask [Tu et al., 2023] architecture and pre-training tasks. The task named Masked Language Modeling in this figure is considerably harder than classical MLM as whole words are masked instead of tokens. Masked Position Modeling is another MLM-derived task that focuses on layout information. Figure reproduced from Tu et al. [2023]. 52
- 2.21 ViBERTgrid approach to multimodality. A rich multimodal feature map P_{fuse} is computed through a U-shaped CNN, enriched with BERT text embeddings. Figure reproduced from Lin et al. [2021]. . . 53
- 2.22 FormNet uses graph convolutions [Gilmer et al., 2017] to produce a layout-based token representation. This representation is then fed into a transformer alongside the text modality in a similar manner 2D positional encoding is inserted in LayoutLM [Xu et al., 2020]. Figure reproduced from Lee et al. [2022]. 53
- 2.23 DocFormer model and pre-training tasks. Both text and image modalities are trained with a task based on masking either input sequence and predicting what was masked in the first place. However because image embedding does not rely on discrete tokens, the classification is replaced with a global image reconstruction task. Figure reproduced from Appalaraju et al. [2021]. 55
- 2.24 LayoutLMv3 architecture. Text and image are tokenized and fed as a single sequence to the model. Pre-training is done using 3 tasks concurrently: MVLM on the text tokens (Masked Language Modeling in the figure), MVLM on the image tokens (Masked Image Modeling in the figure) and Word Pair Alignment on unmasked text tokens. Figure reproduced from Huang et al. [2022]. 56

3.1	The different architectures used in our experiments for encoding documents. From left to right: Transformer-based LayoutLM Xu et al. [2020] with pre-trained weights, LayoutLM with random initialization and a 2-layer bidirectional LSTM also randomly initialized.	61
3.2	A document sample for each dataset alongside their expected field values to extract. For PO-51k, we show a fictive purchase order due to privacy reasons.	63
3.3	Few-shot extraction performance on the SROIE [Huang et al., 2019] test set for the pre-trained LayoutLM [Xu et al., 2020] against its randomly initialized version and a BiLSTM network.	66
3.4	Few-shot extraction performance on the PO-51K test set for the pre-trained LayoutLM [Xu et al., 2020] against its randomly initialized version and a BiLSTM network.	67
3.5	Test F1 scores of pre-trained LayoutLM when transferring extraction knowledge from SROIE to PO-51k tasks. The IE performance is always improved by resorting to SROIE as an intermediate task, the boost being significant with few available PO-51k documents for fine-tuning.	68
4.1	LayoutLM’s [Xu et al., 2020] MVLM on a document. A small proportion of words are corrupted (shown in purple), and the model is trained to predict the original words. The position of words is never corrupted to incentivize the model to learn correlations between a word’s position and its surrounding 2D context.	75
4.2	A pre-training example with NO task. A random token containing a number is selected, then the target is to predict whether other numbers are smaller or bigger. Some random noise can be added by masking tokens’ textual or spatial representations. Only a small part of the document’s input is represented in this illustration.	78
4.3	A pre-training example with LI task. The coordinates of the purple rectangle are drawn uniformly. Random noise is added by masking the 2D position of some tokens. Only a small part of the document is represented.	79
4.4	A document sample for each training dataset annotated with the expected predictions. For BDC-PO, we replaced the document with a fictive one due to privacy reasons.	80
5.1	Sample pages with colored labels similar to those in the BDC-PO dataset. Both pages come from the same document, the first page is on the left and the last page is on the right. Some information is repeated across pages of a document.	93
5.2	DocBank sample image on the left and its corresponding segmentation on the right. Each color represents one class (black for <i>paragraph</i> , purple for <i>equation</i> , ...).	94

5.3	Illustration of the attention mechanism used in LayoutLM, normalization and multiple heads aside. In this example, $N = 5$ and $D = 2$. Due to the softmax operator, the product $\mathbf{Q}\mathbf{K}^\top$ must be computed, resulting in $O(N^2)$ complexity.	95
5.4	LayoutLinformer attention mechanism. In this example, $N = 5$, $D = 2$ and $M = 3$. Efficient matrix multiplication ordering reduces the complexity to $O(NM)$	96
5.5	LayoutCosformer efficient attention mechanism with $N = 5$ and $D = 2$. The linear similarity enables computing first $\phi(\mathbf{K}^\top)\mathbf{V}$ and factorize $\phi(\mathbf{Q})$ out of the summation of the normalization factor. . .	97
5.6	Contour plots for squirele and cross relative attention bias applied to token “210,80” (bottom-right corner). Because token positions are normalized between 0 and 1000, tokens along the same line cannot fully attend to each other on the left while they are unaffected on the right.	99
5.7	F1-score stacked bar plot of multiple models on BDC-PO. In each document length category, models are in the same order.	101

List of Tables

4.1	Performance comparison on SROIE [Huang et al., 2019] and BDC-PO between multiple post-processing algorithms. The score is computed on the exact match between the prediction and the target string. . .	82
4.2	Model performance when fine-tuning on BDC-PO.	83
4.3	Model performance when fine-tuning on SROIE [Huang et al., 2019].	83
4.4	Model performance comparison with the literature on SROIE. Model sizes are expressed in terms of their number of trainable parameters and modalities are text (T), layout (L) and image (I). Although our contribution is overthrown by larger and T+L+I models in terms of pure performance, it achieves better results than the original LayoutLM _{LARGE} [Xu et al., 2020] which is 3 times bigger. (†) Because ERNIELayout [Peng et al., 2022] is not publicly available, we had to estimate its number of parameters with open-source reproductions. (‡) Those models were published after the first publication of this work.	84
5.1	Duration and memory consumption of the 3 models for various sequence lengths on an inference task.	100
5.2	F1 weighted average for each model and document length categories. All models were first pre-trained on BDC.	102
5.3	Results on Docbank dataset for LayoutLMs and long-range models. † Those models were pre-trained on BDC before finetuning on Docbank.	102
5.4	Macro average F1 score on the Business Orders dataset with 2D relative attention.	103

List of Acronyms

AI Artificial Intelligence	16
ANN Artificial Neural Network	105
BD Business Document	106
BDC Business Documents Collection	92
BERT Bidirectional Encoder Representations from Transformers	75
BiLSTM Bidirectional Long Short Term Memory	105
BPE Byte Pair Encoding	37
CE Cross-Entropy	19
CNN Convolutional Neural Network	88
CPC Cell Position Classification	51
CRF Conditional Random Field	89
CV Computer Vision	87
DU Document Understanding	107
ERP Enterprise Resource Planning	2
GNN Graph Neural Network	53
GPT Generative Pre-trained Transformer	48
GPU Graphics Processing Unit	90
IE Information Extraction	87
LI Layout Inclusion	77
LM Language Modeling	74
LLM Large Language Model	107
LSTM Long Short Term Memory	88
ML Machine Learning	59
MLM Masked Language Modeling	105
MLP Multi Layer Perceptron	25

MSE Mean Squared Error	19
MVLM Masked Visual Language Modeling	99
NER Named Entity Recognition	73
NLL Negative Log Likelihood	19
NLP Natural Language Processing	105
NO Numeric Ordering	77
OCR Optical Character Recognition	106
OOV Out Of Vocabulary	37
PDF Portable Document Format	93
PO Purchase Order	92
QA Question Answering	107
ReLU Rectified Linear Unit	27
RNN Recurrent Neural Network	105
SROIE Scanned Receipt OCR and Information Extraction	81
SVM Support Vector Machine	20
VRD Visually-Rich Document	105

List of Notations

The following notations are used throughout this thesis.

\mathbf{x}	model's input
$\hat{\mathbf{y}}$	model's output
$\boldsymbol{\theta}$	model's weights (ie. parameters)
\mathbf{y}	target
(\mathbf{x}, \mathbf{y})	example data
$\mathbf{x}^{[i]}$	i^{th} coordinate of \mathbf{x}

Sets

\mathcal{X}	input space	$\mathbf{x} \in \mathcal{X}$
\mathcal{Y}	output space	$\hat{\mathbf{y}} \in \mathcal{Y}$ and $\mathbf{y} \in \mathcal{Y}$
\mathcal{W}	weights space	$\boldsymbol{\theta} \in \mathcal{W}$
\mathcal{Z}	example space	$(\mathbf{x}, \mathbf{y}) \in \mathcal{Z}$
\mathcal{S}_n	sample space	$\mathcal{S}_n = \mathcal{Z}^n = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{1 \leq i \leq n}$

Models

$f_{\boldsymbol{\theta}}: \mathcal{X} \rightarrow \mathcal{Y}$	function or model	$f_{\boldsymbol{\theta}}(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}) = \hat{\mathbf{y}}$
$f^*: \mathcal{X} \rightarrow \mathcal{Y}$	target function	$f^*(\mathbf{x}) = \mathbf{y}$
$\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$	loss function	
$L: \mathcal{W} \rightarrow \mathbb{R}^+$	training loss	$L_n(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i)$
$\sigma: \mathbb{R} \rightarrow \mathbb{R}$	activation function	
η	learning rate	

Dimensions

B	batch size
N	sequence length
W, H, C	width, height and channel
D	model's hidden size

Miscellaneous

\odot	element-wise product (Hadamard)
\otimes	convolution
\top	transposition

Introduction

Contents

1.1 Context	1
1.1.1 Business Documents	2
1.1.2 Enterprise Resource Planning	2
1.2 Understanding Business Documents	3
1.2.1 Textual information	4
1.2.2 Other information modalities	5
1.2.3 Structure	6
1.2.4 Complexity	7
1.3 Related tasks	8
1.3.1 Layout Analysis	9
1.3.2 Classification	9
1.3.3 Information Extraction	11
1.4 Outline	13

1.1 Context

A document is a medium to carry any kind of information meant to be read by a human. From simple text messages to complex financial reports, without forgetting newspapers, literature and ciphers. It can be exchanged to share its content with someone else, or stored in order to be able to retrieve it in the future. It took many formats over time. Without being exhaustive, we can cite paper sheets, books, paintings, and nowadays a variety of digital formats: raw text, Portable Document Format (PDF), \LaTeX , and many others. Technology advancements made digital documents easier to store thanks to digital storage prices decreasing at an exponential rate for 40 years [Roser et al., 2013]. The development and adoption of the Internet since 1983 simplified file exchanges around the world, with close to instantaneous communication at almost no cost. Finally, contrary to physical ones, digital documents can be directly processed on computers, with all the benefits that come with it.

1.1.1 Business Documents

Companies send and receive large amounts of documents related to their businesses daily. Documents produced by companies in a professional context are called Business Documents (BDs). There are various BD types, the most common are invoices, purchase orders, contracts, expenses, curriculum vitae, . . . For historical reasons, those documents almost always take the form of printed text on letter-sized paper such that it is always possible to physically print the document if needed. Those documents need to be processed by multiple employees in order to acknowledge their content and make sure the issuer's demands will be fulfilled.

When a supplier sends an invoice to a buyer, multiple steps are involved. An invoice typically contains information about the supplier, which items were bought, in which quantities and the total amount due by the buyer to the supplier. At the reception of the invoice, an accountant fetches related orders if they exist and identifies the service at the origin of the purchase in order to share the content of the invoice. This service acknowledges reception of the purchased goods or notices back discrepancies. Errors or missing items are detected and corrected before validating the invoice. Once everything is sorted out, an accounts payable manager can be involved to validate the payment. The invoice is then properly archived with all previous invoices. This process is complex and involves multiple persons from various services. A recent study by ArdentPartners [Cohen and Bartolini, 2023] evaluates the overall price of processing an invoice to \$ 8.93 on average. This cost includes accounts payable staff wages, printing, mailing and information technology expenditures.

Moreover, the use of BDs is subject to each country's legislation. For example, a standard of electronic invoice called eInvoicing has been adopted by the European Parliament in 2014¹. Its main objective is to provide a common structured format containing all necessary information for invoice processing automation. Because it is made with the intent of being processed by a computer system, eInvoice is not an image or PDF document but rather a structured file format with no direct visual representation. Accounts payable systems implementing eInvoicing can automatically receive and process incoming invoices, without any external intervention by a human operator. This format was adopted in Italy between 2014 and 2019 and is becoming mandatory for French companies between 2024 and 2026 depending on their size². Independently of the document format, physical and digital BDs must be archived for years after their use. This induces another cost for companies that must comply with local legislation and its changes through time.

1.1.2 Enterprise Resource Planning

In order to reduce costs related to document processing, enterprises have increasingly adopted Enterprise Resource Planning (ERP) [Shehab et al., 2004] systems

¹<https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014L0055>

²<https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000046383394>

since the 1990s. An **ERP** is a software running management tools and business functions for a company. It integrates information from multiple departments, from human resources to manufacturing, not to mention order management and accounts payable. By centralizing information from all incoming and outgoing events, the **ERP** accurately knows the state of the company.

An **ERP** effectively reduces friction when processing incoming invoices by providing in a single place all the needed information. Because the **ERP** is aware of past orders, received items and the content of the invoice, it can automatically determine whether the invoice should be validated or not, report any error and notify the relevant manager. However, in order to rely on the **ERP** for this task, it must accurately know the exhaustive state of the company and the new incoming invoice. Most of the remaining burden is to provide to the **ERP** all the relevant information about incoming invoices and received goods to keep it up to date.

Esker³, the funding company for this work, provides a software as a service (SaaS) platform that extends the existing **ERP** with new features and tools by wrapping over it. Their commercial solutions interact with most **ERP** systems and are customizable to the needs of the customer. They provide high-level supervision tools and workflow for the company in addition to document automation capabilities which help reduce processing costs. To further drive down the cost of processing **BDs**, one must better understand incoming documents to automatically feed the **ERP** system without any human intervention.

1.2 Understanding Business Documents

BDs are often described as Visually-Rich Documents (**VRDs**) [Liu et al., 2019a; Garncarek et al., 2021; Cheng et al., 2022] because they contain both textual and visual information. As said in the previous subsection, **BDs** are processed using computers to speed up workflow. Incoming mail is scanned into image or **PDF** files which contain all visual details in the document including the text. Figure 1.1 is an example of synthetic Purchase Order (**PO**) provided by Esker.

In the context of documents, printed or written text is the preferred media to transmit information to a reader. Computers represent text as an array of characters, individually encoded into bytes using one of many encoding standards. The most basic encoding for languages using the Latin alphabet, called **ASCII**, was introduced in the 1960s. It can represent 128 different characters and includes letters (upper and lowercases), numbers and some symbols. Nowadays, the most widely used encoding is **UTF-8**, which is based on the **Unicode**⁴ standard. It can appropriately represent all written languages and domain-specific characters such as math symbols, emojis and basic shapes. Because this representation of text is not directly compatible with the way images are stored, an additional processing step is needed to extract text from images.

³<https://esker.com>

⁴<https://home.unicode.org/>

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

REFITECH FRIDGE SERVICES PTE. LTD.
(GST No. 8886038980)

PURCHASE ORDER

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

No: 317030
Date: 9/3/2018
Currency: SGD
Delivery: ASAP
Terms: COD

Delivery address:
 REFITECH Fridge Services Pte. Ltd.
 Blk 311, Ubi Yam Road 5,
 # 1-11, UbiYamplex 1,
 Singapore 408653
 Tel: 67430000, Fax: 67430001

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

Remarks: _____
 Jeffrey Boh
Authorized Signature

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

www.refitech.com.sg

Figure 1.1: Example of a PO drawn from Esker sample documents. It contains all the necessary information to identify completely the buyer and the list of purchased goods. The bottom-left corner asks for the vendor to acknowledge the reception of this document by sending an order acknowledgment, another type of BD.

1.2.1 Textual information

Extracting text from an image is a task (or a group of tasks) usually referred to as Optical Character Recognition (OCR) [Mori et al., 1999]. One possible process is to detect individual lines, and then segment each of them into individual words and characters. Finally, each character is recognized with various Computer Vision (CV) techniques. Although it is been actively developed and researched since the 20th century, it can still be challenging depending on the conditions.

For example, handwritten character recognition is well-known for being vastly more difficult than classical printed-on paper OCR. This is due to an accumulation of factors like lack of consistency of characters, missing separations between letters, low contrast between ink and paper, ... Current OCR systems perform with high accuracy on BDs for black-on-white printed characters. They are usually evaluated by computing the character (resp. word) error rate which measures the ratio between the number of correctly read characters (resp. words) and the total number of characters (resp. words). Stamps, brand logos, and handwritten annotations often give imperfect results. As shown in Figure 1.2, those systems also include line and paragraph segmentation and sometimes provide some hierarchical analysis of the text structure.

Esker heavily relies on OCR software to process BDs in production. For this reason, this work is mostly constrained to models using textual information provided by OCR and other options were excluded early in the thesis. Their ability to leverage

```

1  {
2    "blocks": [
3      {
4        "paragraphs": [
5          {
6            "words": [
7              {
8                "text": "REFITECH",
9                "position": [109, 81, 411, 134]
10             },
11             {
12               "text": "FRIDGE",
13               "position": [439, 87, 568, 137]
14             },
15             {
16               "text": "SERVICES",
17               "position": [578, 89, 753, 140]
18             },
19             {
20               "text": "PTE.",
21               "position": [760, 93, 840, 141]
22             },
23             {
24               "text": "LTD.",
25               "position": [847, 94, 925, 143]
26             }
27           ],
28         },
29       ],
30     },
31     "words": [
32       {
33         "text": "(",
34         "position": [106, 160, 116, 182]
35       },
36       {
37         "text": "GST",
38         "position": [117, 159, 173, 181]
39       },
40       {
41         "text": "No.",
42         "position": [180, 159, 231, 181]
43       },
44       {
45         "text": "8888035666",
46         "position": [239, 159, 396, 181]
47       },
48       {
49         "text": ")",
50         "position": [399, 158, 407, 180]
51       }
52     ],
53   },
54 ],
55 },
56 ...
57 }
58

```

Figure 1.2: Extracted OCR with Google Cloud Vision API⁵ from Figure 1.1 with light preprocessing to improve readability and select valuable information. Each word is a string attached to some location specified by its bounding box. Words are grouped into paragraphs and blocks in a hierarchical manner.

other information modalities was however studied at length.

1.2.2 Other information modalities

Although the text contains most of the needed information to interpret correctly any document with high confidence, one cannot simply replace a VRD with its extracted text and expect a reader to perform identically. Word positions and alignments alongside visual elements help the reader disambiguate the document interpretation and reading order.

First, most documents use specific word positions and sizes to emphasize some part of the content and drive the reader towards relevant information. A layout defines the structure of the document and implicitly delimitates semantic units like address blocks, tables or important notes. In Figure 1.2, horizontal alignments inside the table help separate neighboring columns and reading figures.

Second, even though BDs rarely include graphical elements because of the business context, they can occasionally use brand logos or accent colors to ease document quick identification. It provides the human reader with instantaneous information about the document type and origin.

⁵<https://cloud.google.com/vision/docs/ocr?hl=en>

1.2.3 Structure

As discussed previously, information is organized toward better readability and efficiency. Text layout defines a clear structure of each document type which derives from a historic use. Because they serve similar purposes, invoices and purchase orders might look identical to the untrained eye. Both are structured in three sections: a header, a table and a footer.

The header contains information relative to the issuer and the vendor such as their name, address and contact details. It also contains global information about the document. There is usually some sort of document number or reference code to uniquely identify this exact document alongside the date of creation. Multiple dates can cohabit and should not be confused with each other. To differentiate each of them, each piece of information or *value* is usually associated with a *key* that disambiguates its meaning.

For example in Figure 1.1, the string “9/3/2018” is preceded by “Date:” which implicitly designs the document creation date. This pattern repeats for other keys: document number (“No:”), currency, delivery date and terms of payment. To further emphasize the pattern repeating, this document also aligns each key-value pair on the colon, which visually guides the reader.

The table is the main body of the document. Depending on the type of **BD**, it can contain various information. It is most of the time organized in a grid-like manner with columns representing a type of information and each line representing a unique item. The first line of a table is the table header which specifies each column name. From one document layout to another (even within the same document type), column names might differ in quantity and order. Depending on the number of lines inside the table, it can span multiple pages. In those cases, the header row is usually repeated at the beginning of each page such that each page can be understood on its own.

In the case of purchase orders, the table lists all purchased goods for this transaction. For a purchase order to be processed, some information is mandatory like item reference ID and quantities purchased. Other *fields* are usually also included like descriptions, unit prices, tax percentages or amounts and total prices for each item. Other types of **BD** might include different fields because the context in which to document is involved is different.

For instance, at first glance, purchase orders and invoices might look similar. In practice, orders focus on which items are purchased and in which quantity for the vendor to prepare the command and send it. An invoice also includes those fields for the buyer to acknowledge the reception of goods and the prices to pay the vendor. Although the price usually appears in a purchase order, it is not the most important element.

Finally, the footer follows the table at the bottom of the page. It contains overview information such as gross and net total amounts and taxes. It might also contain some free text remarks, instructions or words left by the issuer. In Figure 1.1, the delivery address block is specified in the footer, probably to optimize space. In

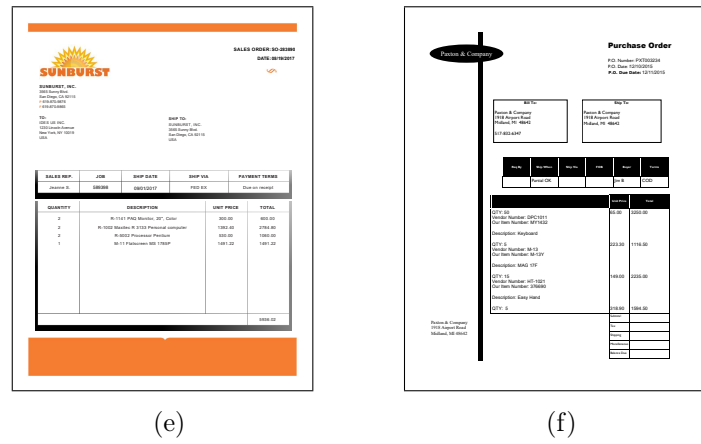
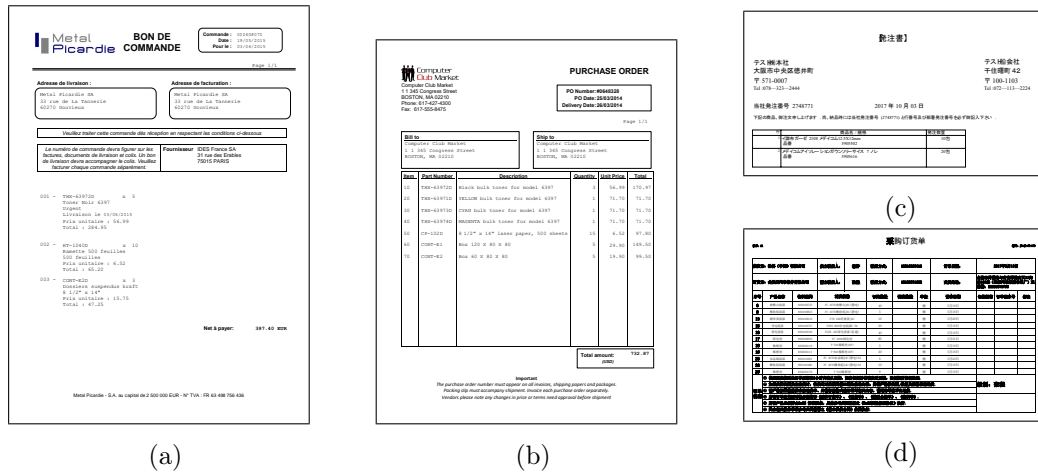


Figure 1.3: A collection of synthetic purchase orders exemplifying the extensive range of template variations and lexical fields for goods. Despite the theoretical absence of correlation between templates and lexical fields, in practice, they tend to be interdependent due to companies reusing identical templates for their documents.

another layout, it could have been in the header.

1.2.4 Complexity

Although BDs are structured, they come in a variety of *templates*, cultures and conditions. Before considering the content itself, because some documents still take their origin in a physical format like paper, they can be damaged or altered before being digitalized. Bad prints, stains, or folded paper negatively impact OCR systems. The most affected documents are those old enough to suffer from time or not important enough to be taken care of like expense receipts. When applying OCR to extract textual features, many parameters need to be taken into consideration. Paper size and shape or rotations are now fully supported by OCR. However, heterogeneous lighting, text written in multiple directions, folded paper [Jian Liang

et al., 2005] or simply handwritten text [Memon et al., 2020] can still be challenging and one should not always expect perfect results.

Even once the document is fully digitalized with text available, many factors can negatively impact BD understanding. Figure 1.3 shows 6 examples of hand-made purchase orders that try to mimic a variety of real-life documents. When working with textual information, it is important to identify the language used. For a document to be understood by both the issuer and the receiver, BDs are always written in a common language. English is often privileged for international trade, but documents that stay inside a country are mostly written in that country’s language. In addition to language, each country has its own culture that influences how BDs are represented. Units, dates and number formats, addresses, phone numbers and currencies are heavily influenced by the document’s culture and vary between countries. For example, dates can be written under multiple similar but different formats. In Europe, the format “dd/mm/yyyy” is used where “dd” is day of the month in 2 digits (with leading 0 if needed), “mm” is month number in 2 digits and “yyyy” is the year in 4 digits. A forward slash “/” is used as a separator, but the hyphen “-” is also common. However, in the USA the format “mm/dd/yyyy” is preferred. The confusion between day and month numbers sometimes can only be avoided thanks to the knowledge of the issuer culture. It is also important to notice BDs are technical documents, with many liberties taken with grammar. In order to keep documents concise, a minimum number of words is used to convey the semantic message. BDs use extensively abbreviations (eg. “QTY” for quantity as seen in Figure 1.3f) and acronyms (eg. “PO” for purchase order seen in Figure 1.3b).

Finally, within a single country, not all documents are structured identically. Whether it be for developing a brand identity or simply to be recognized among others, most issuers personalize their BDs by using colors, logos and fonts, and by emphasizing the table structure with borders. In the end, not 2 documents look the same in Figure 1.3. It helps separate quickly different documents but sometimes makes reading more difficult due to the template differences. The variety of column orders and names makes table parsing a not-so-obvious task for a human reader and a difficult task to automate with an algorithm.

1.3 Related tasks

Document understanding is a large research topic with many intermediate tasks involved. Its automation leverages all sources of information discussed so far and organizes them cleverly in order to better understand a document. OCR was for a long time considered a research topic in itself, resulting in the state-of-the-art tools we have today. Only more complex scenes and images OCR are researched today [Mouchere et al., 2016; Reddy et al., 2020]. Some of the following tasks can also be performed using visual information only. It has the benefit of not relying on a costly OCR system which often drives the cost down and enables real-time processing.

● Text
 ● Caption
 ● List-Item
 ● Formula
 ● Table
 ● Picture
 ● Section-Header
 ● Page-Header
 ● Page-Footer
 ● Title

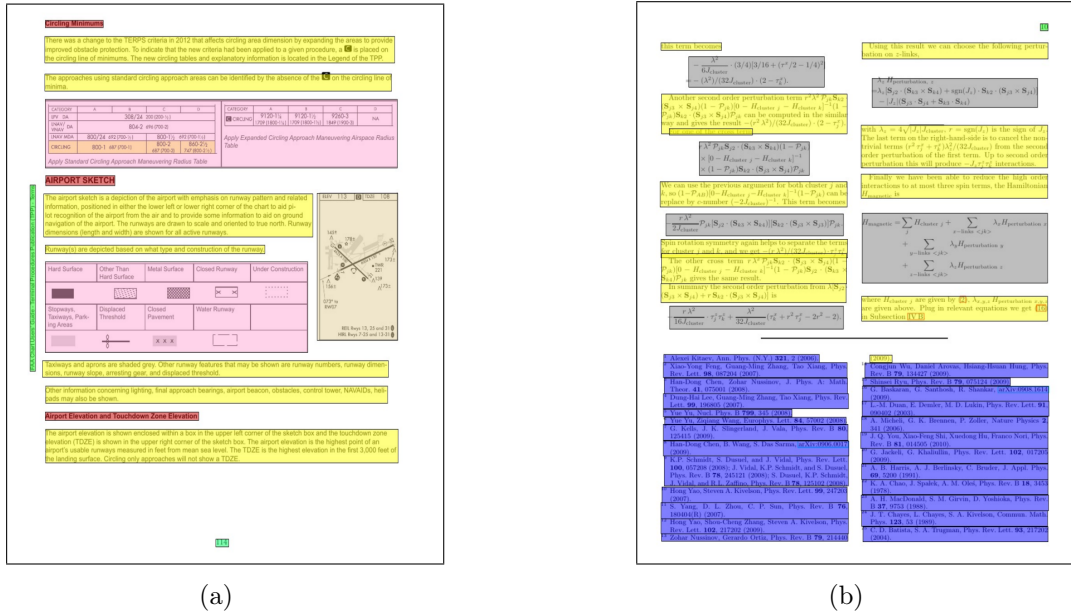


Figure 1.4: Two sample pages from DocBank dataset with annotation inserted as a colored overlay. The dataset provides both pixel-level and word-level annotations. Figures were handpicked from [Pfitzmann et al., 2022].

1.3.1 Layout Analysis

Sometimes done alongside OCR, document segmentation or layout analysis purpose is to extract pre-defined semantic structure in VRDs. Detecting paragraphs, titles and figures helps understanding documents with complex layouts. Multiple datasets exist but most are limited in size because their labeling relies on human annotation. PubLayNet [Zhong et al., 2019] was the first to provide accurate annotations for over 1 million documents. Since then, other datasets have emerged for more diverse and complex layouts [Li et al., 2020; Pfitzmann et al., 2022]. They often use the document’s source code to extract layout semantic information about its content and label areas in the document accordingly as shown in Figure 1.4. Those labels can then be used to fit other algorithms’ parameters.

For example, from a document’s \LaTeX sources, it is possible to automatically label titles, paragraphs, figures and captions. This is the method used by DocBank [Li et al., 2020], which relies on arXiv⁶ to obtain a large amount of source code.

1.3.2 Classification

Classifying VRDs into their various types is a simple useful task. It can involve any combination of visual and textual features. Because text can only be extracted

⁶<https://arxiv.org>

through an OCR, which introduces additional latency and computational cost, it is sometimes chosen to only work on visual information. Algorithms classifying documents into predefined classes can be used as part of a triage system, filtering and separating documents by type before further specific processing.

The reference public dataset for document classification is called RVL-CDIP [Harley et al., 2015]. It contains 400000 documents distributed in 16 classes, more details are available in Figure 1.5. Documents originally come from a larger collection [Lewis et al., 2006] (IIT-CDIP) that include documents that fall into the public domain after legal proceedings against the tobacco industry. Unfortunately, those documents are becoming outdated because of template distributions shifting over time with more modern layouts.

In real life, it is sometimes useful to perform classification without knowing all classes

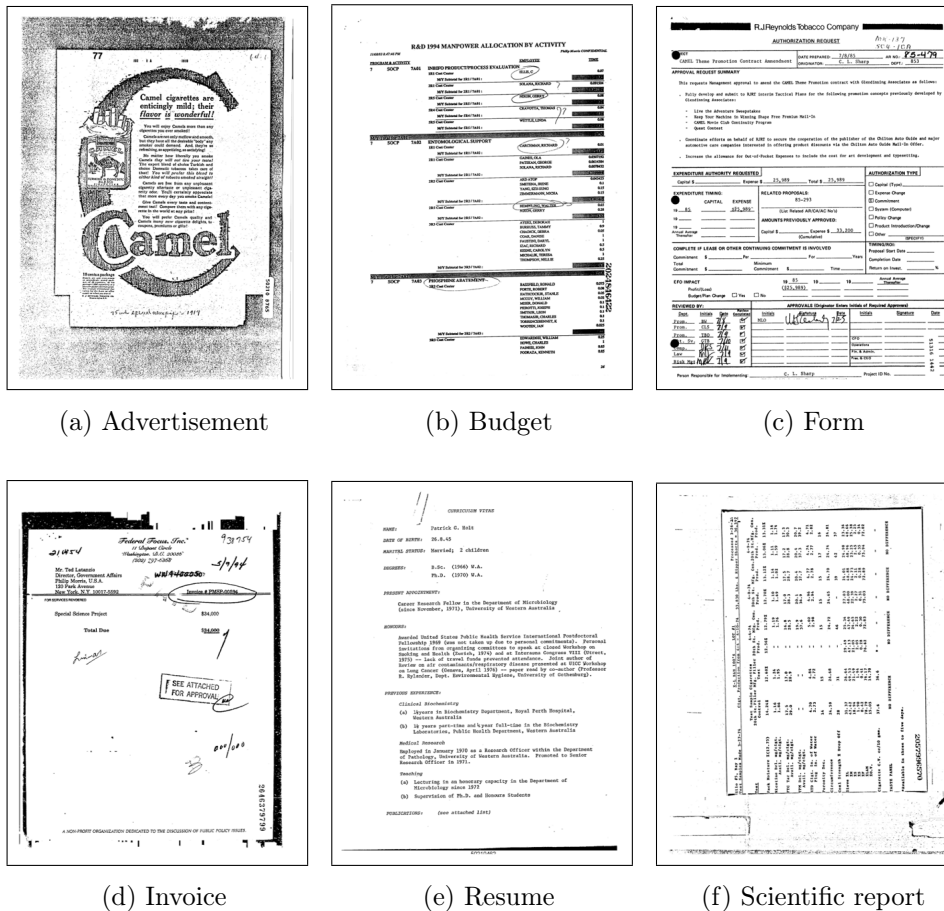


Figure 1.5: Sample images from RVL-CDIP dataset from 6 out of 16 classes. Both images and OCR extracted text are provided. A wide variety of business document types are represented in this dataset, which is why its documents are sometimes reused for other tasks in other datasets. Figures were handpicked from [Harley et al., 2015]

from the beginning. Classes can be added or deleted at any time, fitting the actual needs. This task is much more complex than its classical counterpart, with many specific algorithms developed specifically for Online Learning [Hoi et al., 2021].

1.3.3 Information Extraction

Information Extraction (IE) is a wide task that can be expressed in many ways. Its objective is to, given a document, identify key information and structure it. It has many possible applications when considering how VRDs are processed. As previously said, incoming documents are digitalized and parsed in order to fill ERP forms with accurate information. High quality IE enables fully automated document processing which reduces related costs drastically.

The list of key information varies with document type and context. Some fields might be mandatory in a precise business context but optional in another. In the case of purchase orders, for example, fields can be grouped into 2 categories: header and table fields. Header fields usually only appear once in the whole document but might be repeated identically for long multipage documents. They can appear either in the header or footer of the document. The most common mandatory header fields are document number, gross and net total amounts, date and company name. Table fields on the other hand are structured into a table and are repeated as many times as there are line items ordered. Purchase orders usually include quantity, unit and total price, description, ID and tax for each item purchased.

One part of the IE task is the formatting applied to the extracted values. Depending on the chosen output structure, each field might be typed and formatted precisely to simplify further processes. Figure 1.6 shows extracted information on the example document previously seen in Figure 1.1. In this example, the date is formatted to international standard ISO 8601⁷ which specifies with no possible ambiguity how dates should be written. Dates must follow the pattern “YYYY-mm-dd” which conveniently allows to chronologically sort dates with a lexicographic sort operating on the sequence of characters. To produce this format, it can either be asked directly to produce the formatted date string, or identify individually each sub-field (i.e. day, month and year) and then format with light postprocessing. The same principle applies to figures that are stored as numbers (whole or floating point values) in order to hide the complexity of number formats.

Depending on the method used to perform IE, a postprocessing step might be needed to create the appropriate output format. For example, some field instances might be composed of multiple words but should be considered a single instance. A date written “january 1st, 2023” is technically composed of 3 words but describes a single date instance. On another note, when working with line items, it is often useful for the end user to group table fields by line. It can be complex depending on the layout of the document, with line items spanning multiple physical lines, or irregular spacing between lines. Some specific algorithms can perform most of the postprocessing by themselves by learning to produce the desired output format [Palm et al., 2018].

⁷https://en.wikipedia.org/wiki/ISO_8601

- Date ○ Currency ○ Order Number ○ Delivery address
- VAT ○ Total excl. VAT ○ Total incl. VAT ○ Vendor address
- Item Quantity ○ Item ID ○ Item unit price ○ Item total price

REFITECH FRIDGE SERVICES PTE. LTD.
(GST No.: 88803680)

PURCHASE ORDER

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

No: 817020
Date: 9/3/2018
Currency: SGD
Delivery: ASAP
Terms: COD

Item	Quantity	Description	Unit Price	Total Price
1	10	T6F.L.DN15 5.015.F.L.K	\$ 101.31	\$ 1,013.10
2	4	T6F.L.DN25 5.025.F.L.K	\$ 153.04	\$ 612.16
3	2	T6F.L.DN32 6.032.F.L.K	\$ 172.97	\$ 345.94
4	10	T6F.L.DN15 6.015.F.L.K	\$ 86.55	\$ 865.50
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	\$ 210.35	\$ 631.05

Delivery address:
REFITECH Fridge Services Pte. Ltd.
Blk 311, Ubi Yam Road 5,
#1-11, UbiYamplex 1,
Singapore 680883
Tel: 67430000, Fax: 67430001

Subtotal: \$ 4,353.20
GST 7%: \$ 304.28
Grand Total: \$ 4,787.33

Remarks: _____
Jeffrey Boh
Authorized Signature

*Please send us your order acknowledgement upon receipt of the P.O.
www.refitech.com.sg

Figure 1.6: An example of an annotation of Figure 1.1 for IE. 8 header fields and 4 table fields are displayed. Depending on the end-user application, the choice of annotated fields might differ. Because human annotation is costly, annotated fields are often limited to the strict minimum in practice.

There is a great variety of datasets related to IE. Scanned Receipt OCR and Information Extraction (SROIE) is a document dataset including an IE task was introduced for a competition at ICDAR 2019 [Huang et al., 2019]. It contains about 1000 scanned expense receipts, with both image and text modalities. The extraction task focuses on 4 header fields: company name, address, date and total amount. Model iterations quickly revealed the dataset’s limitations: OCR mismatches, a small number of templates or fields, etc. It is now supplanted by newer datasets including more complex fields to extract. CORD [Park et al., 2019] introduces a hierarchical extraction task with more than 40 fields. Other document types have been used for IE: FUNSD [Jaume et al., 2019] uses forms, Kleister [Stanisławek et al., 2021] uses charity reports and DocILE [Šimsa et al., 2023] uses invoices and orders.

Although there are multiple datasets available, most of them only contain a small number of documents, only about a thousand. This is mostly due to documents containing personal and private sensitive data which cannot be disclosed to the public. Some public datasets however offer large amounts of public training documents with

complex annotated tasks. Mathew et al. [2021] collected 13k heterogeneous documents and annotated them for IE as a Question Answering (QA) task [Gardner et al., 2019]. DocVQA, their proposed dataset, contains 50k question-answer couples, each relative to a single document. Various question types help increase the difficulty of the task without needing more documents to challenge current IE models. Recently in early 2023, dataset DocILE [Šimsa et al., 2023] was released under some non-disclosure constraints. It contains 7k annotated BDs, 100k synthetic documents and 1M unlabeled data, enough to perform pre-trainings that need huge collections of unlabeled data.

Prior to DocILE, many large private IE datasets already existed for business documents. Companies providing services related to BD processing used large annotated datasets to train models automating tasks like IE. In order to avoid labeling each document individually, which would not scale to larger datasets, Esker uses annotations provided by the end user. Indeed, any company employee feeding a document’s information into an ERP solves an IE problem. By saving the output given by the employee for each document, a company can accumulate a dataset with no additional cost. By aggregating the work of multiple users across multiple companies and over multiple years, Esker grew a large collection of BD containing several millions of labeled purchase orders and invoices. However, because annotation is done by multiple users with different needs and objectives, the resulting dataset needs to be extensively cleaned and filtered before use. By this mean, large private datasets for IE were built, at a fraction of the cost by many independent companies [Palm et al., 2017; Katti et al., 2018; Sage et al., 2019].

1.4 Outline

The remaining of this thesis first establishes a detailed history and related works of models able to perform document understanding tasks.

Chapter 2 tells the story of document automation: from the good old rule-based model to the current, state-of-the-art, pre-trained, deep learning model. It details how and why statistical models and machine learning overcame the increasingly difficult tasks of the previous decades in both academia and industry. Their capacity to adapt to a wide range of tasks makes Machine Learning (ML) models the perfect match to tackle both generic and very niche problems. This work focuses on a specific architecture of Artificial Neural Network (ANN) called *transformer* [Vaswani et al., 2017] which has, since its release in late 2017, beaten every competitor in most language-related tasks.

The raw performance of transformers had already been shown on document-related tasks before our work. In Chapter 3, we assess the capacity of pre-trained transformers in a data-constrained situation. This study was motivated by the scarcity of large, publicly revealed, labeled documents on most document-understanding tasks. A data-efficient model means it requires a smaller dataset to first train, opening up opportunities for an industrial actor wanting to develop a new feature.

Using a public receipt dataset, we quantified the efficiency difference between pre-trained transformers and the previous state-of-the-art. Pre-trained models revealed to require much fewer training examples than their competitors to reach close to maximum performances.

Acknowledging the advantages of pre-trained models, we focused on ways to improve the pre-training for **BDs** understanding in Chapter 4. The pre-training relies on pretext tasks that help the model build useful representations of inputs. Pretext tasks from the literature were fitted for Natural Language Processing (**NLP**) involving sentences with rich context which is rarely the case in **BDs**. By introducing specialized tasks for **BDs**, we significantly improved the performance on downstream **IE**.

Although very capable and performant, transformers are limited in their ability to process long sequences. Their memory requirements for large inputs limit their application to smaller sequences and single-paged documents. In Chapter 5, we take a look at alternative architectures more efficient with longer sequences. They prove to be more effective on long documents but the required approximations also hurt the performances on short documents.

Finally, Chapter 6 wraps up this thesis and provides some perspectives for future work.

Related Work

Contents

2.1 Introduction to Machine Learning	16
2.1.1 Supervised Machine Learning	17
2.1.2 Artificial Neural Networks	25
2.2 Deep Learning	29
2.2.1 Convolutional Neural Networks	29
2.2.2 Recurrent Neural Networks	34
2.2.3 Transformers	42

In this Chapter, we draw a picture of the past and recent literature on the topic of document understanding. A historical entry point to document understanding is the document’s image classification [Chen and Blostein, 2007]. The research community explored classifiers working with visual [Byun and Lee, 2000], textual [Li and Jain, 1998; Manevitz and Yousef] and layout-based [Eglin and Bres, 2003] features. Depending on the features used to represent the document, a variety of Machine Learning (ML) algorithms were tested from decision trees [Esposito et al., 2000] to Markov models [Hu et al., 1999] and small neural networks [Cesarini et al., 2003].

On the other hand, document Information Extraction (IE) takes its roots in the Natural Language Processing (NLP) research community and the Message Understanding Conference (MUC) competition [Chinchor, 1998]. State-of-the-art quickly evolved from rule-based systems, either manually handcrafted [Appelt et al., 1993] or automatically learned [Soderland, 1999], to statistical models. Those statistical models such as Hidden Markov Models (HMMs) [Baum and Petrie, 1966; Seymore and Rosenfeld, 1999], Maximum Entropy Markov Models (MEMMs) [McCallum et al., 2000] and Conditional Random Fields (CRFs) [Lafferty et al., 2001] can capture complex patterns given enough data samples.

Despite the advantages of the more recent statistical ML-based models, companies kept relying on rule-based architectures [Dengel and Klein, 2002] until 2010s [Chiticariu et al., 2013]. The recent development of Artificial Neural Networks (ANNs) and deep learning completely superseded previous algorithms. After an introduction on ML and ANN in section 2.1, we will focus on the recent growth of deep learning in section 2.2

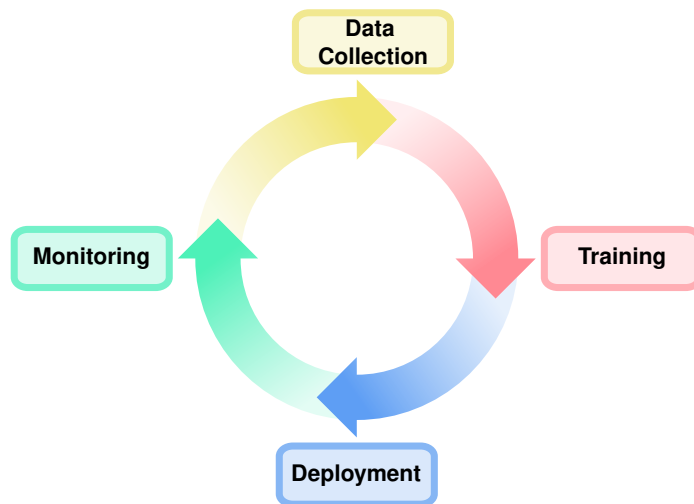


Figure 2.1: A simplified machine learning application life cycle. Once data is collected, cleaned and annotated, models can be trained. Multiple trainings can be done to compare different models’ architectures and evaluate variance. Once satisfied with the results, the chosen model is pushed to production, where it can be monitored closely. With time, training data may become old and different from live production data. This phenomenon is called data drift and necessitates new data collection.

2.1 Introduction to Machine Learning

Mathematical notations used in this chapter and the following are available in page xxiii.

Marvin Minsky once said Artificial Intelligence (AI) is “the science of making machines do things that would require intelligence if done by men.” [Minsky, 1968]. Because it is defined relative to our current knowledge and abilities, what falls under the definition of AI is subject to change with time. ML is a subset of AI which develops systems able to learn patterns from raw data. Using provided data, ML models can improve their predictions and perform tasks they were not explicitly programmed to do. ML techniques allow us to tackle tasks too hard to solve with classical programs. A ML model typically uses parameters (also known as *weights*) that can be tuned to produce the desired result. ML models usually follow a life cycle as described in Figure 2.1. During the training, those parameters are modified according to the learning algorithm using the training data. The model and its parameters are then frozen before being used for inference in production.

ML has been successfully used in many domains with various applications in Computer Vision (CV), NLP and time series forecasting. Document understanding was not left out, capitalizing on both improvements in CV and NLP. ML applications can be categorized into the following paradigms:

- *Supervised* learning refers to a learning process with labeled examples. The

model is learning a parametrized function that produces the desired output for each input in the dataset. In practice, such a function is rarely obtainable. Instead, the trained model is the solution to an optimization problem that involves every example in the dataset. Its objective is to minimize the error in the sample data. Given enough labeled data, supervised learning can be very effective at classification and regression tasks.

- *Unsupervised* learning can be applied to data without any labels involved. It focuses on learning a latent structure of the data in order to better understand it. Unsupervised learning can be used to perform clustering, dimensionality reduction, anomaly detection or even synthetic data generation.
- *Reinforcement* learning enables training autonomous agents that interact with an environment. Through a reward system, the agent learns which behavior is desired and explores its environment through trial and error. Models trained in a simulated environment can be transposed into our physical world. Reinforcement learning has also been proven effective in learning to play games at a high level.
- *Self-supervised* learning sits in between supervised and unsupervised learning. Like unsupervised methods, self-supervised algorithms work with unlabeled data. They learn a representation of the data by creating labeled data using dummy tasks and performing supervised learning on the newly labeled examples. Self-supervised methods allow leveraging large amounts of raw data without the effort of labeling.
- A variety of other paradigms have so far seen less traction but are nonetheless being actively researched. For example, *semi-supervised* learning uses labeled and unlabeled examples to incrementally learn and label the unlabeled examples. *Online* learning disrupts the ML life cycle by continuously training the model with new incoming data. Finally, *federated* learning studies how to collaboratively learn between multiple actors without sharing all the data.

Supervised and self-supervised learning and their applications in document understanding will be further developed in this chapter.

2.1.1 Supervised Machine Learning

Supervised machine learning is a way to perform machine learning by teaching the model its behavior. The desired task is represented by a mapping from an input space \mathcal{X} to an output space \mathcal{Y} . Any input $\mathbf{x} \in \mathcal{X}$ is uniquely associated to an output $\mathbf{y} \in \mathcal{Y}$ by the target function $f^*: \mathcal{X} \rightarrow \mathcal{Y}$. A tuple $(\mathbf{x}, \mathbf{y}) \in \mathcal{Z}$ forms a training example. Those training examples are grouped together into a dataset $\mathcal{S}_n = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{1 \leq i \leq n}$ which contains n examples. Some task examples are available for illustration in Figure 2.2.

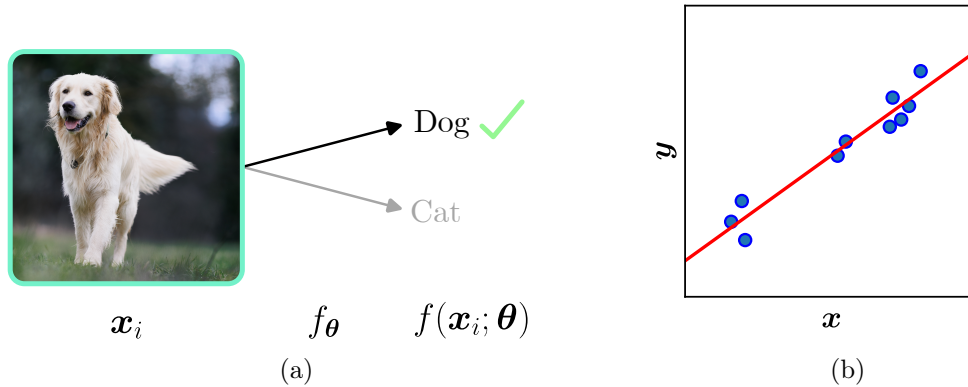


Figure 2.2: On the left is represented a binary classification task. Each input x_i is an image, the model is a function that maps any image to its label. On the right is a linear regression performed on a set of 10 sample data points. In this example, each $x_i \in \mathcal{X} \subset \mathbb{R}$ is mapped to its target $y_i \in \mathcal{Y} \subset \mathbb{R}$. This particular model has 2 parameters: the slope and the intercept of the line which are computed based on the sample data.

The function f^* is the target of the learning process, it exists but its expression is not known and might not be easily expressible. Instead of looking for its exact expression, supervised learning provides tools to approximate f^* . The model is in charge of this approximation. A model can be seen as a function $f_\theta: \mathcal{X} \rightarrow \mathcal{Y}$ that produces an output for any input given with free parameters $\theta \in \mathcal{W}$ where \mathcal{W} is the space of all possible parameters for the model. A simple model that can perform binary classification is described in Figure 2.3 as an example. A learning process is necessary to adequately fit the parameters of the models to the desired task. In other words, the objective of the learning process is to find θ^* such that for any $(x, y) \in \mathcal{Z}$.

$$f_{\theta^*}(x) = f(x; \theta^*) = \hat{y} \approx y = f^*(x) \quad (2.1)$$

Loss functions

To achieve this, a loss function $\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ measures the error made by the approximation \hat{y} compared to y . Many such error functions have been proposed, some better fitted for regression or classification, for discrete or continuous spaces. Loss functions are chosen to respect some properties:

- ℓ should have 0 for its lower bound.
- ℓ should reach its lower bound iif. $\hat{y} = y$, which means $\ell(\hat{y}, y) = 0$.
- Although often considered neat, it is not necessary for ℓ to be symmetric.
- And most importantly, ℓ must be differentiable with respect to its parameters.

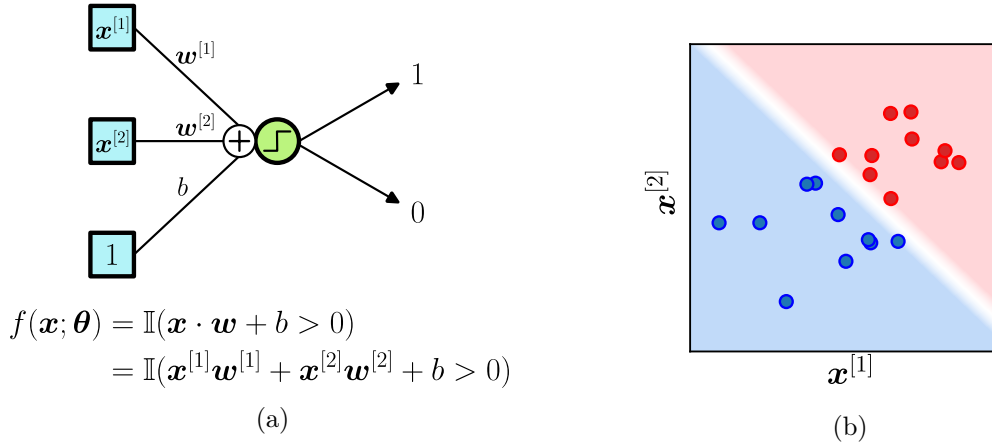


Figure 2.3: A Perceptron [Rosenblatt, 1958] with 2 input neurons (Left) and a visual representation of its decision boundary once trained on a small dataset (Right). From current standards, it is comparable to a single neuron of an ANN. An input point \mathbf{x} is classified into the positive (resp. negative) class 1 (resp. 0) by looking at its position relative to the hyperplane described by \mathbf{w} and b : if \mathbf{x} is over (resp. under), it is classified positively (resp. negatively). The Heaviside function $\mathbb{I}: x \mapsto \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$ is used to discretize the model's output. As shown on the right, its decision boundary can only be a straight line. This limitation led to the further development of non-linear models.

Common loss functions include Mean Squared Error (MSE) which is often used in regression problems. MSE $\ell(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$ penalizes greater error due to its quadratic nature. Contrary to the MSE, the Mean Absolute Error (MAE) will be less influenced by outliers. It uses L1 (ie. absolute value) norm instead of the squared L2 norm which results in $\ell(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_1$. A model trained to minimize the MSE will tend to predict the average of the true data distribution while the mean absolute error will predict the median value [Bengio et al., 2017, p. 175]. Finally, Cross-Entropy (CE) is widely used in classification problems. In a problem with c classes, it is expressed $\ell(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{i=1}^c \mathbf{y}^{[i]} \log(\hat{\mathbf{y}}^{[i]})$. Where $\hat{\mathbf{y}}^{[i]}$ is the the output *logit* of the model for the i^{th} label. In particular, if $\mathbf{y} \in \{0, 1\}^c$, it coincides with the Negative Log Likelihood (NLL).

On top of a loss ℓ , the overall error made by a model $f_{\boldsymbol{\theta}}$ can be computed by aggregating all individual losses for every example in the dataset. The training loss $L: \mathcal{W} \rightarrow \mathbb{R}^+$ can be defined multiple ways, the simplest being the mean of all losses over the dataset:

$$L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i) \quad (2.2)$$

Optimization

The training loss tracks the progression of the model towards the target function f^* . The closer the training loss is to 0, the closer the model behaves to the target. The supervised learning task can be summed up as solving the following optimization problem:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta} \in \mathcal{W}} L(\boldsymbol{\theta}) \quad (2.3)$$

with $f_{\boldsymbol{\theta}^*}$ being the resulting model which minimizes the loss. It is important to note that $f_{\boldsymbol{\theta}^*}$ and f^* are not necessarily the same, $f_{\boldsymbol{\theta}^*}$ is rather an approximation of f^* over the training dataset.

For simple and constrained models, $\boldsymbol{\theta}^*$ can be exactly expressed. For example, in the case of a linear regression model $f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{x} \cdot \boldsymbol{w}$, where \boldsymbol{w} are the weights of the model, the optimal parameters that minimize the MSE can be expressed in term of the inputs and targets of the training data [Bengio et al., 2017, p. 105-106] with a simple formula. More complex models have been proposed, from kernel-based models (eg. Support Vector Machines (SVMs) [Vapnik et al., 1996]) to tree-based architectures (eg. decision trees [Quinlan, 1986] and random forests [Breiman, 2001]). However, those types of optimization strategies constrain the type of model used in order to make the optimization tractable. A more generic method that works for any parametric function $f_{\boldsymbol{\theta}}$ is to perform gradient descent.

Gradient Descent

Gradient descent and all gradient-based optimizers are based upon the computation of the gradient of the training loss with respect to each model's parameter. Model's parameters are updated using equation (2.4) where $\eta \in \mathbb{R}^+$ is the learning rate: a real number that is used as a scaling factor. $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1})$ is the gradient with respect to the parameter.

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1}) \quad (2.4)$$

By updating the parameters with the aggregated gradient over the entire dataset, the process is guaranteed to converge to a minimum. It must be a global minimum if both the parameter space and the training loss are convex, otherwise, it could converge to a local minimum. If the dataset contains lots of examples, the computation of $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1})$ is slowed down. Stochastic gradient descent updates the parameters with every data example encountered. However, the process is no longer guaranteed to converge and the resulting parameters are subject to randomly selected examples. To smooth out each update, it is common to perform mini-batched stochastic gradient descent instead. In equation 2.5, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1})$ is replaced by aggregating over a smaller sample of the dataset $\frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \ell(f(\mathbf{x}_{\pi(i)}; \boldsymbol{\theta}_{t-1}), \mathbf{y}_{\pi(i)})$ where π is some random permutation function of $\llbracket 1, n \rrbracket$ and $m \ll n$ is the size of the minibatch

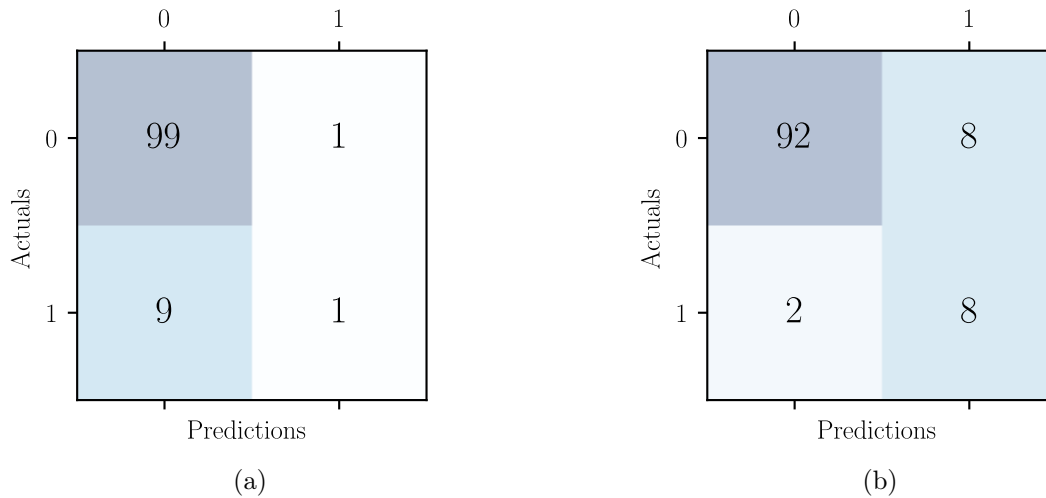


Figure 2.4: Confusion matrices of 2 models on the same binary classification task. Both models have the same accuracy ($\frac{99+1}{99+1+1+9} = \frac{92+8}{92+8+8+2} = \frac{100}{110}$) but they greatly differ on their error repartition. A quick inspection reveals that model (a) on the left commits most of its errors on false negatives. Model (b) on the right is more balanced at the cost of worse performances for the negative class. Computing the F1-score of the positive class for both models reveals the difference in performance: $\text{F1-score}_a = \frac{1}{6} < \text{F1-score}_b = \frac{8}{13}$. This improvement is balanced by a relatively smaller performance metric on the negative class. This toy example illustrates the precision-recall tradeoff that takes place with unbalanced classification tasks.

being aggregated. It provides both stable convergence and speed of computation by controlling the size m of the mini-batches.

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \ell(f(\mathbf{x}_{\pi(i)}; \boldsymbol{\theta}_{t-1}), \mathbf{y}_{\pi(i)}) \quad (2.5)$$

Because complex models' training losses are rarely convex, the surface of parameters over which the optimization is being done contains many pitfalls in which simple gradient descent methods fall. Many upgrades of gradient descent exist, some include an adaptive learning rate conditioned by previous steps [Hinton et al., 2013; Duchi et al., 2011], others use momentum which builds up with past iterations [Kingma and Ba, 2017]. In practice, those methods often converge quicker and towards better local minima. Some of the improvements can be explained by the ability of the iterative process to climb up some hills thanks to momentum instead of directly converging to the closest local minimum [Sun et al., 2019]. Other methods using second order have been proposed, they can be advantageous because they use a finer representation of the curvature of the surface. However, although each step of parameter update is better, it is also more computationally expensive. The most known algorithm is Newton's method, but it isn't used in practice for complex models.

Metrics

If the training loss is a good objective to obtain appropriate model parameters, it is hardly a good measure of the model's performance. For instance in classification tasks, accuracy is way more interpretable than CE to judge the quality of a model. Those functions used to evaluate the quality of a model are called metrics. The optimization process does not directly try to optimize them, but they are used to compare trained models and summarize their performance. They are usually chosen for their interpretability, but some focus more on not being easily cheated on. In classification tasks, the most natural metric is accuracy. It is simply the proportion of correct predictions made by the model.

$$\text{accuracy} = \frac{n_{\text{correct}}}{n_{\text{total}}} \quad (2.6)$$

For a given label, it is sometimes important to differentiate to types of errors made by the model. The first type is a false positive, making a positive prediction for a given example while it is not. The second type is a false negative which is making a negative prediction for an example labeled positively. False positives (resp. false negatives) can be measured with precision (resp. recall). For conciseness true positive, false positive, false negative and true negative are abbreviated respectively TP, FP, FN, TN in (2.7).

$$\text{precision} = \frac{n_{\text{TP}}}{n_{\text{TP}} + n_{\text{FP}}}, \quad \text{recall} = \frac{n_{\text{TP}}}{n_{\text{TP}} + n_{\text{FN}}} \quad (2.7)$$

Precision and recall are important to monitor as they separate first and second-type errors. The F1-score [Murphy, 2022, p. 173] aggregates both precision and recall into a single metric. Because it is computed with the harmonic mean between the two instead of the arithmetic, it heavily penalizes unbalanced model performances.

$$\text{F1-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (2.8)$$

Figure 2.4 is an example of how a good accuracy does not always mean the model is performant. For this reason, complex classification problems use the F1 score instead of accuracy whenever classes are unbalanced, which is often the case in IE tasks. Other tools like the Receiver Operating Characteristic (ROC) curve use a variable threshold to modify the behavior of the classifier. ROC curves can be summarized to a single value by computing the Area Under the Curve (AUC), which is a robust metric against class imbalance.

Generalization

A common pitfall when training a supervised algorithm against a dataset is the overfitting of the model. An overfitted model performs great on the training data but poorly generalizes its results to new data. Figure 2.5 illustrate this behaviour on a regression task. Overfitting originates from the distribution of training samples

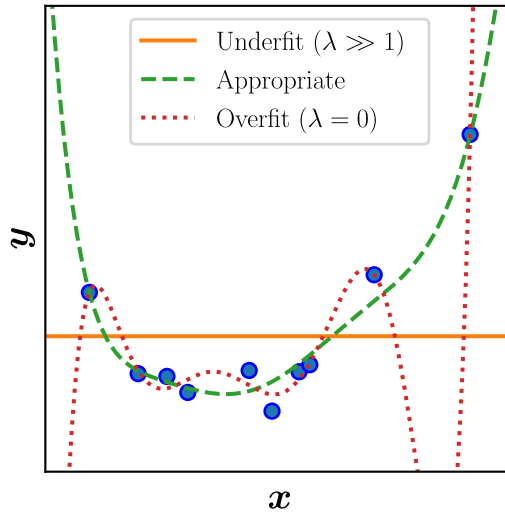


Figure 2.5: Fitting 7th degree polynomials on a set of points. Training is achieved by performing ridge regression which regularizes the model’s weights by penalizing their magnitude. The amount of regularization is controlled by a parameter λ . No regularization (Red dotted curve) results in an overfitted model that passes close to every point in the training data. However, the learned function does not match the actual structure of the data: it decreases sharply on the left side of the plot while the data points towards a steep increase. Too much regularization (Orange plain curve) constrains the model to a simple constant function. The best results are achieved with an intermediate amount (Green dashed curve) which forces the algorithm to learn a simple yet sufficient representation of the data.

$\mathcal{S}_{\text{train}}$ being different from the true (unknown) population distribution p^* . One can define the theoretical loss of a model $L(\boldsymbol{\theta}; p^*)$.

$$L(\boldsymbol{\theta}; p^*) = \mathbb{E}_{p^*(\mathbf{x}, \mathbf{y})} [\ell(f(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y})] \quad (2.9)$$

Then, the generalization of the model can be measured with the quantity $L(\boldsymbol{\theta}; p^*) - L(\boldsymbol{\theta}; \mathcal{S}_{\text{train}})$ which is sometimes called the generalization gap. Unfortunately, the actual distribution of data is either unknown or untractable. The generalization gap can be approximated by randomly partitioning the dataset into a train and a validation subset. Test data remain unseen from the model until training is complete, then the theoretical loss can be approached using the test data. To minimize variance due to the choice made during the dataset split, it is possible to split the dataset into k chunks of equal size. By training the model on $k - 1$ chunks and testing the remaining one for all possible chunks in a round-robin manner, one can better tune its model for generalization. This popular process is called cross-validation.

$$L(\boldsymbol{\theta}; \mathcal{S}_{\text{test}}) = \frac{1}{|\mathcal{S}_{\text{test}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{S}_{\text{test}}} \ell(f(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) \approx L(\boldsymbol{\theta}; p^*) \quad (2.10)$$

Metrics can also be evaluated on the test set to picture the expected performance of the model on previously unseen data once in production. This is the standard used in most competitions and benchmarks when comparing different ML models. To avoid any data leak which could result in unfair performance, the test set is often kept secret by organizers and the number of submissions is limited.

Regularization

Overfitting can be controlled by introducing regularization to the model. Regularization guides the learning process toward simpler representations. The idea of regularizing the model is connected to Occam’s razor which states that among multiple hypotheses that explain the observed data, the simplest should always be preferred. A common way to apply regularization is to penalize the magnitude of the weights. Optimal parameters obtained after training are then given by Equation 2.11 where $\|\boldsymbol{\theta}\|_2^2$ is the squared L2 norm of the weights and $\lambda \in \mathbb{R}^+$ is a hyperparameter controlling the amount of regularization imposed to the model. The effect of varying λ is shown in Figure 2.5.

$$\boldsymbol{\theta}^* = \arg \min L(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2 \quad (2.11)$$

Weight decay is not the only way to perform regularization, another popular technique involves stopping the learning process earlier than scheduled, based on the loss computed on the test set. Early stopping monitors test loss and stops the training process whenever it starts increasing. It works by mechanically limiting the generalization gap described previously.

Linear Algorithms

The foundations of supervised machine learning were laid down as early as the 50s, at a time electronic computers just happened to be invented and connectionist AI wasn’t the dominant approach. Often described as the first ML architecture, the perceptron [Rosenblatt, 1958] can perform binary classification by splitting the input space with a hyperplane (see Figure 2.3). Many subsequent models like the logistic and ridge regressions are closely linked to the perceptron. Because of their linear nature, they aren’t able to learn complex patterns between input and target. This very limitation has been the source of fierce debate between symbolic and connectionist AI researchers [Minsky and Papert, 1969].

Albeit linear models can only learn linear functions of the input, it is possible to circumvent this limitation by augmenting the input space \mathcal{X} with a function $\phi: \mathcal{X} \rightarrow \mathcal{X}'$. A linear model f operating on the augmented input \mathcal{X}' is able to learn non-linear patterns between the original input and the target iif. ϕ is non-linear as described in Equation 2.12. This process is called feature engineering (or feature

preprocessing) and has many applications such as performing basic ridge regression to fit polynomials like in Figure 2.5. In this case, $\phi(x) = [x, x^2, \dots, x^n]$ where n is the desired degree of polynomials.

$$\begin{aligned} f(\mathbf{x}'; \mathbf{w}, b) &= \mathbf{x}' \cdot \mathbf{w} + b \\ &= \phi(\mathbf{x}) \cdot \mathbf{w} + b \end{aligned} \quad (2.12)$$

Feature engineering is a powerful tool that enables tackling more complex tasks with simple models. However, it comes at the price of a larger input space size and increased computational runtime. An early approach was to constrain models to the following form where K is a kernel function and a_i are scalar parameters.

$$f(\mathbf{x}) = \sum_{i=1}^n a_i K(\mathbf{x}, \mathbf{x}_i) \quad (2.13)$$

If K is positive definite, Mercer's theorem provides an alternative representation of K , where $\phi: \mathcal{X} \rightarrow \mathcal{X}'$ is some function which can be interpreted as feature engineering performed on the input. Positive-definite kernel functions are a generalization of a scalar product over a possibly non-linear space.

$$K(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i) \quad (2.14)$$

For some kernel functions, the direct expression of $K(\mathbf{x}, \mathbf{x}_i)$ is much simpler than its alternative using ϕ . It means that one can train a model described in Equation 2.13 with complex feature engineering ϕ without paying the computational cost associated. This method is called the kernel trick. With the appropriate kernel function, training can even be performed on a feature space \mathcal{X}' of infinite dimension. The RBF kernel fits such a description and is widely used in SVM [Vapnik et al., 1996] classifiers. Models such as SVM provide a simple but capable representation of the data which is very efficient at inference time. But for reasons related to its training algorithm (convex optimization), it gets expensive to fit such a model when the size of the dataset increases. Moreover, the architecture of the model is fixed and cannot be modified except for the kernel function. Around the same period, another type of model, often called Multi Layer Perceptrons (MLPs) or more generally ANNs were developed.

2.1.2 Artificial Neural Networks

As shown previously, to increase the flexibility of a model, it is possible to perform feature extraction on the input. This allowed linear models to learn nonlinear relations between input and output. The expression of can be expanded as in Equation 2.12 where $\phi(\mathbf{x})$ is the new input with feature extraction applied. This structure is however limited by the fact ϕ is fixed before training and cannot be learned by the model itself. One approach to solve this is to provide learnable parameters $\boldsymbol{\theta}_\phi$ into ϕ to get

$$f(\mathbf{x}; \boldsymbol{\theta}) = \phi(\mathbf{x}; \boldsymbol{\theta}_\phi) \cdot \mathbf{w} + b \quad (2.15)$$

where $\boldsymbol{\theta} = (\mathbf{w}, b, \boldsymbol{\theta}_\phi)$ describes all parameters of f . And nothing prevents us from applying the same trick multiple times recursively by performing multiple feature extraction steps ϕ_1, \dots, ϕ_n each respectively parametrized by $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n$ to obtain an even more complex model where feature engineering functions are composed into each other. Without loss of generality, the outermost linear operation involving \mathbf{w} and b can also be replaced with a function ϕ .

$$\begin{aligned} f(\mathbf{x}; \boldsymbol{\theta}) &= \phi_{n-1}(\dots(\phi_1(\mathbf{x}; \boldsymbol{\theta}_1); \dots); \boldsymbol{\theta}_{n-1}) \cdot \mathbf{w} + b \\ &= \phi_n(\phi_{n-1}(\dots(\phi_1(\mathbf{x}; \boldsymbol{\theta}_1); \dots); \boldsymbol{\theta}_{n-1}); \boldsymbol{\theta}_n) \\ &= (\phi_n \circ \dots \circ \phi_1)(\mathbf{x}; \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n) \end{aligned} \quad (2.16)$$

If one restrict each ϕ_i to be a perceptron (see Figure 2.3), Equation 2.16 describes what is called a **MLP**. Each layer except the last composing the model is commonly referred to as a hidden layer, the last one being the output layer. The width D_l of the l^{th} layer $\phi_l: \mathbb{R}^{D_{l-1}} \rightarrow \mathbb{R}^{D_l}$ refers to the dimension of its output space, which is also the size of the input space of the next layer. Although models with a single hidden layer sufficiently wide can approximate any desired function [Hornik et al., 1989], depth has proven theoretically and experimentally [Montufar et al., 2014; Raghu et al., 2017] to produce better models. The key idea is that depth allows for the composition of relevant features learned by the model in the previous layers. The depth of ANNs used in the last decade is at the origin of the name of the field of study: *deep learning*.

Backpropagation

To train such models, we rely on gradient descent algorithms 2.4 to update each weight. However, it assumes the gradient of the loss with respect to each parameter $\nabla_{\boldsymbol{\theta}_i} L = \frac{\partial L}{\partial \boldsymbol{\theta}_i}$ can be computed. On simple, single chain MLP, it can be obtained using the chain rule for partial derivatives. For a n layers model f , we name \mathbf{x}_l the intermediate input of layer l such that $\mathbf{x}_{l+1} = \phi_l(\mathbf{x}_l; \boldsymbol{\theta}_l)$ is the activation of the layer. To simplify notations, let the loss be the output of the model $L = f(\mathbf{x}; \boldsymbol{\theta})$.

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_n} L &= \frac{\partial L}{\partial \boldsymbol{\theta}_n} \\ \nabla_{\boldsymbol{\theta}_{n-1}} L &= \frac{\partial L}{\partial \mathbf{x}_n} \frac{\partial \mathbf{x}_n}{\partial \boldsymbol{\theta}_{n-1}} \\ &\vdots \\ \nabla_{\boldsymbol{\theta}_i} L &= \frac{\partial L}{\partial \mathbf{x}_n} \frac{\partial \mathbf{x}_n}{\partial \mathbf{x}_{n-1}} \dots \frac{\partial \mathbf{x}_{i+2}}{\partial \mathbf{x}_{i+1}} \frac{\partial \mathbf{x}_{i+1}}{\partial \boldsymbol{\theta}_i} \end{aligned} \quad (2.17)$$

where $\frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l} = \frac{\partial \phi_l(\mathbf{x}_l)}{\partial \mathbf{x}_l} = \mathbf{J}_{\phi_l}(\mathbf{x}_l)$ is the Jacobian of the l^{th} layer. The computation for all weights can be performed efficiently by storing the activation of each layer

during the inference and using them to compute relevant partial derivatives in a backward manner. This specific procedure is called the *backpropagation* [Bryson et al., 1979; Rumelhart et al., 1988]. Although we demonstrated how it works for a simple MLP, it can be generalized to more complex architectures as long as the graph of computation can be described as a directed acyclic graph.

Activation Functions

If every feature extraction function ϕ_i is a linear function, the whole model collapses to a single function with very limited benefits. Using perceptrons as described in Figure 2.3 avoid this caveat by using Heaviside function which is non-linear. The architecture can be generalized by introducing a nonlinear activation function $\sigma: \mathbb{R} \rightarrow \mathbb{R}$ such that

$$\phi(\mathbf{x}; \boldsymbol{\theta}) = \sigma(\mathbf{x} \cdot \mathbf{w} + b) \quad (2.18)$$

For the resulting model to be trainable through gradient descent, it is important for the activation function to be differentiable almost everywhere. Early architectures used the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ which is a softened version of Heaviside. However, with models getting deeper and deeper, the sigmoid caused vanishing gradients which prevent future weight updates during training. The root cause of vanishing gradients is the behavior of the sigmoid for inputs of large positive and negative values. The sigmoid saturates at +1 and 0 while the gradient with respect to the input is 0. When applying backpropagation 2.17, the update signal is nullified because $\mathbf{J}_{\phi_l}(\mathbf{x}_l) = 0$. When designing deep models, it is important for the term $\prod_l \mathbf{J}_{\phi_l}(\mathbf{x}_l)$ to stay close to 1. Otherwise, the gradient could vanish (resulting in the weights not updating) or explode (resulting in the weights updating erratically, without converging).

To solve this issue, many other activation functions have been proposed. The most common modern activation is the Rectified Linear Unit (ReLU) [Glorot et al., 2010] defined as $\text{ReLU}(x) = \max(x, 0)$. Although negative inputs are muted, it does not suffer from gradient vanishing for large positive inputs. It has proven to be sufficient to train deeper models but many other activations have been proposed that improve its behavior for negative inputs [Maas et al., 2013; Ramachandran et al., 2017; Hendrycks and Gimpel, 2020]. But ReLU is still very popular to this date because of its simplicity and efficiency and is widely used for hidden layers.

When performing classification tasks, instead of simply feeding the output logits of the model into the NLL loss, we prefer to normalize the logits such that they can be interpreted as a probability or a confidence score. We use the softmax [Bridle, 1990] function which helps the model focus on the label with maximum confidence while still being differentiable. It also has the nice property to be computable in a numerically stable manner when combined with a NLL without risking overflowing the exponentials.

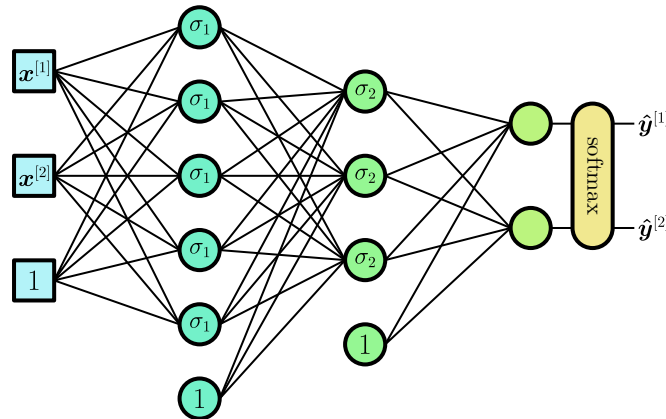


Figure 2.6: Architecture of a MLP with 2 hidden layers and a binary classification head. Bold links between nodes denote a trainable parameter of the model with a total of 41 parameters. Typical implementations of MLPs use matrices of parameters such that a layer is a single matrix multiplication followed by the activation function σ_i .

$$\text{softmax}(\mathbf{x})^{[i]} = \frac{e^{\mathbf{x}^{[i]}}}{\sum_{j=1}^c e^{\mathbf{x}^{[j]}}} \quad (2.19)$$

Packing it all together results in a complete MLP architecture as shown in Figure 2.6.

From Neural Networks to Deep Learning

Although most of the tools were discovered more than 20 years ago, ANNs only became competitive against other models recently in the 2010s. The first successful applications of deep learning methods were developed in the late 80s. Convolutional Neural Networks (CNNs) [Lecun et al., 1989] were applied to handwritten digit recognition which later resulted in the MNIST [Lecun et al., 1998] dataset, the “hello world” of supervised learning. Recurrent Neural Networks (RNNs) [Bengio et al., 1994], another architecture of ANN was later proposed to manipulate sequences of data. They were successfully used in handwritten [Graves and Schmidhuber, 2008] and speech recognition [Graves et al., 2013]. Tackling more difficult tasks requires bigger models with complex architectures, more data to train and thus more computation power.

Multiple factors made the revolution of deep learning possible. First, the early 2000s saw the rise of the capabilities of computers and the democratization of Graphics Processing Units (GPUs), speeding up the training by multiple orders of magnitude. Because working with GPUs is non-trivial, it led to the development of machine learning libraries and frameworks built for performance [Bergstra et al., 2011; Abadi et al., 2015; Paszke et al., 2019]. Secondly, large datasets with competitions attached were released. The most notable is ImageNet [Deng et al., 2009]

which contains more than 3 million images and more than 5000 classes. Since then, many fields [Redmon et al., 2016; Silver et al., 2017; Ramesh et al., 2021; Jumper et al., 2021] have been impacted by the deep learning revolution. In the following section, we will further develop deep learning architectures and applications in Business Document (BD) understanding.

2.2 Deep Learning

Throughout the previous section, we've introduced all necessary building blocks to train ANNs. A MLP can approximate any mapping from inputs to outputs given enough examples and parameters. However, in practice, growing the size of a model is rarely sufficient to solve a complex task due to overfitting and gradients vanishing. Instead, multiple architectures have been invented to accommodate specific properties of the processed data. Those architectures often take the shape of a specific layer acting as a building block of ANNs. In modern deep learning, the layers of MLPs are referred to as dense or linear layers instead. We will further review layers and techniques related to BD understanding and their applications.

2.2.1 Convolutional Neural Networks

Images are commonly manipulated by computers as they can be described by their pixels. They can be represented as dense matrix $\mathbf{x} \in \mathbb{R}^{W \times H}$ where W is the width and H is the height of the image in pixels. Because this definition only works for black and white images, a third dimension C is usually appended for the color channels, it only contains 3 values for red, green and blue components (RGB). By this definition, a color image is a 3-dimensional object $x \in \mathbb{R}^{W \times H \times C}$. Classical MLPs are not well suited to manipulate such objects for several reasons. First, not all images have the same dimensions, which is not compatible with how the layers of a MLP work. Images would need to be resized to the same shape or padded to the largest image in order to use the same model for every example. Second, and this is the main issue, is the huge amount of parameters needed to map the input to the first hidden representation of the model. Given a hidden layer of size D , it would require $(W \times H \times C) \times D$ parameters. It represents $8D$ million parameters for an image with a 4K UHD definition.

Convolution

CNNs [Lecun et al., 1989] have been proposed to overcome those issues by replacing the matrix multiplication with a 2D convolution operation. As explained in Figure 2.7, the idea behind convolution is to divide the input image into possibly overlapping patches and perform a dot product with a fixed patch called a filter. A CNN uses convolutions with multiple filters which are learned the same way MLPs' weight matrices are learned: through gradient descent. It solves most problems of MLPs when manipulating images as convolutions can be performed on any input

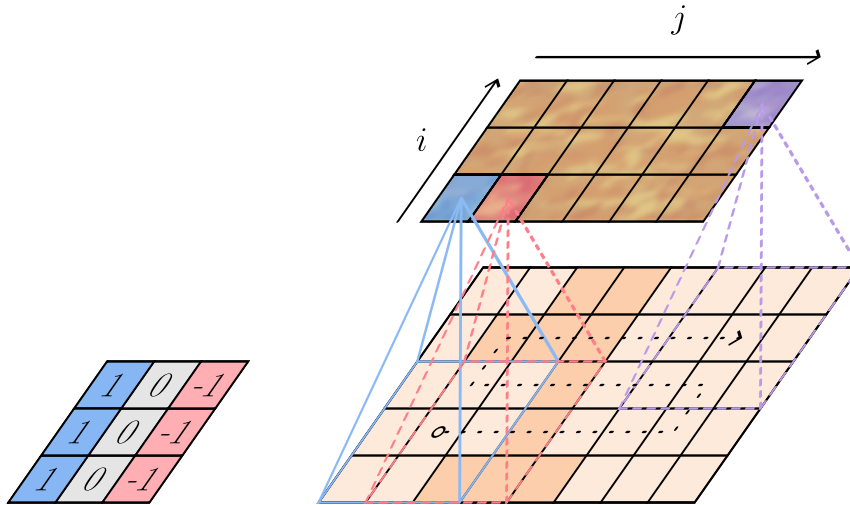


Figure 2.7: Illustration of a convolution operation between a 3×3 filter (Left) and a 5×8 image (Bottom right). Convolution can be represented as overlaying the filter over the image and computing the dot product between the filter and the part of the image underneath. By repeating this operation for every possible position where the filter fits by following the black dotted line, an output image (Top right) is computed where each output pixel is the result of one dot product. In this very example, the filter detects vertical edges in the input image by producing large outputs around those edges. Figure freely inspired of Géron [2017, Figures 14.3-14.4].

image size. In practice the change in size of the output can be managed through a combination of tricks including pooling and padding we will detail later. Moreover, the number of trainable parameters of a CNN does not depend on the input size like a MLP does but on the number F of filters and their width $W_f \ll W$ and height $H_f \ll H$ for a total of $F \times W_f \times H_f$ parameters. It results in a drastically smaller amount of parameters. Finally, because filters are constant during the convolution, CNNs are translationally invariant by design.

Convolution can also be applied to 1D vectors. Given a vector $\mathbf{x} \in \mathbb{R}^N$ and a filter $\mathbf{F} \in \mathbb{R}^L$, their convolution is defined as

$$(\mathbf{F} \otimes \mathbf{x})^{[i]} = \sum_{u=1}^L \mathbf{F}^{[u]} \mathbf{x}^{[i+u-1]} \quad (2.20)$$

which can be useful when working with 1D sequences. This definition is slightly different from the canonical mathematical definition where input is flipped along the axis. It would rather be referred to as *cross-correlation*, but the difference does not matter to our use case because the filter \mathbf{F} is learned.

Because the convolution is a linear operator, it can also be expressed as a matrix-vector multiplication. Equation 2.21 implies a convolutional layer is equivalent to a dense layer with sparse weights and redundancy. In this toy example, the convolution uses 2 parameters instead of 16 for the dense network.

$$\begin{aligned}
\mathbf{y} &= \mathbf{F} \otimes \mathbf{x} \\
&= \begin{pmatrix} \mathbf{F}^{[1]} \\ \mathbf{F}^{[2]} \end{pmatrix} \otimes \begin{pmatrix} \mathbf{x}^{[1]} \\ \mathbf{x}^{[2]} \\ \mathbf{x}^{[3]} \\ \mathbf{x}^{[4]} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{F}^{[1]}\mathbf{x}^{[1]} + \mathbf{F}^{[2]}\mathbf{x}^{[2]} \\ \mathbf{F}^{[1]}\mathbf{x}^{[2]} + \mathbf{F}^{[2]}\mathbf{x}^{[3]} \\ \mathbf{F}^{[1]}\mathbf{x}^{[3]} + \mathbf{F}^{[2]}\mathbf{x}^{[4]} \end{pmatrix} \tag{2.21} \\
&= \begin{pmatrix} \mathbf{F}^{[1]} & \mathbf{F}^{[2]} & 0 & 0 \\ 0 & \mathbf{F}^{[1]} & \mathbf{F}^{[2]} & 0 \\ 0 & 0 & \mathbf{F}^{[1]} & \mathbf{F}^{[2]} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^{[1]} \\ \mathbf{x}^{[2]} \\ \mathbf{x}^{[3]} \\ \mathbf{x}^{[4]} \end{pmatrix}
\end{aligned}$$

Convolutions can generalize to any number of dimensions. In 2D, Equation 2.20 becomes

$$(\mathbf{F} \otimes \mathbf{x})^{[i,j]} = \sum_{u=1}^{H_F} \sum_{v=1}^{W_F} \mathbf{F}^{[u,v]} \mathbf{x}^{[i+u-1, j+v-1]} \tag{2.22}$$

As said before, images usually include a 3rd dimension for the color channels. A convolutional layer takes an input image \mathbf{x} of E channels and produces an output image of different sizes with F channels. It uses $E \times F$ filters $\mathbf{F}_{i,j}$ of identical height H_F and width W_F alongside F multiple biases \mathbf{b}_j . For the same reasons brought earlier about dense layers needing an activation function, a convolutional layer also includes an activation function σ . The j^{th} channel of the output image is given by

$$\mathbf{y}_j = \sigma(\mathbf{b}_j + \sum_{i=1}^E \mathbf{F}_{i,j} \otimes \mathbf{x}_i) \tag{2.23}$$

The output of the convolutional layer is also called a *feature map* because the convolution acts as a feature detection tool. The bigger the filter size, the bigger the objects it can detect in a single convolution. It is common practice to use rather small filter sizes up to a dozen wide and tall and stack multiple convolutional layers. Subsequent layers can learn to detect large objects by combining small features in the intermediate feature maps.

CNN architectures combine convolutional layers with other layers to further reduce parameter number and tendency to overfitting. Pooling layers reduce the size of a feature map without introducing any trainable parameters by selecting some values and throwing others. Max pooling uses a sliding window of size $H_F \times W_F$ which outputs the maximum value within that window. Feature map size can be further reduced by introducing vertical or horizontal stride which effectively skips some positions of the sliding window. Convolutional and max-pooling layers are the main building blocks of the LeNet [Lecun et al., 1998] which performs handwritten digit recognition with unprecedented accuracy for the time. VGG architecture [Harley

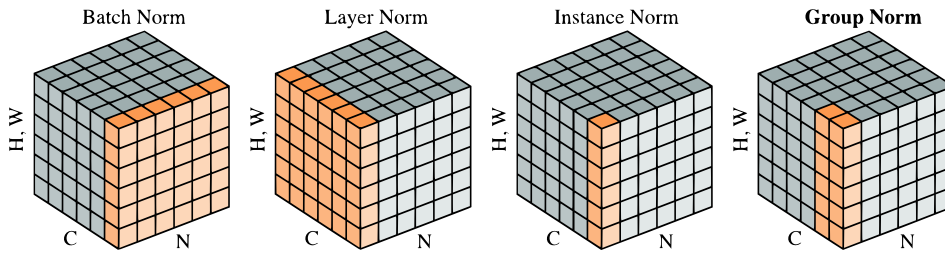


Figure 2.8: Different normalizations being performed on a tensor. Values in shades of orange are standardized to the same mean and variance which is computed by aggregating those same values. H and W stand for the height and width of the feature map, while C is the channel dimension and N is the batch size. Figure freely inspired of Wu and He [2018].

et al., 2015] later revolutionized ImageNet [Deng et al., 2009] by drastically improving performance over all other image classification methods. CNNs models were also applied to document image classification on RVL-CDIP [Harley et al., 2015], relation extraction [Davis et al., 2019] and table segmentation [Zhang et al., 2020].

Normalization and Regularization

Deeper models suffer from vanishing and exploding gradients which makes the training chaotic and unpredictable. Researchers have come up with multiple solutions to help smooth the propagated gradient. Introducing normalization layers at several steps in the model helps to standardize the mean and variance of the intermediate hidden representations. Batch normalization [Ioffe and Szegedy, 2015] is popular for CNNs, it ensures inputs across a sampled batch follow a standard distribution. Most normalization layers [Ba et al., 2016; Wu and He, 2018; Zhang and Sennrich, 2019] use the relations described in Equation 2.24 to standardize the input \mathbf{x} where μ and σ^2 are respectively the mean and the squared variance of the inputs over the normalized dimension. Small weights \mathbf{w} and biases \mathbf{b} are introduced after normalization and are learned with other parameters. It can be inappropriate when inputs are processed in small batches or over multiple devices, that's why normalization can be performed over other dimensions. Figure 2.8 shows several possible choices of dimension to normalize over.

$$\mathbf{y} = \mathbf{w} \odot \frac{\mathbf{x} - \mu}{\sigma} + \mathbf{b} \quad (2.24)$$

In addition to normalization, dropout layers [Srivastava et al., 2014] can be used to further reduce the risk of overfitting. By randomly dropping neurons at train time, dropout forces the model to be redundant and resilient to noise. Figure 2.9 shows how it might affect the output of a model by turning off some neurons as if they did not exist. However, when testing or during inference, the dropout layer becomes inactive and lets all neurons be active. Dropout has proven to be a very effective tool to precisely control overfitting for large models.

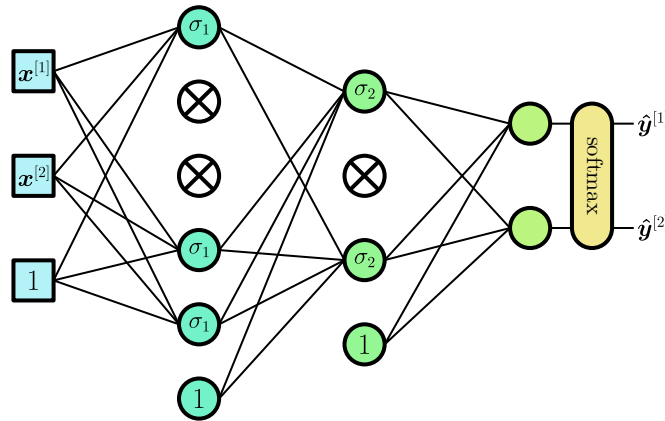


Figure 2.9: Same architecture as Figure 2.6 but with dropout [Srivastava et al., 2014] applied to each hidden layer. Neurons crossed out have been dropped out and are not used by the model during this training step. With a probability $P = \frac{1}{3}$ of dropping each neuron, the architecture of the model is completely different. And because the dropped-out neurons change at each training step, the model must be resilient to this perturbation.

To solve vanishing gradients occurring in deeper models, He et al. [2016] proposed a simple trick to efficiently guide the gradient through many layers. Their solution is to introduce shortcuts for the gradient around a block composed of possibly multiple layers. Let $f_{\theta}: \mathcal{X} \times \mathcal{W} \rightarrow \mathcal{X}$ be the parametrized block mapping from an intermediate hidden space to itself. A residual layer g_{θ} around f_{θ} is defined as

$$g(\mathbf{x}; \theta) = f(\mathbf{x}; \theta) + \mathbf{x} \quad (2.25)$$

ResNet [He et al., 2016] architecture was able to train models with more than 100 layers deep and is still a solid baseline for most image comprehension tasks. The same strategy was used in U-Net models [Ronneberger et al., 2015] for image segmentation tasks. It is able to produce pixel-level classification or regression. It later led to novel approaches for document IE proposed by Katti et al. [2018] and Yang et al. [2017] we discuss more thoroughly in subsection 2.2.3.

Data Efficiency

Because building a large dataset to solve a task is costly, many strategies exist to increase the data efficiency of trained models. One approach is to artificially increase the dataset size by applying random transformations to the input during the training. By carefully choosing transformations that do not change the target associated with the input, *data augmentation* teaches data invariants to the model. Common transformations for image-like inputs rely on affine transformations of the image: rotations, zooms, flips, distortion, etc. A variety of advanced transformations have proven to significantly improve performances on most tasks and datasets [Yang et al., 2022]. Most complex transformations involve generative models [Goodfellow

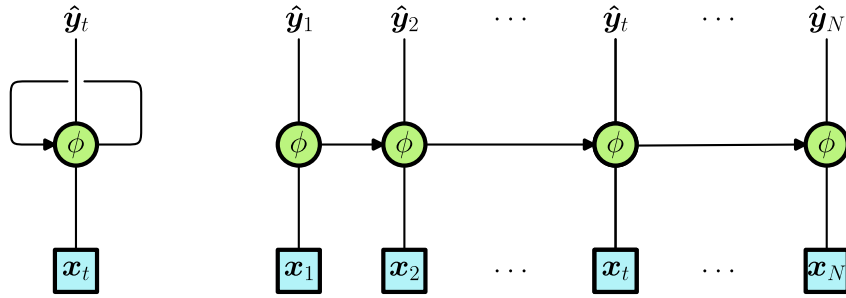


Figure 2.10: The input of the network \mathbf{x} is a sequence $(\mathbf{x}_t)_{1 \leq t \leq N}$. The network can produce a sequence of outputs $(\hat{\mathbf{y}}_t)_{1 \leq t \leq N}$ aligned with the input. Some output of the well ϕ at step t is fed to the next cell at step $t + 1$. The compact version (Left) showcases the recurrence in the model whereas the unrolled version (Right) better represents the sequence length dimension and the multiple copies of the recurrent cell. Figure freely inspired of Olah [2015].

et al., 2014; Choi et al., 2020] or mixing together multiple examples and interpolating in between [Zhang et al., 2018].

Another method to improve data-efficiency of a model is to first train on a large, somewhat related dataset. And then transfer the learned knowledge to a smaller downstream dataset. Transfer learning became relevant for image tasks thanks to ImageNet [Deng et al., 2009] which still provides a large high-quality dataset for general-purpose images [Kolesnikov et al., 2020]. Generally, the closer upstream and downstream tasks are the greater the improvement on the downstream dataset. For Document Understanding (DU) related tasks, IIT-CDIP Lewis et al. [2006] and RVL-CDIP [Harley et al., 2015] are good candidates for pre-training a model. The idea to re-use large datasets as a pre-training step before *fine-tuning* on the desired task enabled larger and more efficient models in NLP and DU.

2.2.2 Recurrent Neural Networks

Sequential data can be represented as an array of inputs $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_t, \dots, \mathbf{x}_N)$. Although 1D CNNs can be used to manipulate sequential data, they struggle to propagate information across a long range. This is mainly due to the locality of computation imposed by the size of the sliding window. Instead of CNNs, the standard approach was for a long time to use RNNs. RNNs use a recurrent architecture composed of cells where the output of cell t is directly linked to the input of cell $t + 1$. RNNs can be unrolled as in Figure 2.10 to better envision their ability to manipulate sequential data.

Recurrent Cells

The main computation unit in a RNN is a recurrent cell $\phi: \mathcal{X} \times \mathcal{Y} \times W \rightarrow \mathcal{Y}$. Let $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_t, \dots, \hat{\mathbf{y}}_N)$ be the outputs corresponding to \mathbf{x} , then the following relation holds

$$\hat{\mathbf{y}}_t = \phi(\mathbf{x}_t, \hat{\mathbf{y}}_{t-1}; \boldsymbol{\theta}) \quad (2.26)$$

where $\boldsymbol{\theta} \in \mathcal{W}$ are learned parameters of the recurrent cell. Because the cell is identical for each time step t , a RNN can manipulate long sequences without needing large amounts of parameters. The recurrent connection between cells allows the model to propagate information from the first input towards future time steps without limitation in theory. Simple recurrent cells take the shape of a single dense layer with an activation. Equation 2.26 becomes

$$\hat{\mathbf{y}}_t = \sigma([\mathbf{x}_t, \hat{\mathbf{y}}_{t-1}] \cdot \mathbf{w} + b) \quad (2.27)$$

where $[\mathbf{x}_t, \hat{\mathbf{y}}_{t-1}]$ is the concatenation of \mathbf{x}_t and $\hat{\mathbf{y}}_{t-1}$ along their feature dimension. Historically RNNs used the sigmoid or the hyperbolic tangent as their activation function but were quickly replaced by more efficient cells.

To update the cell's parameter, backpropagation must be performed through the sequence dimension, often referred to as *Backpropagation Through Time* [Werbos, 1990]. The effective depth of the model grows with the length of the sequence, which makes RNNs extremely sensitive to vanishing and exploding gradients. If the exploding gradient can be tempered by clipping the propagated gradient at each time step, vanishing gradients remain an issue that slowly mutes the learning of interactions between time steps. Hochreiter and Schmidhuber [1997] proposed an alternative recurrent cell called Long Short Term Memory (LSTM). It introduces several intermediate results called gates that carry information between time steps. First, the input gate \mathbf{i}_t controls which part of the input gets read into the memory \mathbf{m}_t . The forget gate \mathbf{f}_t determines which part of the memory from the previous time step should be kept. The output gate \mathbf{o}_t filters the current memory to create the output \mathbf{y}_t . A LSTM cell is driven by the following equations

$$\mathbf{i}_t = \sigma(\mathbf{x}_t \cdot \mathbf{w}_{xi} + \mathbf{y}_{t-1} \cdot \mathbf{w}_{yi} + \mathbf{b}_i) \quad (2.28)$$

$$\mathbf{f}_t = \sigma(\mathbf{x}_t \cdot \mathbf{w}_{xf} + \mathbf{y}_{t-1} \cdot \mathbf{w}_{yf} + \mathbf{b}_f) \quad (2.29)$$

$$\mathbf{o}_t = \sigma(\mathbf{x}_t \cdot \mathbf{w}_{xo} + \mathbf{y}_{t-1} \cdot \mathbf{w}_{yo} + \mathbf{b}_o) \quad (2.30)$$

$$\tilde{\mathbf{m}}_t = \tanh(\mathbf{x}_t \cdot \mathbf{w}_{xm} + \mathbf{y}_{t-1} \cdot \mathbf{w}_{ym} + \mathbf{b}_m) \quad (2.31)$$

where $\tilde{\mathbf{m}}_t$ is an intermediate candidate for the memory at step t . The outputs of the cell are

$$\mathbf{m}_t = \mathbf{f}_t \odot \mathbf{m}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{m}}_t \quad (2.32)$$

$$\hat{\mathbf{y}}_t = \mathbf{o}_t \odot \mathbf{m}_t \quad (2.33)$$

If $\mathbf{f}_t = 1$ and $\mathbf{i}_t = 0$, the network is able to propagate information over a long range thanks to its memory. LSTMs can be very sensitive to initial parameters, as some initializations block the signal coming from previous time steps \mathbf{m}_{t-1} . For

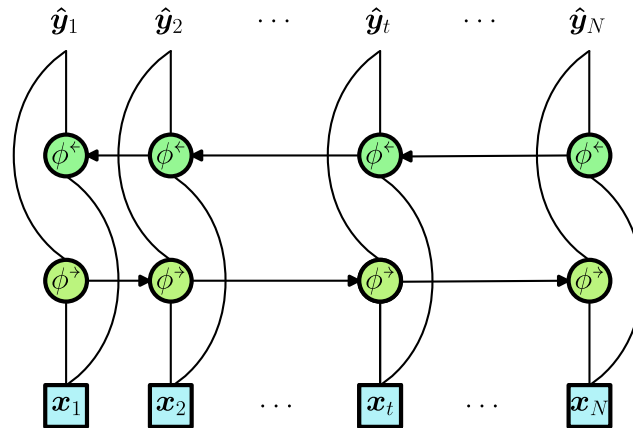


Figure 2.11: A bidirectional RNN is composed of two independent RNNs: the forward and backward networks. Like in Figure 2.10, the forward network ϕ^{\rightarrow} treats the input sequence from the beginning to the end. In contrast, the backward network ϕ^{\leftarrow} processes the sequence in reverse order. By combining both forward and backward representations, a bidirectional RNN is able to capture both past and future context into its prediction.

those reasons, it is recommended [Jozefowicz et al., 2015] to initialize the bias of the forget gate b_f to a large value such that gradient can be effectively backpropagated through time.

Since LSTM [Hochreiter and Schmidhuber, 1997] architecture was first released, many modifications have been proposed. The most notable is probably the Gated Recurrent Unit (GRU) cell [Cho et al., 2014] which is an equally performant simplification of the LSTM cell. Gers and Schmidhuber [2000] showed that peephole connections between the memory and the gates improve the abilities of the network on some specific tasks.

RNNs are able to make predictions at step t based on all previous time steps. This behavior might be relevant for some applications where causality is involved but is a limitation in the case of textual information. To leverage both past and future information, Schuster and Paliwal [1997] proposed to use two RNNs. The first one goes in the forward direction while the second one starts from the end and goes backward as pictured in Figure 2.11. That specific combination of two networks is called a bidirectional RNN. When applied to a LSTM cell, the resulting layer is commonly referred to as a Bidirectional Long Short Term Memory (BiLSTM).

Text Representation

On the one hand, ANNs are great at scrambling numbers and learning patterns and correlations in samples. On the other hand, text might seem incompatible with a numeric representation at first glance. To feed textual information to any model, one must first use an appropriate representation. To do so, text strings are split into atomic units called *tokens* and each token is then converted into its vector

representation also called its *embedding*. The sequence of tokens can then be fed to a RNN which can infer the meaning of the whole sequence and perform computation on it.

The most natural way to tokenize a sequence is to split by words, each time a space is encountered. This is great for natural text with a limited number of different words. However, it fails at giving a representation to unknown words such as proper nouns, brands or in the context of BDs: item IDs, phone numbers and prices. Indeed, when using word-level tokenization, the model can only represent words included in the *vocabulary*, which is built based on the training data. Words that are Out Of Vocabulary (OOV) are then assigned to a unique representation for all unknown words, often referred to as [UNK].

An alternative tokenization strategy is to use characters instead of words. By carefully choosing the vocabulary, it is impossible to meet OOV tokens. Such vocabulary could be based on the ASCII table for Latin-only text, or the whole Unicode which includes all human characters and symbols. On the positive side, character tokenization can represent any sequence with a bounded vocabulary. It is also more resilient to Optical Character Recognition (OCR) errors than word tokenization because a single character error does not completely change the overall word representation. However, character tokenization produces longer sequences and must use contextual information to derive any semantics in the sentence.

An early approach to avoid OOV while still having semantic tokens was to use n -grams [Robertson and Willett, 1998]: sequences of consecutive n characters. It effectively allows to control finely the vocabulary size but fails at giving an accurate representation of rare sequences. More recently, Sennrich et al. [2016] proposed an intermediate tokenization strategy between words and characters with variable-length tokens. It relies on a compression algorithm called Byte Pair Encoding (BPE) [Gage, 1994]. It constructs a vocabulary starting with individual characters by iteratively merging the most frequent pair of tokens into a new one. By applying the same merging operation until the vocabulary reaches the desired size, common sequences of characters are directly in the vocabulary as a single token. If the noun `trust` might probably be a token, the adjective `trustful` might not because of its rarer use. Instead, it would be tokenized into the radical `trust` and the suffix `#ful` which are both common. However, because BPE tokenization is driven by statistics, tokens might not reflect any actual morpheme with actual semantics. Other tokenization schemes in between characters and words exist, such as WordPiece [Wu et al., 2016], Unigram [Kudo, 2018] and SentencePiece [Kudo and Richardson, 2018] with little variations over BPE. Figure 2.12 summarizes word, character and sub-word level embeddings.

Once the vocabulary is fixed, each token is given a unique index $t_i \in \llbracket 1, V \rrbracket$ where V is the size of the vocabulary. A sentence is then represented as a sequence of N indices $\{t_1, \dots, t_N\}$. One possibility is to use one-hot encoding token representation: each token t_i is associated with a vector of size V filled with zeros except for a 1 at position t_i . However, this is highly inefficient for large vocabularies, instead, it is preferred to use embeddings of dimension $D \ll V$. An embedding layer then takes

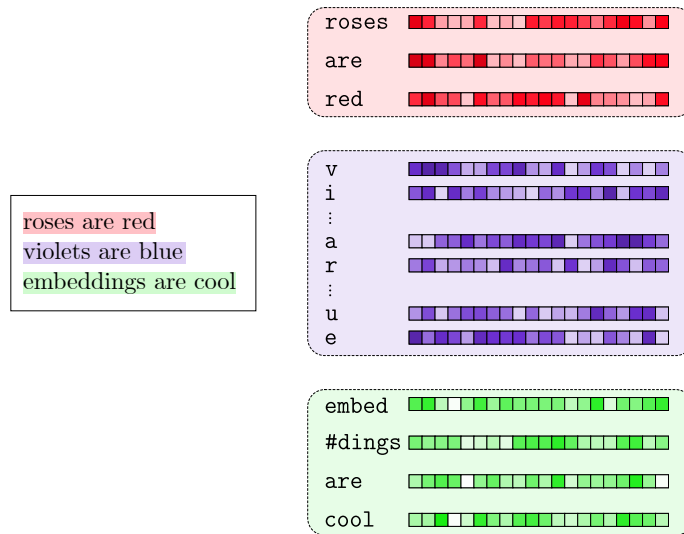


Figure 2.12: Three short sentences (Left) being tokenized and embedded. The first sentence (Top) is tokenized into words while the second (Middle) is tokenized into characters and the third (Bottom) uses sub-word tokenization techniques. Colored squares represent embeddings of each corresponding token. At the cost of a slightly longer tokenized sequence, sub-word tokenization can tokenize unknown words like `embeddings` into known pieces (`embed` and `#dings`). The hash symbol `#` at the beginning of a token denotes the continuation of a word.

the shape of a function $\phi_{\text{emb}}: \mathbb{R} \rightarrow \mathbb{R}^D$ and can be implemented as a lookup table into a matrix.

In a perfect world, embeddings capture the semantics of tokens and attribute similar vector representations to similar tokens. This would allow the downstream layer to reason about semantic relationships and generalize to unseen tokens. This especially applies to word and subword tokens which bear a meaning by themselves. Various attempts have been made to compute useful token embeddings, by dimensionality reduction [Deerwester et al., 1990] of an occurrence matrix, or as a byproduct of a Language Modeling (LM) training [Bengio et al., 2003]. Collobert and Weston [2008] first proposed to train embeddings as a separate task: word embeddings were computed once and used for multiple downstream tasks.

Mikolov et al. [2013b] have drawn attention of the whole NLP community by releasing Word2Vec [Mikolov et al., 2013b,a], a successful to train word embeddings from unlabeled text. It assumes the semantics of a word is completely defined by the context in which it occurs, and thus that two words that could be replaced one by the other in every possible context have the same meaning. They proposed two strategies called Continuous Bag Of Words (CBOW) and Skip-gram, both involving a simple MLP. Given some context words in a sliding window around a target token t_i , CBOW trains the model to predict which token is t_i . Skip-gram on the contrary predicts which tokens are in the context window around t_i given which

token is t_i . Both strategies are treated as classification tasks with as many classes as tokens in the vocabulary. The desired embeddings are the weights of the first dense layer of the model which act as a projection of the input space of size V onto the embedding space of size D . By construction, Word2Vec [Mikolov et al., 2013a] provides similar embeddings to tokens occurring in the same context. Pennington et al. [2014] later proposed GloVe embeddings which combine techniques from global occurrence matrix factorization and local context windows such as Word2Vec.

Although word embeddings like Word2Vec and GloVe capture the semantics of a word by its context, a word can only have a single embedding associated. For example the word “play” can be used in the context of a sports game or a theater stage, its meaning is contextual. Hence, because static embeddings always provide the same representation, they cannot appropriately adapt to the surrounding context. To remedy this issue, Peters et al. [2018] learn a contextualized representation of tokens. They introduce ELMo, a multilayer BiLSTM network pre-trained on a large textual corpus to predict words based on the context. The pre-trained model is then re-used as is, in the same fashion as transfer learning was performed on deep CNNs. Contextual embeddings [Peters et al., 2018; McCann et al., 2017] showed how self-supervised training could value a large corpus of unlabeled data. They also greatly improved performances on downstream tasks and enabled tackling problems with less labeled data.

Encoders

RNNs can process textual input as a sequence of tokens $(t_i)_{1 \leq i \leq N}$ and produce a prediction for each token $(\hat{y}_i)_{1 \leq i \leq N}$. In order to use a similar network for sequence classification, it is possible to use the last output of the model \hat{y}_N which has received information from the whole sequence thanks to recurrent connections. In bidirectional RNNs, the same trick could be performed by aggregating the last output of both forward and backward networks. That architecture where the number of outputs is either 1 or N is commonly referred to as an *encoder*. They allow to tackle text classification, Named Entity Recognition (NER), part of speech tagging but fail at text generation, summarization and translation.

For tasks involving token classification, multiple improvements over a barebone BiLSTM have been proposed. Using CRF [McCallum, 2012] on top of BiLSTMs, Lample et al. [2016] improve the model’s performance as the predictions are jointly determined by the CRF layer. Sometimes, an occurrence of a label spans multiple tokens. The model must correctly classify each token, which is noticeably harder at the boundaries of the occurrence. To improve the quality of predictions for those occurrences, several tagging schemes have been proposed. BIO [Ramshaw and Marcus, 1999; Ratnoff and Roth, 2009] is the simplest which stands for Begin, Inside, Outside. According to this scheme, each `label` is derived into `B-label` and `I-label`. The first token of an occurrence is labeled `B-label` while all following tokens of the same occurrence are labeled `I-label`. This scheme also helps when dealing with multiple occurrences touching each other. `0` is kept for all other tokens without

any specific label associated. It can be extended with `E-label` and `S-label` which respectively stand for the end of an occurrence and a single token occurrence. The quality of this kind of tagging scheme resides in its expressivity and the ability to parse them linearly in a single pass.

[Palm et al. \[2017\]](#) and [Sage et al. \[2019\]](#) proposed to use `BiLSTM` encoders to perform `IE` in invoices, targeting both header and table fields. They represent a `BD` as a sequence of words detected by an `OCR` and associate each relevant information to a unique label. To cope with the limitation of `RNNs` to represent words laid on a page, they introduce 2D positional embeddings, providing information to the model where each word is located. The word order imposed by the `RNN` architecture is usually chosen to be the closest possible to human read order. To improve performance, they also propose specific feature extraction in addition to word embedding to better detect dates, numbers and other static formats with regular expressions. Sequence encoders surpassed previous models thanks to their ability to model the whole sequence at once which enables the propagation of information over a long range.

Decoders

Whenever the output necessitates a variable size or is not necessarily aligned with the input, [Sutskever et al. \[2014\]](#) proposed to use *decoder* networks. Decoders are autoregressive networks that predict the next token \hat{z}_m of a sequence based on the previously predicted tokens $(\hat{z}_j)_{1 \leq j < m}$. During inference, decoders iteratively generate the following token which is then appended to the input sequence. By repeating multiple times this procedure until an end-of-sequence token [`EOS`] is generated, decoders are able to generate variable-length text. In association with an encoder, they form an encoder-decoder architecture, as illustrated in [Figure 2.13](#).

Originally used for sentence translation [[Cho et al., 2014](#)], encoder-decoder networks displayed their ability to generate long sequences. This architecture allows the model `NER` and `IE` tasks as a Question Answering (`QA`) [[Gardner et al., 2019](#)] instead of token classification. It alleviates the data labeling for two reasons: key information does not need to be labeled at a token level and it does not need to be exactly represented in the text. To put it another way, instead of searching for where exactly the date “`january 1st, 2023`” is located in the document, one can simply put the label “`2023-01-01`” in ISO format. The model can learn how to find the date and how to parse it at the same time in an end-to-end way. [Sage et al. \[2020\]](#) and [Aggarwal et al. \[2020\]](#) demonstrated the applications of encoder-decoders in conjunction with an attention layer for `IE` in documents.

Attention

Attention was first proposed by [Bahdanau et al. \[2016\]](#) as a solution to improve communication between encoder and decoder networks. An attention layer between an encoder and its decoder provides the decoder direct short-circuiting access to the

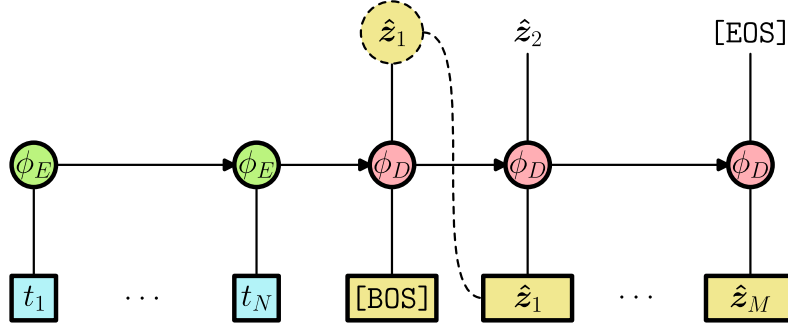


Figure 2.13: An encoder-decoder architecture with RNNs. The encoder (Left, in green) processes a context sequence $(t_i)_{1 \leq i \leq N}$ and produces its representation. The decoder (Right, in red) receives the representation of the context sequence in its recurrent input and starts its prediction with a special begin-of-sequence $[BOS]$ token. At each inference step j , the previous decoder's output \hat{z}_{j-1} is appended to its input for the next inference. Multiple strategies exist to select which output token is selected, given the model's probabilities distribution Graves [2012].

output sequence of the encoder $(\hat{\mathbf{y}}_i)_{1 \leq i \leq N}$. They modify the decoder recurrent cell ϕ_D by providing it an additional input \mathbf{a}_j as follows

$$\hat{\mathbf{z}}_{j+1}, \mathbf{m}_{j+1} = \phi_D(\hat{\mathbf{z}}_j, \mathbf{m}_j, \mathbf{a}_j) \quad (2.34)$$

where \mathbf{m}_j is the recurrent memory from the previous cell. \mathbf{a}_j is the attention paid by the decoder to the encoder at step j , which is a weighted mean of the encoder outputs:

$$\mathbf{a}_j = \sum_{i=1}^N \alpha_{ij} \hat{\mathbf{y}}_i \quad (2.35)$$

$$\alpha_{ij} = \frac{\exp(\text{sim}(\hat{\mathbf{y}}_i, \hat{\mathbf{z}}_j))}{\sum_{k=1}^N \exp(\text{sim}(\hat{\mathbf{y}}_k, \hat{\mathbf{z}}_j))} \quad (2.36)$$

where sim is a similarity function between two vectors of dimension D . Attention is further explained in Figure 2.14. First introduced for machine translation, attention greatly improved performance on long sequences over previous encoder-decoder models. It can be explained intuitively because the encoder is not required anymore to encode the whole context sequence into a fixed-size recurrent memory given to the decoder. Only a light contextual description of each token is needed as the decoder can fetch through attention precise representations of the context sequence. Bahdanau et al. [2016] also showed that visual inspection of coefficients $(\alpha_{ij})_{ij}$ provides useful information on how to interpret the model's outputs as shown in Figure 2.15 Attention usage quickly developed outside neural machine translation. As previously said, Sage et al. [2020] demonstrated the relevance of attention in document

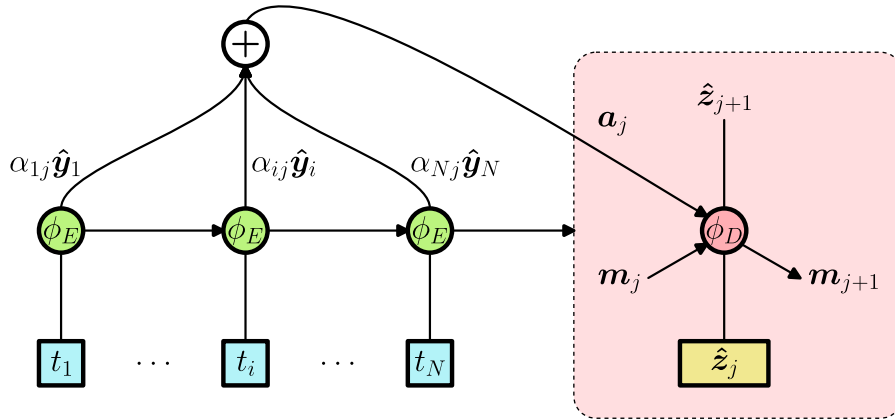


Figure 2.14: An encoder-decoder with attention. The decoder (Right, in red) can retrieve the encoder’s (Left, in green) outputs through an attention layer. For readability reasons, only the cell at step j of the decoder is represented. The scalar α_{ij} directly describes how much the decoder at step j pays attention to the encoder’s i^{th} output. It is normalized with a softmax operator so that $\sum_i \alpha_{ij} = 1$ and $(\alpha_{ij})_i$ can be interpreted as a probability distribution.

IE with an encoder-decoder architecture. Further work extends the attention mechanism with multiple modules [Palm et al., 2018] or between image and text modalities [Cheng et al., 2022].

2.2.3 Transformers

Since their introduction by Vaswani et al. [2017], *transformers* have revolutionized deep learning. Vaswani et al. [2017] did so by centralizing most advances in NLP from the last decade into a single architecture. The primary objective of the transformer architecture is to enhance parallelization in contrast to recurrent networks. This is achieved by replacing recurrent connections with *self-attention*.

Transformer Layer

Self-attention is a special usecase of Bahdanau et al. [2016] attention mechanism. Instead of two different sequences where one pays attention to the other, self-attention allows a sequence to pay attention to itself. Vaswani et al. [2017] defines attention as an operation mapping *queries* $\mathbf{Q} \in \mathbb{R}^{M \times D}$, *keys* $\mathbf{K} \in \mathbb{R}^{N \times D}$ and *values* $\mathbf{V} \in \mathbb{R}^{N \times D}$ to an output. N and M are the lengths of the two sequences ($N = M$ for self-attention) and D is the dimension of the model. Assuming the chosen similarity is the dot product, Equations 2.35-2.36 can be rewritten with matrix notations as

$$\text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V} \quad (2.37)$$

with $\hat{\mathbf{y}}$ being associated to both \mathbf{K} and \mathbf{V} , and $\hat{\mathbf{z}}$ being associated to \mathbf{Q} . The similarity matrix visualized in Figure 2.15 is in fact the matrix $\text{softmax}(\mathbf{Q}\mathbf{K}^\top)$.

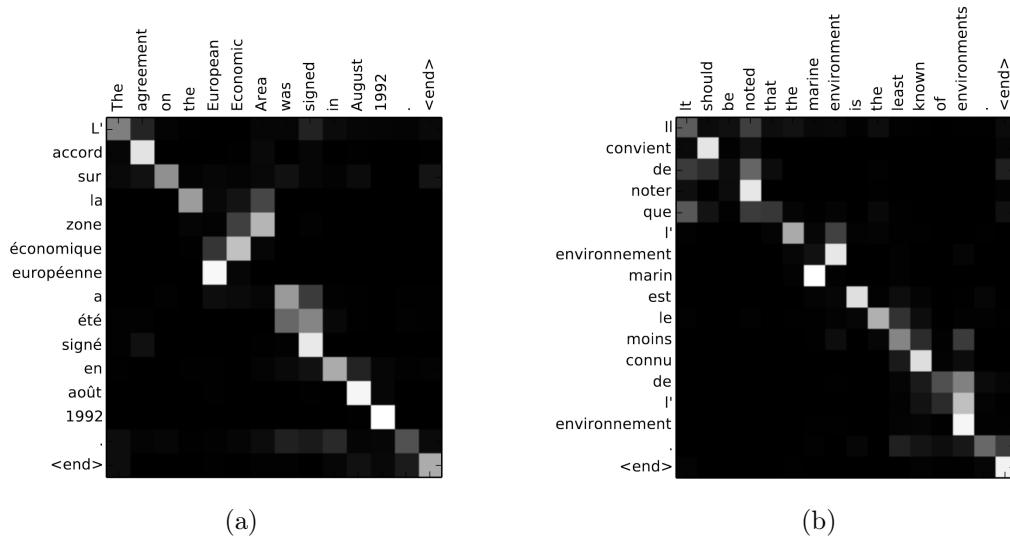


Figure 2.15: Visualization of two attention maps on a translation from English to French task. Because English and French are grammatically close, words are mostly aligned with their direct translation. This alignment results in a bright diagonal almost everywhere. However, some nominal groups are formed backward in French like “European Economic Area” in Subfigure 2.15a where the attention locally takes the shape of an anti-diagonal matrix. Visualizing attention map reveals in Subfigure 2.15b how the model understands the definite pronoun “l’” and associates it to the correct noun “environnement”. Figures reproduced from Bahdanau et al. [2016].

In practice, Vaswani et al. [2017] use $\frac{QK^\top}{\sqrt{D}}$ instead of QK^\top to limit the variance in higher model dimension. They also use *multi-head* attention, where Q , K and V are chunked into n_{head} independant attention heads. The final attention is the concatenation of the n_{head} different results given by the heads. This allows the model to focus on multiple locations at the same time, without increasing the cost of computation.

Equation 2.37 allows each token in the sequence to attend to any past and future token in a bidirectional way. While this is beneficial for an encoder, it is not suitable for decoders. In the context of a decoder, a token can only attend to previous tokens since future tokens have not yet been generated during inference. To address this, the formula can be adjusted by applying a mask to the similarity matrix, thereby forbidding attention outside of the mask. The resulting causal attention can be written as

$$\text{causalAttention}(Q, K, V) = \text{softmax}((QK^\top) + M)V \quad (2.38)$$

where $M \in \mathbb{R}^{M \times N}$ is a upper triangular matrix which non-zero values are all $-\infty$. Once softmax is applied, all attention interactions inside the mask are nullified.

To help the training, transformers use residual connections [He et al., 2016]

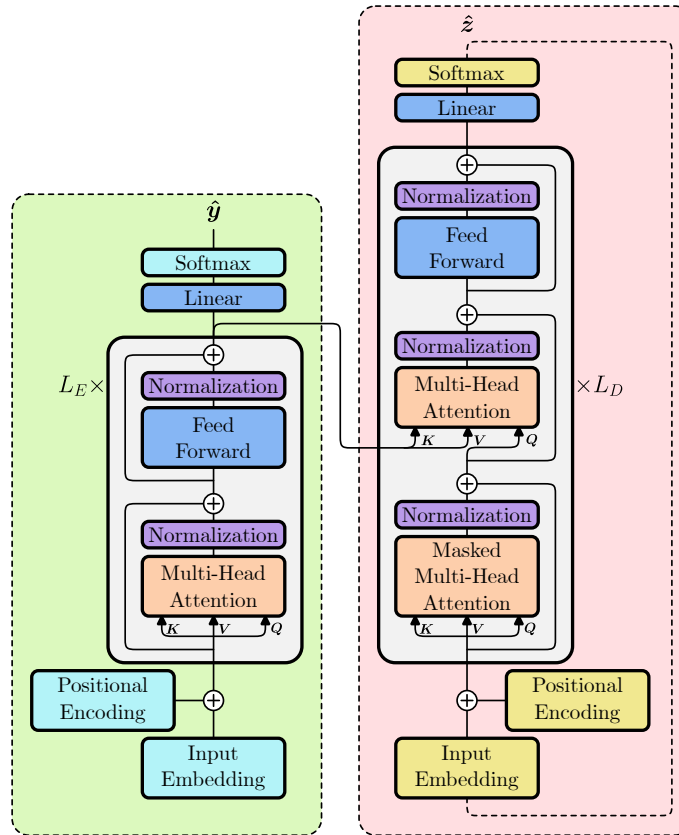


Figure 2.16: The original transformer architecture. Originally presented as an encoder-decoder architecture, the encoder (Left, in green) and the decoder (Right, in red) can be used independently. State-of-the-art models stack multiple transformer layers. A decoder-only transformer does not use the multi-head attention that takes the encoder’s output. Figure freely inspired of Vaswani et al. [2017]

around every operation, alongside layer normalization [Ba et al., 2016] and dropout [Srivastava et al., 2014]. Since the first release of transformers, Xiong et al. [2020] made the argument to move the layer normalization before self-attention, with a significant boost in performance. Zhang and Sennrich [2019] on their side proposed to replace the layer normalization with root mean squared normalization which showed speed improvements at no performance cost. But except for some minor changes, the overall transformer architecture has stayed identical to what is illustrated in Figure 2.16.

Scaling Transformers

Self-attention as described in Equation 2.37 involves a quadratic time and space complexity relative to the sequence length. It involves the computation of the $N \times N$ similarity matrix \mathbf{QK}^\top which requires $O(N^2)$ operations, hence the quadratic complexity. Most transformer models effectively limit the maximum sequence length

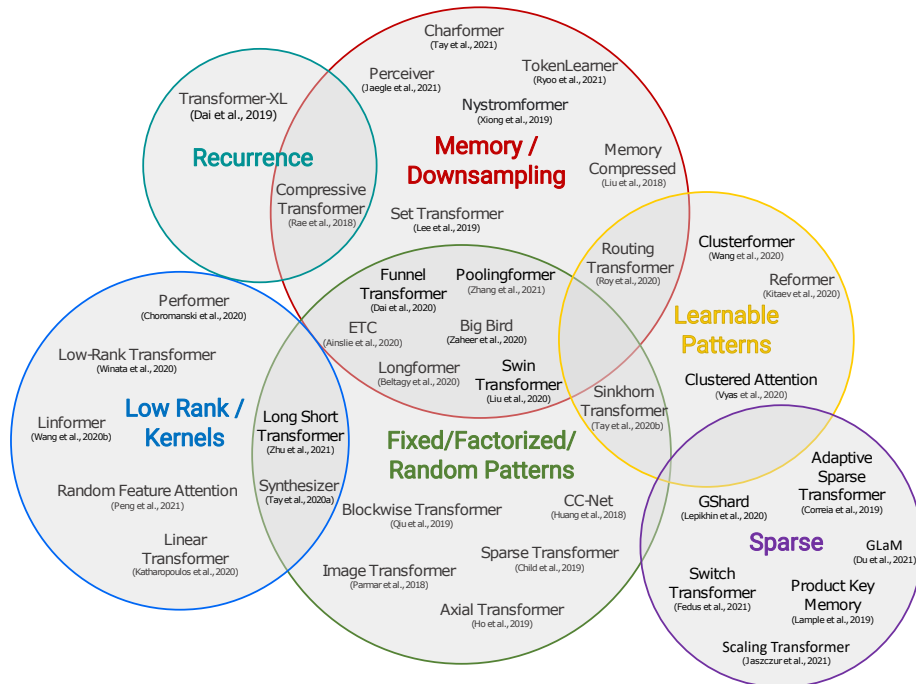


Figure 2.17: Venn diagram of efficient transformers implementations adapted to long sequences. Figure reproduced from [Tay et al. \[2022a\]](#).

to 512 or 1024 tokens due to excessive memory usage on the GPU. To reduce the memory usage of transformers, [Sanh et al. \[2020\]](#) proposed to use knowledge distillation from a large model to a smaller one, keeping most of the performance of the bigger model in the process. More recently, [Dettmers et al. \[2022\]](#) demonstrated how extreme quantization can reduce memory footprint using only 8-bit integer representations of model weights. This quantization is an improvement from the now standard 16-bit float representation used to train large models to cut the required memory in half.

To further alleviate the issue and reduce memory consumption without impacting the results of the attention operation, [Dao et al. \[2022\]](#) optimized it by taking advantage of the high bandwidth memory of GPUs. Their algorithm, FlashAttention, enables processing longer sequences by reducing the space complexity to $O(N)$ and a significant speedup over classical implementations although the time complexity is unchanged. They demonstrate the ability of a transformer with FlashAttention to deal with sequence lengths up to 64k tokens.

Several alternatives to self-attention have been developed recently with the same common objective: enabling the process of longer sequences at a minimal performance cost. Those efficient transformers [[Tay et al., 2022a](#)] use a variety of techniques to approximate full self-attention and bring down the time and space complexity closer to $O(N)$. Figure 2.17 features a proposed classification as of 2022. A simple solution is to limit the attention span of a token to a local neighborhood

around it. Longformer [Beltagy et al., 2020] explores multiple definitions of locality and context sizes. This idea was developed simultaneously by Zaheer et al. [2021] and Ainslie et al. [2020] which respectively released Big Bird and ETC. However, not all efficient transformers use fixed attention patterns, Katharopoulos et al. [2020] and Qin et al. [2022] use kernel approximation to replace the softmax term in self-attention with a product $\phi(Q)\phi(K^\top)$. This rewriting avoids explicitly computing the $N \times N$ similarity matrix which improves the complexity but removes the ability to interpret the model as shown in Figure 2.15. Other techniques rely on leveraging the low-rank properties of the similarity matrix like proposed in the Linformer [Wang et al., 2020b] or learned attentional patterns as in the Reformer [Kitaev et al., 2020]. If most efficient transformers were initially developed with NLP tasks in mind, their ability to tackle very long sequences opens up new research fields. Efficient transformer architectures applied to DU is the research topic of Chapter 5.

Input Encoding

Since transformers were initially developed for translation and other NLP related tasks, they also use tokenizers and embeddings to convert text strings into vectors. Except for rare use cases [Clark et al., 2022; Xue et al., 2022], transformers use subword tokenizers like those described in Subsection 2.2.2.

However, because transformers don't structurally represent the computation like RNNs do with one-way recurrence, transformers are invariant to permutations of their input. Vaswani et al. [2017] propose to provide additional information to the model by using a *positional encoding*. A function $\phi_{\text{pe}}: \mathbb{N} \rightarrow \mathbb{R}^D$ is charged to map each token position to a unique vector embedding with the same dimension D as the token's embedding. Multiple solutions are available, the simplest being a lookup table into a matrix of trainable weights. However, this solution does not generalize well if encountered sequences at inference are longer than those the model was trained on. To remedy this, they introduce the following parameter-free function

$$\begin{cases} \phi_{\text{pe}}(i)^{[2d]} &= \sin\left(\frac{i}{10000^{2d/D}}\right) \\ \phi_{\text{pe}}(i)^{[2d+1]} &= \cos\left(\frac{i}{10000^{2d/D}}\right) \end{cases} \quad (2.39)$$

where $i \in \llbracket 1, N \rrbracket$ is the position in the sequence and $d \in \llbracket 1, D/2 \rrbracket$ spans over the embedding dimension. This particular formula allows the model to compute embeddings of relative position shifted by k in a linear operation, which would help generalization with unseen sequence length. For example, it has been shown recently that a model trained with such fixed positional encoding on a small sequence length can be easily adapted to longer sequence lengths by interpolating the positional encoding. Several researchers proposed to enrich the positional encoding with relative bias [Shaw et al., 2018; Raffel et al., 2020; Press et al., 2022; Su et al., 2022] into the attention mechanism, contrasting with Vaswani et al. [2017] absolute positional encoding. In addition to helping the model generalize with longer sequences, it also

incentivizes the model toward short-range attention. Recently, [Kazemnejad et al. \[2023\]](#) even proposed to not use any positional encoding in a decoder-only architecture. They show that masked self-attention alone can learn both absolute and relative positioning.

Previous positional encodings provide information to the model about the position of tokens along the sequence dimension. In Visually-Rich Documents (VRDs), words are not laid in a linear sequence, they are rather arranged in the 2D plane described by the document’s page. [Xu et al. \[2020\]](#) incorporated 2D positional encoding to LayoutLM by resizing each page to a fixed size and adding a learned positional embedding for both x and y axis. If LayoutLM used word-level 2D positional encoding, StructuralLM [[Li et al., 2021a](#)] and later LayoutMask [[Tu et al., 2023](#)] proposed shared embeddings for all tokens belonging to the same line or semantic box, resulting in a better comprehension of the structure of the document by the model. Multiple models later featured 2D relative attention such as LAMBERT [[Garncarek et al., 2021](#)], TILT [[Powalski et al., 2021](#)] and FormNet [[Lee et al., 2022](#)].

If most 2D positional encodings are aligned with the input sequence in the first stages of the model, [Wang et al. \[2022\]](#) proposed to process text and layout tokens in parallel. It forces the model to learn a language-independent representation of the document’s layout and helps to build cross-language models.

Pre-Training

The revolutionary impact of transformers in the field of NLP cannot be overstated. By pushing the attention operation to its limits and introducing crucial tools like positional encoding, transformers paved the way for a transformative breakthrough. However, it was the extensive pre-training process that truly unlocked the potential of transformers in NLP. Inspired by the success of transfer learning in CV [[Yosinski et al., 2014](#)], where learned features could be effectively applied to specific tasks, researchers studied the potential of pre-training in NLP. While CV models benefited from the rich dataset provided by ImageNet [[Deng et al., 2009](#)], NLP faced a different challenge: the scarcity of large-scale supervised datasets. Nevertheless, NLP had a vast amount of valuable text available from books, news articles, and the Internet. The principle of self-supervised pre-training is to teach models how correct text is structured. Similar to how transfer learning reused learned representations of images, a pre-trained *language model* can leverage its representations of language to learn complex tasks. One way to accomplish this is to ask the model to predict the next token based on all previous tokens. This autoregressive task is commonly referred to as LM. Another way, more suited to encoders, consists of masking part of an input sequence and asking the model to recover the masked tokens. This is sometimes referred to as a *cloze* task or Masked Language Modeling (MLM) in the literature.

The first use of self-supervised pre-training in NLP could be attributed to learned word embeddings like Word2Vec [[Mikolov et al., 2013a](#)] and GloVe [[Pennington et al.,](#)

2014] which learned useful embeddings which could grasp the semantic of a word but failed at providing contextual information. Peters et al. [2018] later proposed ELMo that produces contextualized word embeddings. Howard and Ruder [2018] advocated for a novel approach to pre-training. Instead of only pre-training the embedding layer and then freezing it, they introduced a multi-step involving pre-training the language model on a large corpus, then on the downstream corpus and finally on the downstream task by adding a few additional weights on top of the model. Thanks to several training strategies, including the now commonly used triangular learning rate scheduler, they avoid catastrophic forgetting during fine-tuning. Their work has led Radford et al. [2018a] and Devlin et al. [2019] to apply similar pre-training strategies to transformers.

Generative Pre-trained Transformer (GPT) [Radford et al., 2018a] uses the standard LM task to pre-train a decoder-only transformer. It simply consists in minimizing the NLL of token t_i given all previous tokens $(t_j)_{j<i}$ as follows

$$L(\theta) = - \sum_{i \leq N} \log P(t_i = f(t_1, \dots, t_{i-1}; \theta)) \quad (2.40)$$

which can be interpreted as teaching the model f_θ to predict the very next token given all preceding tokens. Using large amounts of text for pre-training, they showed GPT performed over the top on several NLP tasks, from sentiment analysis to question answering and sentence similarity. For downstream tasks, the whole model is reused, with the addition of a small classification head specific to the involved task. Because of the relatively small amount of new parameters needing to be fully trained, GPT is very data-efficient during fine-tuning. It also significantly outperforms previous LSTM-based language models on zero-shot tasks where no fine-tuning is involved [Brown et al., 2020].

Devlin et al. [2019] later proposed Bidirectional Encoder Representations from Transformers (BERT), an encoder-only transformer able to perform a wide range of text-understanding tasks. Its bidirectional self-attention enables deep contextual representations of language. When ELMo [Peters et al., 2018] used 2 independent autoregressive LM pre-training tasks (one forward, and the other backward), BERT introduces a new task adapted to bidirectional networks called MLM. Given a sequence of tokens with a small proportion of corrupted tokens, the model needs to fix the corrupted tokens by predicting the correct original tokens. They introduce a special [MASK] token that is used as a replacement for a corrupted token. The model learns to replace every [MASK] token according to the surrounding context as described

$$L(\theta) = - \sum_{i \in \mathcal{M}} \log P(t_i = f(\bar{t}_1, \dots, \bar{t}_i, \dots, \bar{t}_N; \theta)) \quad (2.41)$$

where \bar{t}_i is a possibly corrupted token and \mathcal{M} is the set of corrupted indices. To further force the model to learn useful representation, they advocate for a slightly more complex corruption strategy where some corrupted tokens are replaced with another

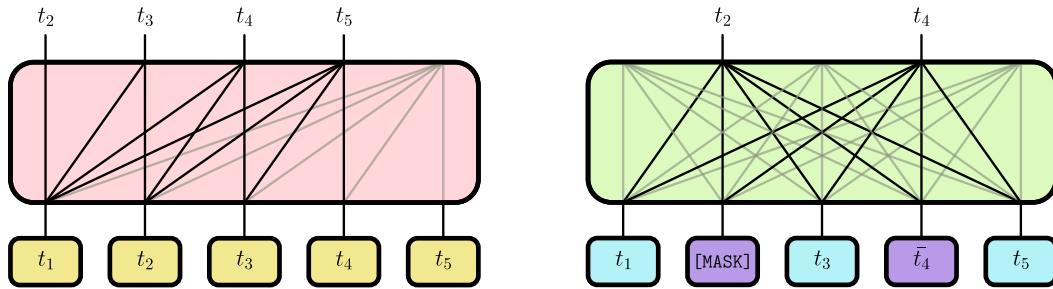


Figure 2.18: Common pre-training tasks for encoder and decoder architectures. Links in the model’s box mean the model can attend to other inputs to produce its output at a given position. Some links are grayed out as they are not used in this example to solve the task but are nevertheless allowed attention links. Decoders (Left) are trained with an autoregressive LM task in which the model learns to predict the next token based on all previous tokens. Encoders (Right) cannot be trained using the same task as they are aware of both past and future tokens. Instead, given a sequence with corrupted tokens (in purple), they learn to predict the original tokens.

random token or not replaced at all. Both LM and MLM tasks are represented in Figure 2.18. It helps the model with generalization and improves fine-tuning performance. MLM was later improved multiple times, with dynamic masking changing every epoch [Liu et al., 2019b], span-based masking strategy [Joshi et al., 2020] or by introducing a discriminative network [Clark et al., 2020].

Fully autoregressive or bidirectional models are specialized at either generating content or analyzing text based on context. By jointly training UniLM on multiple pre-training tasks including LM and MLM, Dong et al. [2019] looked for better generalization on both text understanding and generation. They were able to perform both tasks on an encoder-only transformer by modifying the attention mask according to the task. They also proposed a mixed attention mask which acts as if the encoder was an encoder-decoder model with shared weights. Bao et al. [2020] further improved the model and showed both text understanding and generation benefited the joint pre-training.

To train the encoder-decoder T5, Raffel et al. [2020] crawled a text corpus several orders of magnitude larger than any dataset at the time. They also explored a variety of sequence-to-sequence self-supervised tasks adapted to encoder-decoder models, settling on a pre-training involving multiple tasks similar to MLM. Using a larger corpus enabled the training of bigger models without any overfitting, with state-of-the-art performances achieved by an 11 billion parameters model.

Recently, Tay et al. [2022b] showed using a mixture of corruption strategies during pre-training performed better than previous pre-training tasks. Although Devlin et al. [2019] and Raffel et al. [2020] already explored MLM with a very high token corruption rate ($> 40\%$) without success, they did not vary the corruption rate during the pre-training. By using corruption strategies with variable corruption

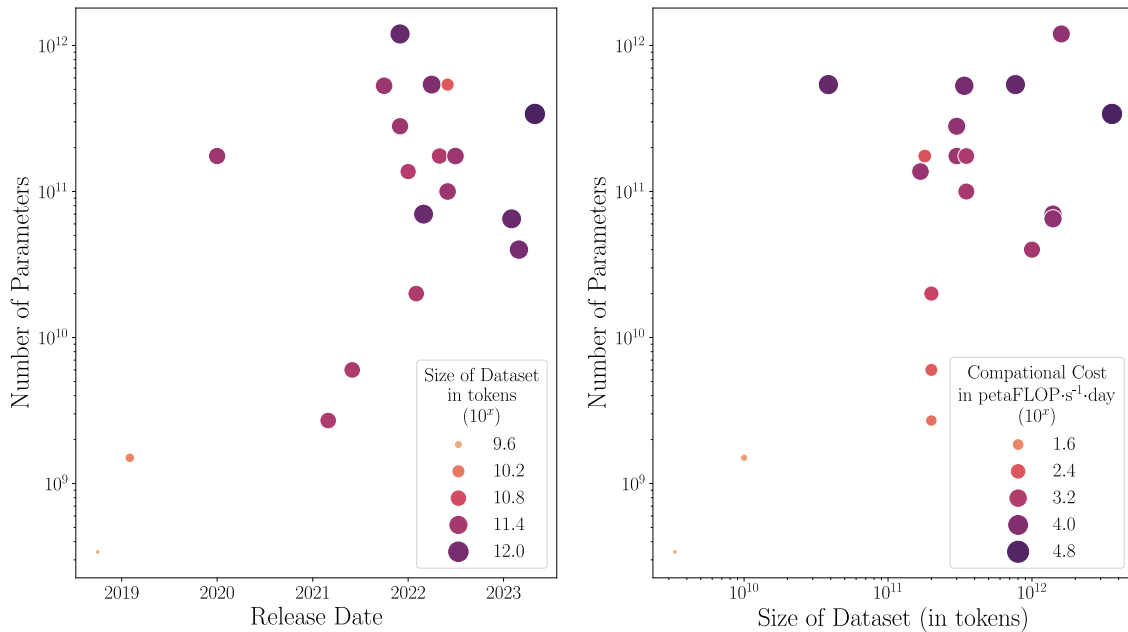


Figure 2.19: Evolution of LLMs sizes in the number of parameters, number training tokens and computational operations involved. Since late 2018 [Devlin et al., 2019] (Left), models’ sizes grew exponentially up to hundreds of billions of parameters [Chowdhery et al., 2022]. The cost of training a model (Right) has also increased by several orders of magnitude in the same period, requiring clusters of GPUs for multiple months and millions of dollars. Data is publicly available on Wikipedia¹

rates and span length, UL2 [Tay et al., 2022b] further pushed the limits of decoder and encoder-decoder models.

Since the outstanding results of Raffel et al. [2020] and Radford et al. [2018b] with T5 and GPT-2, researchers started exploring extreme model sizes. Kaplan et al. [2020], Hoffmann et al. [2022] and Scao et al. [2022] discovered an empirical relation that, for a given compute budget, links the model size with the pre-training dataset size. Today referred to as the Chinchilla scaling law, it is now guiding the development of Large Language Models (LLMs). Large models [Brown et al., 2020; Touvron et al., 2023; Workshop et al., 2023] are now associated with equally large datasets [Gao et al., 2020; Kocetkov et al., 2022; Laurençon et al., 2023] with extensive filtering and cleaning involved in order to optimize final performance and minimize the training cost associated. Figure 2.19 shows the evolution of model sizes in the last 5 years. LLMs trained on general-purpose text corpus exhibit an outstanding ability to few-shot learning [Brown et al., 2020], enabling their use for tasks with close to no supervision except for a small prompt describing the objective

¹https://en.wikipedia.org/wiki/Large_language_model#List_of_large_language_models

and a short list of examples. Most recent works show their ability to use external tools and fully interact with them [Schick et al., 2023], and build complex reasoning through chain-of-thought prompting [Wei et al., 2023].

Multimodality

When tackling VRD understanding, most model architectures involve some level of aggregation of multiple modalities such as text, layout and image. We’ve already discussed how Palm et al. [2017] and Sage et al. [2019] successfully incorporated layout information into a BiLSTM network to perform IE on BDs. As shown in Subsection 2.2.3, the same idea was applied to transformers by introducing 2D positional encodings. In LayoutLM, Xu et al. [2020] adapted the linear positional encoding from Vaswani et al. [2017] to accurately describe word position and size. To guide the model toward using jointly both text and layout modality, they introduce a layout-aware MLM task called Masked Visual Language Modeling (MVLM). They adapt the MLM task by masking the text representation of a token while keeping its 1D and 2D positional encoding intact. It incentivizes the model to manipulate 2D positions to infer the current token based on its neighborhood. Pre-training was performed on the IIT-CDIP collection [Lewis et al., 2006] with an encoder-only model based on BERT [Devlin et al., 2019]. LayoutLM has had a significant influence on recent pre-trained models for DU due to its early open-sourcing and breakthrough performance. Li et al. [2021a] use the same architecture as LayoutLM but introduce a layout classification pre-training task called Cell Position Classification (CPC). During CPC, tokens’ positional encodings are randomly masked similarly to MVLM with a predefined unique 2D position. The model is tasked to classify each corrupted position into a fixed set of document areas. This task helps the model retrieve a token’s position on the page using textual information. Using both MVLM and CPC further teaches the model how to reason jointly with both text and layout modalities. Several works later iterated over MLM for text, layout and image modalities to teach each model useful representations of documents.

More recently, Tu et al. [2023] has demonstrated the benefits of using pre-training tasks with increased difficulty [Tay et al., 2022b] compared to the classical MVLM. They adopted several strategies aimed at removing shortcuts used by the model to solve the tasks such as masking whole words at once instead of single tokens or biasing masks toward the first and last words of a segment, forcing the model to look for context in neighboring segments. This last part is complex because LayoutMask [Tu et al., 2023] does not use global 1D positional encoding, but rather one local to each segment. It keeps the model away from relying on an imperfect reading order provided by the OCR and it enables the model to learn more complex and adequate ordering of segments. LayoutMask comes with its own more difficult version of CPC, which tightens the relations between text and layout. By masking a word’s 2D position and isolating it from its segment, Tu et al. [2023] force the model to only reason based on the text in order to deduce its most likely position in the document. It is described in great detail in Figure 2.20. The design of pre-

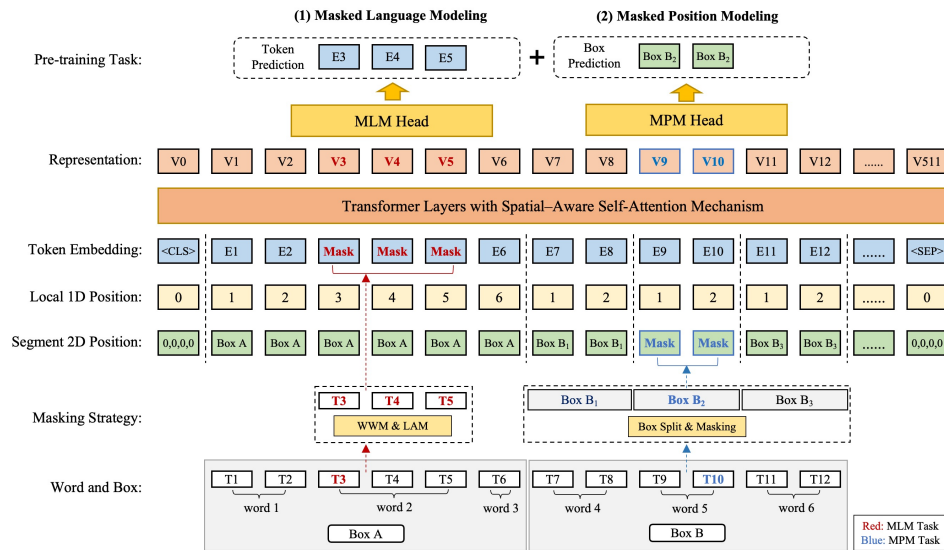


Figure 2.20: LayoutMask [Tu et al., 2023] architecture and pre-training tasks. The task named Masked Language Modeling in this figure is considerably harder than classical MLM as whole words are masked instead of tokens. Masked Position Modeling is another MLM-derived task that focuses on layout information. Figure reproduced from Tu et al. [2023].

training tasks is essential for the model’s downstream ability to jointly reason with the modalities it has access to. The same encoder-only encoder-only architecture in LayoutLM [Xu et al., 2020], StructuralLM [Li et al., 2021a] and LayoutMask [Tu et al., 2023] provided drastically different performance depending on the way information is encoded and the pre-training tasks performed.

Instead of using positional encoding attached to each token, some proposed to lay tokens onto a regular 2D grid according to their position on the page Katti et al. [2018]; Zhao et al. [2019]; Denk and Reisswig [2019]; Lin et al. [2021]. The document is then associated with an image where the channel dimension is used for token representation. This representation of a document paired with CNNs enforces the locality of the computation of features which is often desired in VRDs understanding [Garncarek et al., 2021; Lee et al., 2022]. Katti et al. [2018] first introduced Chargrid as an alternative to BiLSTM networks [Palm et al., 2017] setting the baseline for multimodal models. It uses a U-Net architecture such that IE is performed as an image segmentation task instead of the more classical token classification. It was quickly improved with the introduction of BERT contextual token embeddings [Denk and Reisswig, 2019] and the combined use of text, layout and image modality [Lin et al., 2021]. ViBERTgrid’s architecture is further described in Figure 2.21 Such methods however received less attention in recent years but are still relevant nonetheless and might surface again with pre-training tasks adapted to the architecture.

Another popular alternative to encoding layout information for the model is to

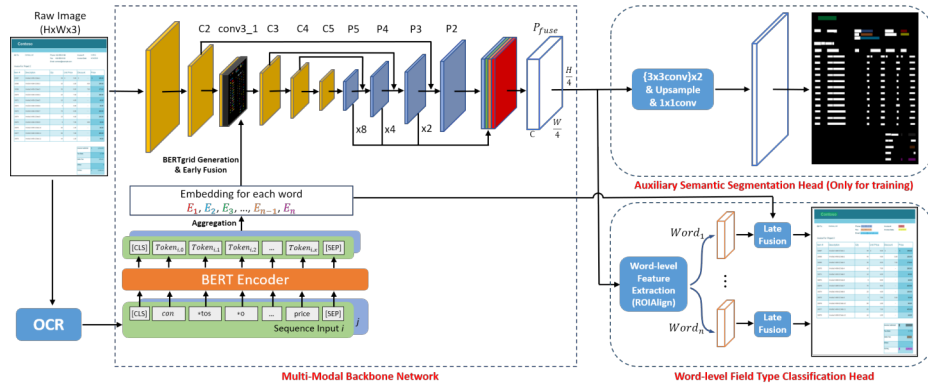


Figure 2.21: ViBERTgrid approach to multimodality. A rich multimodal feature map P_{fuse} is computed through a U-shaped CNN, enriched with BERT text embeddings. Figure reproduced from Lin et al. [2021].

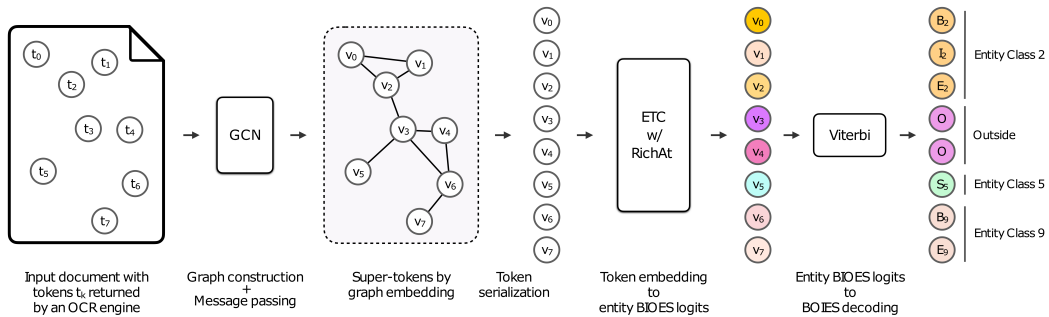


Figure 2.22: FormNet uses graph convolutions [Gilmer et al., 2017] to produce a layout-based token representation. This representation is then fed into a transformer alongside the text modality in a similar manner 2D positional encoding is inserted in LayoutLM [Xu et al., 2020]. Figure reproduced from Lee et al. [2022].

use a graph representation of a document. Although they were first introduced 2 decades ago, Graph Neural Networks (GNNs) [Gori et al., 2005] were recently popularized by the invention of more efficient layer architectures [Kipf and Welling, 2017; Veličković et al., 2018; Brody et al., 2022] and successful applications [Wu et al., 2021]. A document can be represented as a sparsely connected graph, where each node is associated with a word and is connected to its close neighboring. Several strategies have been used to create the graph based on the positions of words, by limiting to one neighbor in each cardinal direction [Lohani et al., 2019; Gal et al., 2020], by using established graph construction algorithm [Kirkpatrick and Radke, 1985; Lee et al., 2022] or by learning the graph in the process [Yu et al., 2020]. Recent architectures often use GNNs as a layout processing module in conjunction with BiLSTM [Qian et al., 2019] or transformers [Wei et al., 2020; Lee et al., 2022, 2023]. Node representations computed by the graph module are used to provide complex layout information to subsequent layers, as illustrated in Figure 2.22.

When most of the work focused on using graph representation to better encode the document, Hwang et al. [2021] introduced a graph-based alternative to sequence tagging. Named SPADE, it uses a relation tagging task between nodes of the graph to perform a variety of VRD understanding tasks expressive enough to reorder words inside the document or represent tabular and hierarchical structures. It was later used in BROS [Hong et al., 2021], which also incorporated pre-training tasks inspired by LayoutLM [Xu et al., 2020] and StructuralLM [Li et al., 2021a].

In addition to the text and its positioning inside the page, VRDs may include visual elements with semantic signification which can be helpful to interpret correctly its content. That’s why several multimodal models also use image modality in addition to text and layout. One solution is to compute the document’s representation for each modality separately and perform the fusion of modalities later in the model. This is the approach chosen in PICK [Yu et al., 2020] and DocStruct [Wang et al., 2020c]. Xu et al. [2020] also proposed a version of LayoutLM with a CNN that computes image embeddings. The main benefit of late fusion is the relative independence of the modalities until the fusion, which allows the use of specialized and established architectures for each modality. More recently, Lee et al. [2023] upgraded FormNet by adding image modality to the model. FormNetV2, the resulting model, performs the modality fusion inside a multimodal graph built based on the layout. Image features are computed over small image patches that include 2 connected words in the graph. To pre-train the model with all modalities, they introduce a new task based on contrastive learning over the graph [Li et al., 2019]. The model is trained to identify pairs of identical nodes from 2 corrupted versions of the same initial graph. By controlling which modalities are corrupted, the model is unable to rely on a single input source and must develop complex reasoning implying interactions between modalities. FormNetV2 significantly outperforms its little brother while using a similar number of parameters in several VRD understanding tasks, achieving close to SotA to this date.

Another approach to multimodality is to perform the fusion early in the model and let the model learn how to process this rich representation. It enables potential interactions and dependencies between modalities to be captured throughout the network. As previously shown in Figure 2.21, ViBERTgrid [Lin et al., 2021] uses a multimodal U-shaped CNN which processed text embeddings laid on a grid. Most works however use a multimodal transformer as the main model instead of a CNN. In TILT, Powalski et al. [2021] extract image embeddings from a U-Net [Ronneberger et al., 2015] with ROI pooling [Dai et al., 2016] to align text and image modality and sum their embeddings. Instead of manually aligning text and image modalities, Docformer [Appalaraju et al., 2021] uses ResNet [He et al., 2016] to pre-compute an image representation. The main transformer performs self-attention on both text and image sequences at each layer. To enforce collaboration between textual and visual features in the model, [Appalaraju et al., 2021] introduce new pre-training tasks. In addition to the image counterpart of MVLM, they train the model to detect when image and text features are not issued from the same document as described in Figure 2.23.

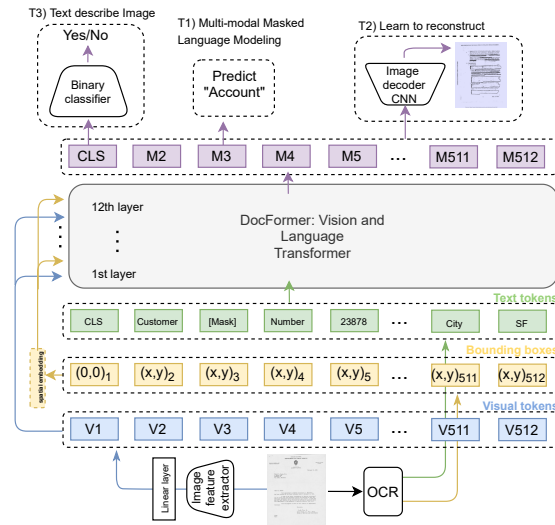


Figure 2.23: DocFormer model and pre-training tasks. Both text and image modalities are trained with a task based on masking either input sequence and predicting what was masked in the first place. However because image embedding does not rely on discrete tokens, the classification is replaced with a global image reconstruction task. Figure reproduced from Appalaraju et al. [2021].

Although transformers were first invented for NLP, they were also proven effective at CV. Dosovitskiy et al. [2021] showed a simple transformer encoder was able to outperform CNN based models on many datasets including ImageNet. By first splitting the input image into a grid of small patches and using those image patches along 2D positional encoding as the input sequence of the model, ViT has proven the effectiveness of transformers in image processing. Other transformer-based architecture quickly followed, with more efficient design [Touvron et al., 2021] or hierarchical representations adequate for small object detection and segmentation Liu et al. [2021]. Several works used the same principle for VRD understanding, using separate transformer encoders for text and image modalities like SelfDoc [Li et al., 2021b] or a single multimodal transformer encoder with an input composed of textual and visual tokens [Li et al., 2021c; Xu et al., 2021; Peng et al., 2022]. The differences between models reside in the image embedding procedure and the pre-training tasks used to teach the model how to reason using every provided modality. However, they all use continuous image embedding, unlike the discrete text tokens with finite vocabulary.

Recent work in image generation on DALL-E [Radford et al., 2021] used image tokenization in its process which converts continuous image space into a sequence of discrete tokens. Bao et al. [2022] further proposed a joint procedure to learn a visual vocabulary and pre-train a visual transformer encoder at the same time. Instead of using a general image collection to pre-train their image tokenizer, Li et al. [2022] used IIT-CDIP [Lewis et al., 2006] to obtain DiT, an image tokenizer and encoder

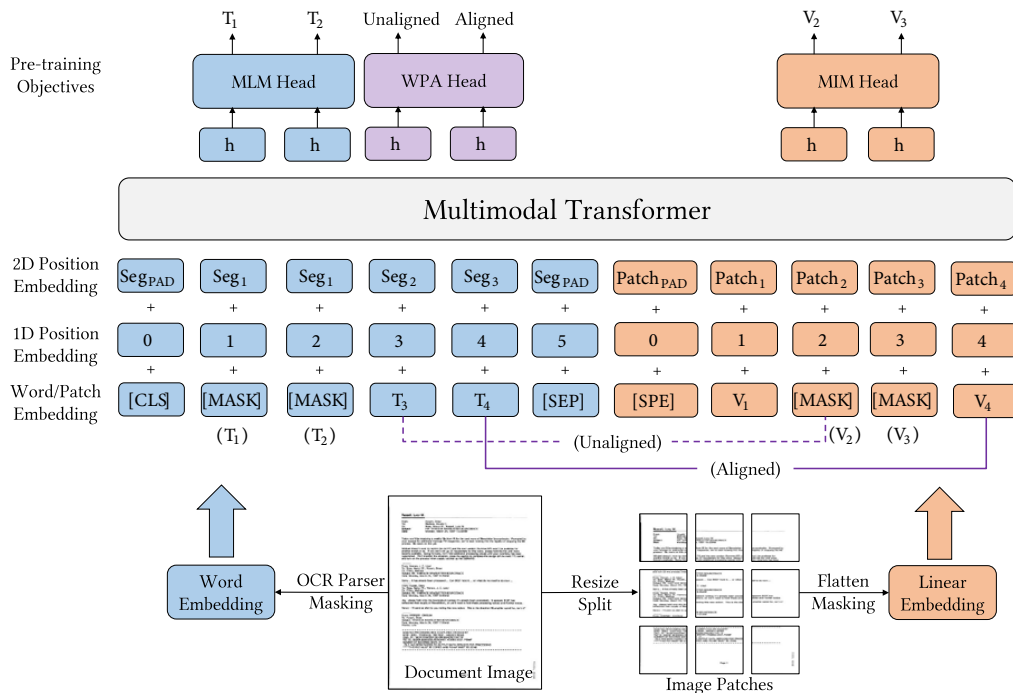


Figure 2.24: LayoutLMv3 architecture. Text and image are tokenized and fed as a single sequence to the model. Pre-training is done using 3 tasks concurrently: **MVLM** on the text tokens (Masked Language Modeling in the figure), **MVLM** on the image tokens (Masked Image Modeling in the figure) and **Word Pair Alignment** on unmasked text tokens. Figure reproduced from [Huang et al. \[2022\]](#).

specialized for **VRD**. Finally, [Huang et al. \[2022\]](#) reused the DiT tokenizer and LayoutLMv2 encoder in its latest iteration of LayoutLM as illustrated in Figure 2.24. By leveraging the discrete nature of its image tokens, LayoutLMv3 [[Huang et al., 2022](#)] is able to use the same pre-training task for text and image modalities, bridging the gap of representation between them.

If most of the previous models are mainly using a transformer encoder, it might not hold anymore in the near future. Recent works on **LLMs** [[Chowdhery et al., 2022](#); [Workshop et al., 2023](#); [Touvron et al., 2023](#)] have proven the ability of decoders to manipulate finely the language. Encoder-decoder multimodal architectures like **TILT** [[Powalski et al., 2021](#)] and **Donut** [[Kim et al., 2022](#)] will probably play a major role in the future of **VRD** understanding.

Data-Efficient Information Extraction from Documents with Pre-Trained Language Models

Chapter abstract

Like for many text understanding and generation tasks, pre-trained language 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 paper, 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. We therefore advocate the use of pre-trained language models for tackling practical extraction problems.

This work was conducted with Clément Sage and has led to the following publication in 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)*, Saint-Étienne, June 2021.

It was also published in an international workshop:

- 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), ICDAR 2021*, Lausanne, September 2021. Springer International Publishing.

Contents

3.1	Introduction	58
3.2	Related works on Information Extraction	59
3.3	Models	62
3.3.1	Encoder	62
3.3.2	Classifier	63
3.4	Datasets	63
3.4.1	Scanned Receipts OCR and Information Extraction (SROIE)	64
3.4.2	Real-world purchase orders (PO-51k)	64
3.5	Experiments	65
3.5.1	Experiment settings	65
3.5.2	Few-shot learning	65
3.5.3	Intermediate learning	68
3.6	Conclusion	69

3.1 Introduction

As shown in Chapter 2, training Artificial Neural Networks (ANNs) on a supervised task through gradient descent requires a labeled dataset. If larger models are usually more computationally powerful than smaller ones, they also need more data samples to avoid overfitting to the training examples.

Following the work of Mikolov et al. [2013a] and Heinzerling and Strube [2018] on Language Modeling (LM), the parameters related to word embeddings could be frozen during training [Lohani et al., 2019; Denk and Reisswig, 2019]. In addition to relieving the model from learning word representations, those representations usually better capture the semantics of words than what would have been learned by the model alone. We previously described how pre-trained embeddings improved models’ language understanding in Subsection 2.2.2.

In recent years, self-supervised learning strategies have gained a lot of traction in the Natural Language Processing (NLP) community. Whole models like BERT [Devlin et al., 2019] and GPT [Radford et al., 2018b] are pre-trained on masked and autoregressive language modeling tasks. Through self-supervised pre-training, those models can leverage large amounts of unlabeled data and learn useful representations. Those representations can then be adapted to any related task by fine-tuning the model on a downstream task, thus reusing the same pre-trained model for a variety of language understanding and generation tasks.

Following the current trend in the NLP field, a number of works [Xu et al., 2020; Pramanik et al., 2020; Xu et al., 2021; Hong et al., 2021] have adapted pre-trained language models from plain text to documents by performing the self-supervised

pre-training on large collections of documents. They have been fine-tuned and evaluated on several document analysis tasks such as information extraction but also document-level classification and visual question answering. Their pre-trained models have considerably outperformed 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 [Jaume et al., 2019; Huang et al., 2019; Park et al., 2019] annotated sets for training. However, this comparison has not been conducted in even more data-constrained settings that are encountered in practical applications of IE models. In this paper, we aim to quantify to what extent the pre-trained models are sample-efficient for IE tasks by comparing LayoutLM [Xu et al., 2020] with two models without pre-training.

We present three main findings that we experimentally validated using the public Scanned Receipt OCR and Information Extraction (SROIE) benchmark [Huang et al., 2019] as well as a private real-world dataset:

- 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-pre-trained 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 private dataset.

Corroborating the data efficiency of such models already observed in other NLP tasks [Howard and Ruder, 2018; Chen et al., 2020; Brown et al., 2020], our results show that using pre-trained models dramatically reduces the number of annotations required for achieving satisfying performance which is appreciable for industrial IE systems.

3.2 Related works on Information Extraction

3.2.0.1 Fully supervised models

Historically tackled by rule-based approaches [Cesarini et al., 1998; Li et al., 2008], the IE task has lately been dominated by solutions based on machine learning [Chiticariu et al., 2013]. Most Machine Learning (ML) approaches first employ an encoder, usually a few neural network layers, to obtain contextualized high-level representations of all the tokens of the document. Then, a classifier module composed of a couple of dense layers is immediately applied to these representations to classify each token according to the type of information that it carries. Most works adopting this sequence labeling approach for extracting information have focused on constituting

more powerful representations of the document tokens. The first encoders to appear were recurrent neural networks [Palm et al., 2017; Sage et al., 2019] that operate on a uni-dimensional arrangement of tokens. Later, encoders that explicitly consider the 2D structure of business documents have been proposed, thus leveraging physical layout information. These methods either represent a document as a graph of tokens [Liu et al., 2019a; Qian et al., 2019; Yu et al., 2020; Gal et al., 2020] or a regularly shaped grid on which the tokens are embedded [Katti et al., 2018; Denk and Reisswig, 2019; Zhao et al., 2019; Dang and Thanh, 2019]. Some convolutional layers are then applied to these models of documents to obtain the token representations. In addition to better understanding the document layout, some authors [Katti et al., 2018; Palm et al., 2018] also include the pixel values of the document images in the input for capturing clues not conveyed by the text modality such as table ruling lines, logos and stamps.

In all these extraction models, the whole set of their parameters, except perhaps the token embeddings [Denk and Reisswig, 2019], are learned in a fully supervised task-specific way. Specifically, they are attributed random values at the beginning of the model training. The parameters' values are then updated by directly minimizing the cross-entropy 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 from scratch.

3.2.0.2 Pre-trained models

Since the recent development of language modeling techniques [Devlin et al., 2019; Brown et al., 2020], NLP models for understanding and generating text are not learned from scratch anymore [Qiu et al., 2020b]. Rather, the mainstream approach to reach state-of-the-art performance on many downstream tasks is to adapt the parameters of models that have already learned powerful representations of the language. Such pre-training is performed in a self-supervised way on a large quantity of text data. Starting from LayoutLM [Xu et al., 2020], pre-trained models that were originally operating on serialized text have been extended to process the spatially distributed text contained in business documents, e.g. text blocks and tables.

To that end, positional embedding vectors relative to their absolute 2D coordinates are included in the token representations that are given to the Transformer encoder. Before fine-tuning the model on the downstream tasks like the fully supervised models, LayoutLM is first pre-trained on millions of document pages [Lewis et al., 2006] using a self-supervised Masked Visual Language Modeling (MVLM) task that naturally expands the main pre-training objective of Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019].

This work further inspires other language models dedicated to 2D documents. While the visual modality was introduced only at the fine-tuning stage in LayoutLM, later models [Pramanik et al., 2020; Hong et al., 2021; Xu et al., 2021] include visual descriptors from convolutional layers directly into the token representations used for pre-training. These recent works mainly focus on adding new pre-training objectives

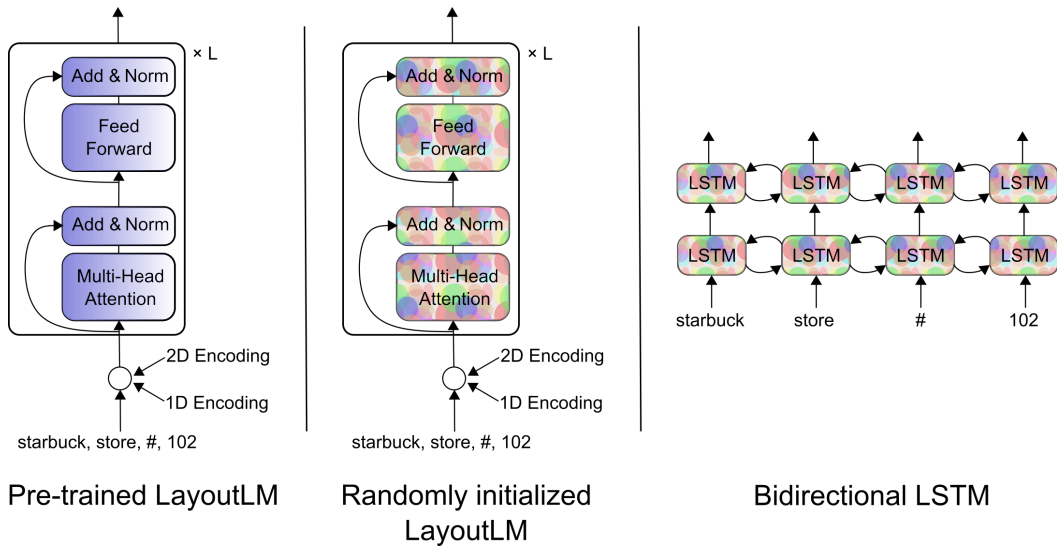


Figure 3.1: The different architectures used in our experiments for encoding documents. From left to right: Transformer-based LayoutLM Xu et al. [2020] with pre-trained weights, LayoutLM with random initialization and a 2-layer bidirectional LSTM also randomly initialized.

complementing **MVLM** to more effectively mix the text, layout and image modalities when learning the document representations, for example, the topic-modeling and document shuffling tasks of [Pramanik et al., 2020], the sequence positional relationship classification objective [Wei et al., 2020], the text-image alignment and matching tasks leveraged in [Xu et al., 2021] and the 2D area-masking strategy from [Hong et al., 2021]. Moreover, [Xu et al., 2021; Hong et al., 2021] both modify the computation of the self-attention scores to better encompass the relative positional relationships among the tokens of the document. Finally, [Pramanik et al., 2020] has resorted to page index embeddings and the Longformer’s [Beltagy et al., 2020] self-attention that scales linearly with the sequence length in order to process multi-page and longer documents.

All these pre-trained models largely surpass fully supervised models and have established state-of-the-art performance on multiple document understanding benchmarks, including common information extraction datasets [Jaume et al., 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** with few annotated data compared to models without such pre-training. Our contribution consists here of showing how pre-trained models can lead to a performance gain on low-resource downstream **IE** tasks.

3.3 Models

In our experiments, we follow the sequence labeling approach for performing IE. The evaluated models are composed of an encoder delivering contextualized representations of the tokens and a linear classifier that decodes this sequence of representations to extract information. All models only differ by their encoder.

3.3.1 Encoder

As shown in Figure 3.1, we use three different networks for encoding the business documents. We compare a pre-trained encoder with two fully supervised encoders.

3.3.1.1 Pre-trained model

We use LayoutLM from [Xu et al., 2020] as the pre-trained model, since this is the only IE work that publicly releases its pre-trained model parameters. We use its base-uncased version¹ which consists of a 12-layer transformer with a hidden size of 768 and 12 attention heads per layer, resulting in 113 million weights. It is built upon the BERT base-uncased model with 4 additional embedding vectors to represent the position of each token on the document page. This 2D positional encoding, coupled with a pre-training task that strongly binds the token’s semantic representation with its surroundings, allows LayoutLM to take advantage of the structure of the documents. Although proposed in their paper for the fine-tuning stage, we do not leverage the visual modality since it brings marginal improvements for IE. We thus solely rely on the text and its layout for constructing token embeddings. We refer the reader to their paper for more details about its architecture and pre-training stage.

3.3.1.2 Fully supervised models

For 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 IE works [Palm et al., 2017; Sage et al., 2019]. This is a 2-layer Bidirectional Long Short Term Memory (BiLSTM) network with a 128 hidden size. We reuse the same sub-word tokenizer as LayoutLM and employ only textual embeddings for tokens. The resulting model contains 8.5 million parameters.

Following standard practices, transformer and embedding layers are respectively initialized with a truncated normal and Gaussian distribution. BiLSTM layers resort to Glorot initialization [Glorot and Bengio, 2010].

¹<https://github.com/microsoft/unilm/tree/master/layoutlm>

3.3.2 Classifier

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 BIESO 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. 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 BIESO classes to obtain the field-level predictions. If a document has more than 512 tokens, its text is split into multiple sequences that are independently processed by the extraction model.

3.4 Datasets

As illustrated in Figure 3.2, we consider two IE datasets that cover different document types and extraction objectives.

64

company
AEON CO. (M) BHD. (126926-H)

3 FLOOR, AEON TAMAN MALURI SC
JLN JEJAKA, TAMAN MALURI
CHERAS, 55100 KUALA LUMPUR

GST ID : 002017394688 **address**


SHOPPING HOURS
SUN - THU: 1000 HRS - 2200 HRS
FRI - SAT: 1000 HRS - 2300 HRS

1x 00004497295 5.90SR
WET TISSUE 150S

Sub-total 5.90
Total Sales Incl GST 5.90
Total After Adj Incl GST **total 5.90**
CASH 50.00
Item Count 1 Change Amt 44.10
Invoice No: 2018032251310415556

GST Summary Amount Tax
SR @ 6% 5.57 0.33
Total **date** 5.57 0.33
22/03/2018 10:18 5131 041 0415556
0305215 DIANA BINTI JASMAN

AEON DAISO FESTIVAL MALL
TEL 1-300-80-AEON(2366)
THANK YOU FOR YOUR PATRONAGE
PLEASE COME AGAIN



REFITECH FRIDGE SERVICES PTE. LTD.
(GST No.: 888803696)

PURCHASE ORDER

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

No: **317030** **document number**
Date: **9/3/2018** **date**

Currency: SGD
Delivery: ASAP
Terms: COD

Item	Quantity	Description	Unit Price	Total Price
1	10	TSF.L, DN15 5.015.F.L.K	\$ 101,31	\$ 1 013,10
2	4	TSF.L, DN25 5.025.F.L.K	\$ 153,04	\$ 612,16
3	2	TSF.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: Subtotal: \$ 4 455,50
REFITECH Fridge Services Pte. Ltd. GST 7%: \$ 311,89
Blk 311, Ubi Yam Road 5, Grand Total: \$ **4 767,39**
1-11, UbiYamplex 1, Singapore 408663 total
Tel: 67430000, Fax: 67430001

Remarks: Jeffrey Boh
Authorized Signature

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

www.refitech.com.sg

(a) receipt from SROIE

(b) purchase order from PO-51k

Figure 3.2: A document sample for each dataset alongside their expected field values to extract. For PO-51k, we show a fictive purchase order due to privacy reasons.

3.4.1 Scanned Receipts OCR and Information Extraction (SROIE)

We train and evaluate the models on the public SROIE dataset [Huang et al., 2019] containing restaurant receipts. We only consider its information extraction task that aims to retrieve the *name* and *address* of the company issuing the receipt, the *total amount* and the *date*. The dataset gathers 626 receipts for training and 347 receipts for testing. 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.

Each receipt is given the ground-truth value for the four targeted fields. The comparison with the model predictions is made in terms of exact matching of strings, leading to precision, recall and F1 score metrics². For the sake of readability, we only report the F1 scores averaged over all the targeted fields. To establish the BIESO labels, we look for the receipt words matching the ground-truth field values. For the total amount, a value may match different sets of words, e.g. the amounts without taxes or after rounding. If so, we select the bottom-most occurrence having the keyword `total` in its line.

We use the provided Optical Character Recognition (OCR) results containing a list of text segments and their bounding boxes. As noticed by many submissions in the leaderboard including LayoutLM’s authors, 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. Therefore, following previous works, we manually fix them in the test set while we perform fuzzy matching for deriving the token labels in the training set. Because the order of text segments is sometimes faulty, we also re-arrange them from top-to-bottom.

3.4.2 Real-world purchase orders (PO-51k)

To prove the efficiency of the IE models, we also conduct experiments on a private dataset composed of 51,000 English Purchase Orders (POs) that were processed on Esker’s document automation solution. We split the dataset into 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 implies that for a large portion of the test set, the layout and content organization of documents have not been seen at training time.

We aim to extract 3 different fields among these purchase orders: the *document number*, the *date* and the *total amount*. The ground truth for these fields is directly provided by the end-users of the automation software, ensuring high-quality annotations. We employ the same methodology as in SROIE for evaluating the models. Text of documents is retrieved thanks to a commercial OCR engine.

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

²The metric values are obtained at <https://rrc.cvc.uab.es/?ch=13&com=evaluation&task=3>

a total amount on the first page while only 10% of the documents are affected for the two other fields.

3.5 Experiments

3.5.1 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 in [Xu et al., 2020], 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 BiLSTM model, we employ a higher initial learning rate of 5e-3 since the former value did not converge correctly. 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 fixed 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 dataset³.

3.5.2 Few-shot learning

For both datasets, we first study the performance when the models independently learn the IE task from a 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.

3.5.2.1 SROIE

We show F1 scores for the SROIE dataset in the Figure 3.3. We first notice that we get 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 its paper [Xu et al., 2020] when considering the 626 documents of the original training set.

³<https://github.com/clemlsage/unilm>

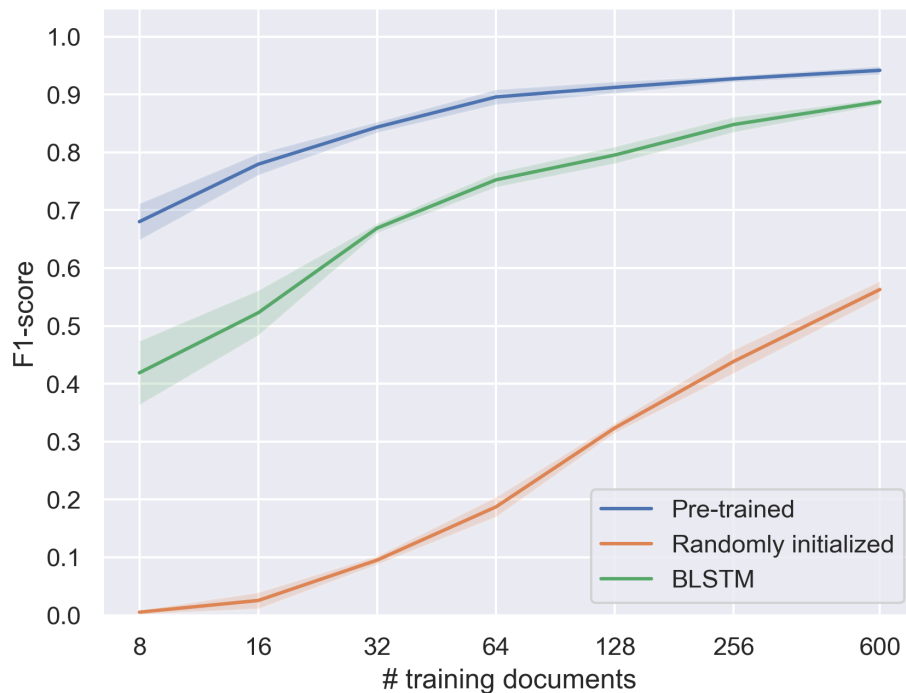


Figure 3.3: Few-shot extraction performance on the SROIE [Huang et al., 2019] test set for the pre-trained LayoutLM [Xu et al., 2020] against its randomly initialized version and a BiLSTM network.

The model convergence is fast, hitting 90% of its full performance with only 32 documents, i.e. an 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 the difference in F1 score between these two classes of models. For instance, even if the BiLSTM 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 which 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.

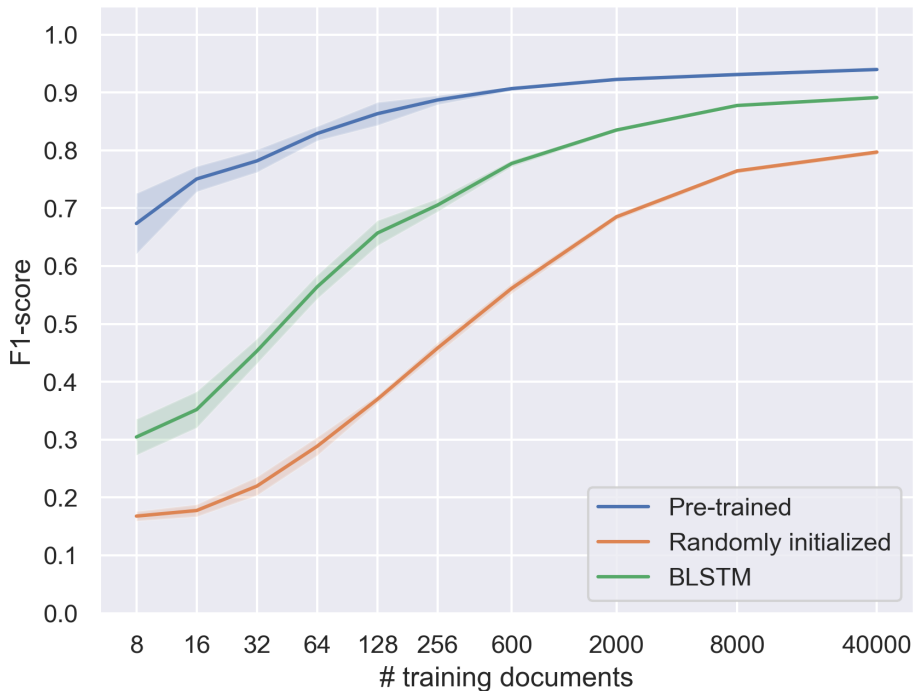


Figure 3.4: Few-shot extraction performance on the PO-51K test set for the pre-trained LayoutLM [Xu et al., 2020] against its randomly initialized version and a BiLSTM network.

3.5.2.2 PO-51k

We show F1 scores for the PO-51k dataset in the Figure 3.4. 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 BiLSTM model is on average of 0.37 F1 score until 32 documents while it was on average of 0.23 points for SROIE. The BiLSTM trained with 600 documents performs on par with LayoutLM fine-tuned on only 32 documents, i.e. an order of magnitude less annotations. We also note that this real-world dataset is notoriously more complex than SROIE since a few hundred 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 BiLSTM.

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

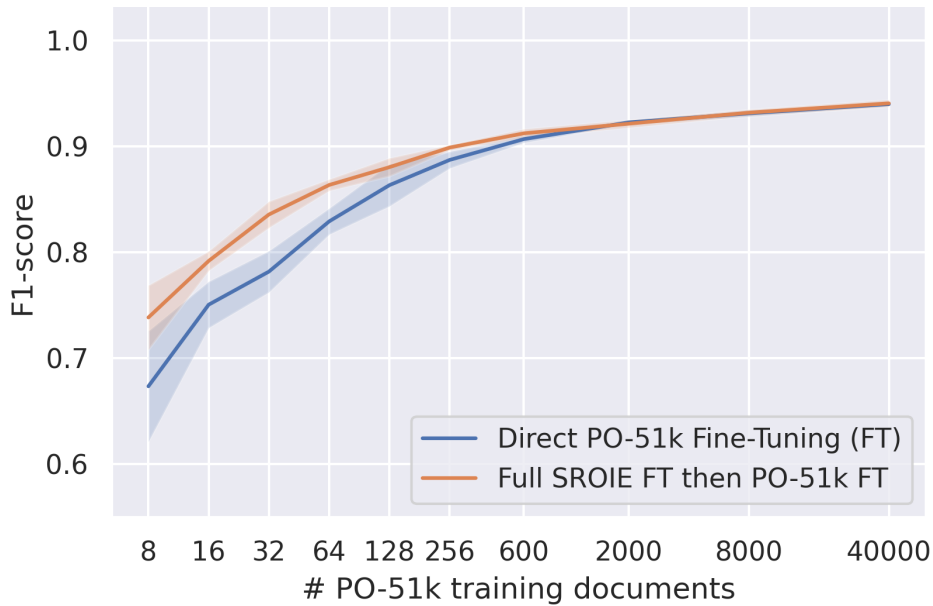


Figure 3.5: Test F1 scores of pre-trained LayoutLM when transferring extraction knowledge from SROIE to PO-51k tasks. The IE performance is always improved by resorting to SROIE as an intermediate task, the boost being significant with few available PO-51k documents for fine-tuning.

3.5.3 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 learning the classifier parameters from scratch between the two IE tasks, there are only a few thousand compared to the million weights of the encoder. We therefore hope that LayoutLM can still transfer some knowledge from SROIE to PO-51k tasks.

3.5.3.1 SROIE to PO-51k

We compare the few-shot performance on PO-51k when having first 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 3.5.

We note that the fine-tuning on [SROIE](#) considerably improves the extraction for a 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.

3.6 Conclusion

In this chapter, we showed that pre-trained language models like LayoutLM [[Xu et al., 2020](#)] are highly beneficial for extracting information from a 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 learned from scratch the [IE](#) task. We finally demonstrated that pre-training brings additional improvements when transferring knowledge from an [IE](#) task to another.

The valuable insight into pre-trained models from this work has led us to further study pre-training tasks to optimize the resulting model. The next chapter describes how specialized pre-training tasks for documents can improve a model’s performance on downstream [IE](#).

Improving Information Extraction on Business Documents with Specific Pre-Training Tasks

Chapter abstract

Transformer-based Language Models are widely used in Natural Language Processing (NLP) related tasks. Thanks to their pre-training, they have been successfully adapted to Information Extraction (IE) in Business Documents (BDs). However, most pre-training tasks proposed in the literature for business documents are too generic and not sufficient to learn more complex structures. In this paper, we use LayoutLM, a language model pre-trained on a collection of business documents, and introduce two new pre-training tasks that further improve its capacity to extract relevant information. The first is aimed at better understanding the complex layout of documents, and the second focuses on numeric values and their order of magnitude. These tasks force the model to learn better-contextualized representations of the scanned documents. We further introduce a new post-processing algorithm to decode BIESO tags in IE that performs better with complex entities. Our method significantly improves extraction performance on both public (from 93.88 to 95.50 F1 score) and private (from 84.35 to 84.84 F1 score) datasets composed of expense receipts, invoices, and purchase orders.

This work has led to a presentation at an international symposium:

- Thibault Douzon, Stefan Duffner, Christophe Garcia, and Jérémy Espinas. Extraction automatique d'informations dans les documents d'entreprise. In *Symposium International Francophone sur l'Écrit et le Document (SIFED)*, Lyon, December 2021.

It also led to the publication of a paper and an oral presentation at an international workshop. It received the Nakano award, rewarding the best paper of the event.

- Thibault Douzon, Stefan Duffner, Christophe Garcia, and Jérémy Espinas. Improving information extraction on business documents with specific pre-training tasks. In *Proceedings of the 15th International Workshop on Document Analysis Systems (DAS)*, La Rochelle, May 2022. Springer International Publishing.

Contents

4.1	Introduction	72
4.2	Related Work	73
4.2.1	Information Extraction	73
4.2.2	Pre-Training	74
4.3	Models	75
4.3.1	Architecture	75
4.3.2	ConfOpt Post-Processing	76
4.4	Pre-training	77
4.4.1	Numeric Ordering Task	77
4.4.2	Layout Inclusion Task	78
4.5	Datasets	79
4.5.1	Business Documents Collection	80
4.5.2	Business Documents Collection – Purchase Orders	81
4.5.3	ICDAR 2019 – Scanned Receipts	81
4.6	Experiments	81
4.6.1	Post-Processing	82
4.6.2	Business Document-Specific Pre-training	83
4.7	Conclusion	85

4.1 Introduction

Chapter 3 showed how models historically used for natural language, could also perform IE on Business Documents (BDs). Although text and documents differ in many ways, they share sufficient similarities such that once the text is extracted from the document, it can be processed in the same way as a text would be with great results. The difference between text and documents are many, the first one being the 2D structure of the document. The position of words forms a complex structure that guides the reader toward a reading order. An incorrect reading order inference might mislead the reader and alter the meaning of the content of the document. Comparatively, a text is a simple linear sequence of words forming sentences and paragraphs. The second critical difference between natural text and BDs is the grammar ruling the construction of a document. BDs don't use correctly formed sentences, instead they heavily use noun phrases to designate fields: "invoice number", "date" and "total amount" for instance. The context around a word is often poorly defined as those groups usually only form a key-value colon-separated pair with the associated value. Besides the lack of context, BDs use a tremendous amount of numerical figures with different semantics: gross amounts should not be mistaken with taxes or quantities and are not expressed with the same unit. Their

meaning can only be inferred by the correct association with its key in the case of a header field or with the correct table header for table fields.

To help models process documents, many have proposed modified architectures to incorporate layout information by providing the position of words to the model. We already have extensively discussed text and layout multimodality in subsection 2.2.3. Most recent proposals often include a pre-training step, where the model is trained on a pretext task. Those pretext tasks are self-supervised problems that teach the model many useful skills for manipulating the data. When pre-training an encoder network, Masked Language Modeling (MLM) is the most common task to perform. It teaches the model to use the surrounding context of a position to predict which word would best fit. The specificity of BDs needs specific pre-training tasks to learn a finer representation of a document.

In this work, we focus on LayoutLM [Xu et al., 2020], a pre-trained transformer [Vaswani et al., 2017] that is specialized in documents. It reuses the same transformer layer with multi-head attention with the addition of a 2D positional encoding. Its larger version achieved state-of-the-art performance in both document classification and information extraction. However, the required hardware to train it can be repelling. In this paper, we propose new pre-training tasks specific to BDs that will provide additional skills to the model. We also propose a new decoding post-processing algorithm that prevents many errors made by the model due to ambiguities. Combined, our contributions¹ allow for the base LayoutLM model to perform on par with the larger version.

4.2 Related Work

4.2.1 Information Extraction

Rule-based approaches [Li et al., 2008] have been supplanted by deep learning models in the last decade. Document IE first capitalized on the state of the art in Named Entity Recognition (NER) for NLP [Lample et al., 2016]. Recurrent Neural Networks (RNNs) with Long Short Term Memory (LSTM) cells were first used to encode documents at a word level [Palm et al., 2017; Sage et al., 2019], allowing a simple classifier to predict each word’s associated label. With models performing word-level predictions, IE can be represented as a sequence labeling task. Instead of a softmax and cross-entropy loss, a Conditional Random Field (CRF) [Lafferty et al., 2001] model has been used in addition to BIESO tags. Other architectures have also been proposed to better adapt to the specificity of the document. For example, graphs [Lohani et al., 2019; Liu et al., 2019a; Yu et al., 2020; Gal et al., 2020] and convolutions over a grid [Katti et al., 2018; Denk and Reisswig, 2019; Lin et al., 2021] constrained the model based on the words’ positional information. Because most architectures relied on textual representations, they benefited from pre-trained word embeddings like Word2Vec [Mikolov et al., 2013a] or GloVe [Pennington et al.,

¹Code available here: <https://github.com/thibaultdouzon/business-document-pre-training>

2014].

With the emergence of transformers [Vaswani et al., 2017] and text encoders like BERT [Devlin et al., 2019], attention-based document analysis models [Xu et al., 2020; Garncarek et al., 2021; Tu et al., 2023] evolved quickly, which resulted in a large improvement of state-of-the-art performance. In line with [Katti et al., 2018] which included both textual and visual representations, multimodal transformers [Xu et al., 2021; Appalaraju et al., 2021; Lee et al., 2023] superseded conventional textual models.

In parallel to the rise of transformers, end-to-end IE models tried to reduce the labeling cost. First using Bidirectional Long Short Term Memorys (BiLSTMs) with attention layers [Palm, 2019; Sage et al., 2020], then shifting to transformers [Powalski et al., 2021]. Adopting at the same time the Question Answering (QA) format [Gardner et al., 2019], instead of the usual sequence labeling, provided more flexibility on the predicted labels. Combining end-to-end models with multimodality enables removing the separate Optical Character Recognition (OCR) and including it directly in the model as shown by Kim et al. [2022].

4.2.2 Pre-Training

Self-supervised training and pre-trained models were popularised in NLP with enriched word embeddings [Mikolov et al., 2013a; Pennington et al., 2014; Peters et al., 2018]. Their capacity to leverage large amounts of unlabeled text has proven to be extremely effective in initializing word representations. With the emergence of the transformer architecture, large pre-trained models have been proposed [Wolf et al., 2020]. Thanks to their pre-training, they can efficiently adapt to various tasks [Wang et al., 2019, 2020a] and data types. In general, these models are pre-trained on large unlabeled datasets in a self-supervised manner. This self-supervision removes parts of the burden of data labeling as shown in Chapter 3 and leverages the huge quantities of available data.

A wide variety of pre-training tasks have been proposed. General-purpose tasks aiming at learning the language and grammar were used first. If auto-regressive Language Modeling (LM) tasks [Radford et al., 2018a] are adequate for decoder models, MLM tasks [Devlin et al., 2019] best fits encoders as they allow the model to use the context coming from both directions. MLM tasks have gradually evolved to adapt to multimodal inputs [Xu et al., 2020] and served as a proxy to teach the model to combine modalities together early in its computational process [Huang et al., 2022; Lee et al., 2023]. Most models use several pre-training tasks, either varying the impacted modality [Appalaraju et al., 2021] or the difficulty [Tay et al., 2022b].

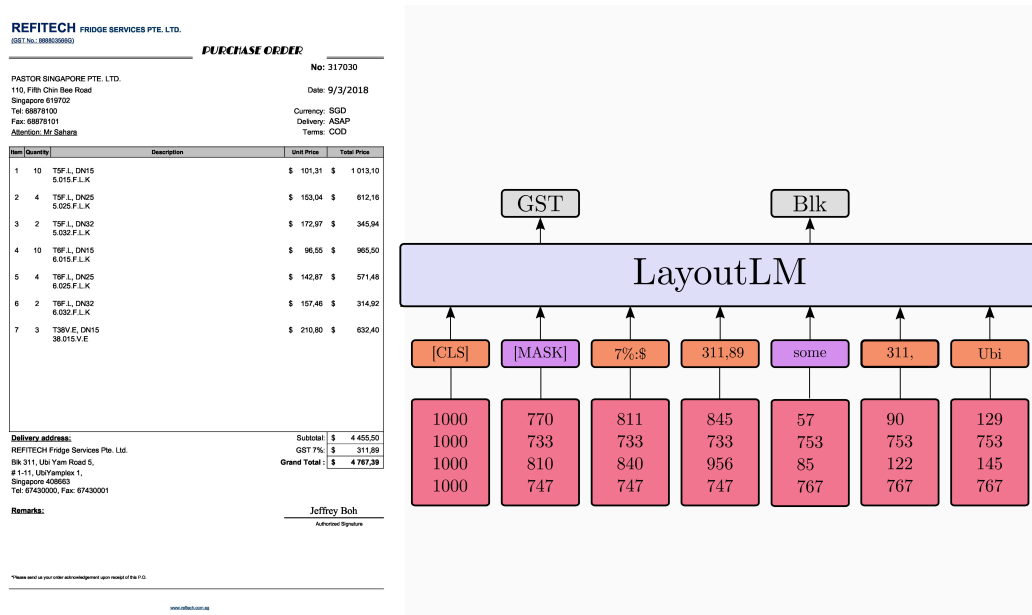


Figure 4.1: LayoutLM’s [Xu et al., 2020] MVLM on a document. A small proportion of words are corrupted (shown in purple), and the model is trained to predict the original words. The position of words is never corrupted to incentivize the model to learn correlations between a word’s position and its surrounding 2D context.

4.3 Models

4.3.1 Architecture

We used the well-established LayoutLM architecture [Xu et al., 2020] which itself is based on Bidirectional Encoder Representations from Transformers (BERT) transformer encoder architecture [Devlin et al., 2019]. More specifically, we chose the base model² with 12 layers and 512 dimensions for token embeddings. This model is computationally much more efficient compared to the larger version while still giving very good performance.

Transformer models work on tokens that are in between characters and words. LayoutLM and BERT both use the WordPiece algorithm [Wu et al., 2016]. We use the same tokenizer as LayoutLM to compare our performance with the base LayoutLM model. It uses a vocabulary size of 30000, and we limit the sequence length to 512 tokens, including the special tokens [CLS] and [SEP]. This limitation due to GPU memory consumption of self-attention operations often forces us to cut documents into multiple pieces of 512 tokens and process them separately.

Contrary to RNNs, all positions in the sequence are equivalent in a transformer model. To provide information about position inside the sequence, a linear positional encoding [Vaswani et al., 2017] is added for each token. Then LayoutLM adapted this positional encoding to a 2D version that can represent the positions of words

²Pre-trained weights available here: <https://huggingface.co/microsoft/layoutlm-base-uncased>

on a page.

For both pre-training tasks and fine-tuning, we use a simple dense layer to map each token’s final internal representation to the dimension of the prediction space. A softmax layer is applied to produce the final model confidence scores. For training, the cross-entropy loss is used on the model confidence scores.

4.3.2 ConfOpt Post-Processing

We model the Information Extraction task as sequence tagging on tokens. Predictions are done at the token level and then aggregated by a lightweight post-processing step to give the model’s final prediction. In all experiments, we use BIESO tagging. That is, each field to extract is composed of a sequence of target tags of the following types: **B** for the beginning of the entity, **I** for inside, **E** for its end, or otherwise **S** for a single token entity. **O** is used for any token that is outside any target label. BIESO is widely used in IE as it provides structure to the target sequence that helps the model.

Instead of the trivial post-processing which consists of simply following the maximum confidence of the model, we decided to decode a model’s prediction by solving a basic optimization problem. We will refer to this method as ConfOpt in the rest of the paper. The predicted sequence for a target label is the sequence that maximizes model confidence over the whole input sequence. There is a constraint to decode a prediction: it must match the following regular pattern: $(BI^*E) | S$ where $*$ denotes zero or many occurrences and $|$ denotes an alternative.

This optimization problem can be solved with a dynamic programming approach. The model’s predictions for one target label can be represented as a $4 \times N$ dimensional matrix where N is the sequence length and 4 comes from the 4 tags **B, I, E, S**. By noting $C_{T,0}$ the model’s confidence in **T** tag at position 0 and $P_{T,i}$ the best prediction confidence ending at token i with tag **T**, the objective is to determine $S = \max_{\substack{0 \leq i < N \\ T \in \{E, S\}}} P_{T,i}$ where

$$\begin{aligned}
 P_{B,i} &= C_{B,i} ; P_{I,i} = C_{I,i} + \max \begin{cases} P_{B,i-1} \\ P_{I,i-1} \end{cases} \\
 P_{S,i} &= C_{S,i} ; P_{E,i} = C_{E,i} + \max \begin{cases} P_{B,i-1} \\ P_{I,i-1} \end{cases}
 \end{aligned}
 \tag{4.1}$$

One drawback of this post-processing is dealing with no predictions and non-unique predictions. It can be solved with an empirically determined threshold below which no predictions are made. Though in this paper this is not further studied because fields are mandatory in a document and always unique.

4.4 Pre-training

Transformer models provide great performance when first pre-trained on pretext tasks on very large unlabelled datasets. This pre-training is most of the time done in a self-supervised manner in order to avoid the labeling cost. LayoutLM uses Masked Visual Language Modeling (MVLM) [Xu et al., 2020] which is adapted from BERT’s MLM [Devlin et al., 2019]. It teaches the model how text and documents are formed at a token level. In practice, at each training step, 15% of the tokens are randomly chosen and replaced by either a [MASK] token, a random token, or not replaced at all. The model tries to guess which token is the most probable right replacement at those positions.

For all pre-training tasks when a document is too long to be processed at once, we randomly select a continuous span of words of maximum size and provide it to the model instead. We expect the model to learn useful features on various parts of documents thanks to the long training. For very short documents, the input is padded to the maximum size.

We introduce two new specific pre-training tasks in addition to MVLM. The first one, Numeric Ordering (NO) teaches the model how to compare and order numbers. The second one, Layout Inclusion (LI) focuses on words in the 2D plane and their relative positioning. We chose to avoid regression tasks, even though their implementation would have been simpler. For example, simply removing the 2D positioning of some tokens, and asking the model to predict tokens’ position is an alternative to what we propose. But this does not behave well for a token that could appear either at the top or the bottom of the document: the model would learn its mean position – the middle – where the token would never appear. In the following, we will describe the two pre-training tasks in detail.

4.4.1 Numeric Ordering Task

NO focuses on numeric figures in the document and their relative values. Contrary to MVLM which only relies on self-supervised data, NO relies on a handcrafted number parser to find and parse all numbers that appear in a document. Because business documents are mostly made of decimal numbers written with digits, we ignore those written out in alphabetical characters. The numeric value of each token is determined by parsing each word in the document, looking for numbers and ignoring irrelevant characters.

As shown in Figure 4.2, the model must predict for every numeric figure in the document if its parsed value is smaller, equal or greater than a randomly selected number in the document. The loss is only computed on tokens starting a new word, but tokens continuing a word are important to determine the value represented by a word.

We want the model not only to reason on the textual features but also on the spatial context surrounding each figure in the document. Therefore, we randomly mask the textual representations of 15% of the numbers in the document and replace

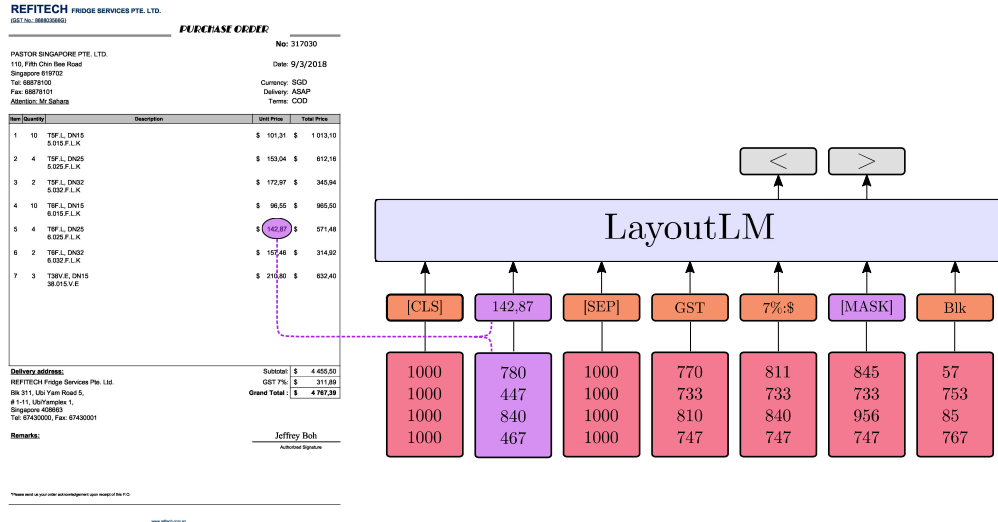


Figure 4.2: A pre-training example with NO task. A random token containing a number is selected, then the target is to predict whether other numbers are smaller or bigger. Some random noise can be added by masking tokens’ textual or spatial representations. Only a small part of the document’s input is represented in this illustration.

them with the [MASK] token as shown in Figure 4.2. For the same reason, we also mask the spatial encoding of 15% of the numbers and make sure both text and position are not masked at the same time. All masked positions are replaced with (1000, 1000, 1000, 1000).

4.4.2 Layout Inclusion Task

We introduce another pre-training task focusing on the 2D positional encoding, which we call LI. Its purpose is to provide a better understanding of document layouts and complex structures. In fact, most business documents, including invoices, purchase orders, and expense receipts, contain tables where the meaning of tokens is mostly driven by their position relative to headers.

As shown in Figure 4.3, LI is formatted like a QA prompt: a question followed by the content of the document. The question is simply a special token [LAYOUT] positioned at random coordinates (x_1, y_1, x_2, y_2) . The model must then classify every token in the document into 2 groups: either **inside** or **outside** of the question token. More precisely, the target answer is whether the middle point of a document token

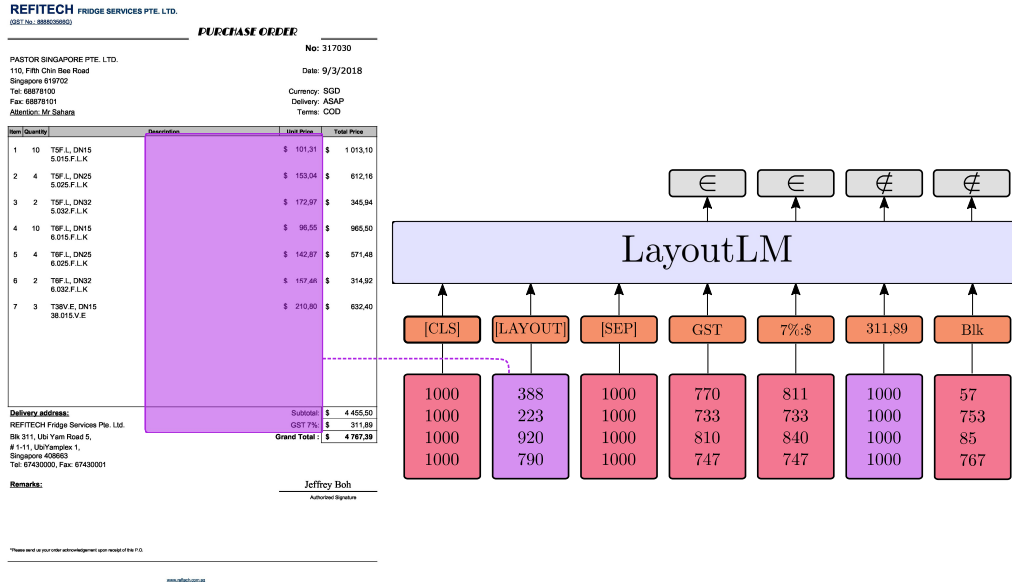


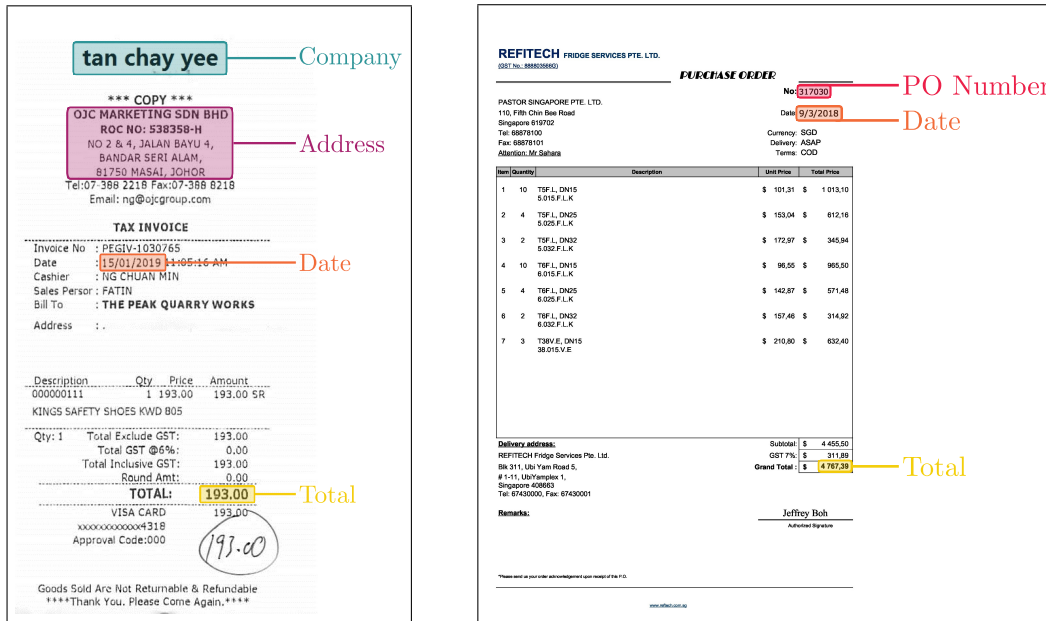
Figure 4.3: A pre-training example with LI task. The coordinates of the purple rectangle are drawn uniformly. Random noise is added by masking the 2D position of some tokens. Only a small part of the document is represented.

is inside or outside the rectangle described by the coordinates of the question.

Again, the objective is for the model to not only reason on the 2D positions of tokens but also use their textual embedding. In order to force the model to use both representations, we randomly replace 15% of the document’s token positions with (1000, 1000, 1000, 1000). In case of a random position replacement, the target value is still computed based on the real position of the token, and the model must make its prediction based on the token’s text and the neighboring tokens using the classical 1D positional encoding.

4.5 Datasets

We used 2 different collections of documents to build 3 datasets for training and evaluation as described in the following. They all contain business documents: invoices and purchase orders for the private collection and expense receipts for the public one. The largest dataset used for pre-training isn’t labeled, document samples with their target fields for the other datasets are shown in Figure 4.4.



(a) Receipt from SROIE.

(b) Purchase Order from BDC-PO.

Figure 4.4: A document sample for each training dataset annotated with the expected predictions. For BDC-PO, we replaced the document with a fictive one due to privacy reasons.

4.5.1 Business Documents Collection

The Business Documents Collection (BDC) is a large private dataset composed of 100k invoices and 300k purchase orders. Those real documents were submitted and processed on a commercial document automation solution in the last 3 years. It contains English-only documents divided into 70000 different issuers. All documents sharing the same issuer usually use the same information template. Therefore, we limited the maximum number of documents of the same issuer to 50. It is important to keep the number of similar layouts in the collection low and the variety of examples high. We used this collection for pre-training language models on business documents that are closer to our final objective than RVL-CDIP [Lewis et al., 2006]. In practice, the collection is a superset of Purchase Order (PO)-51k used in Chapter 3, with vastly more documents and diversity.

Textual and positional information has been extracted using a commercial OCR system. It achieves excellent accuracy on properly scanned documents and provides accurate word positions. We also use the provided read order to determine the order of tokens when feeding the network. This order determines the 1D positional encoding given to each token that complements the 2D positional encoding.

Because we only used this collection for pre-training models on self-supervised tasks, most documents do not have extraction labeling. Only a subset composed of purchase orders is labeled for the IE task.

4.5.2 Business Documents Collection – Purchase Orders

We selected a subset of the Business Documents Collection (BDC) to build a labeled dataset of English purchase orders called BDC-PO. It contains almost 9000 different issuers split into training, validation, and test sets. In order to not introduce bias for models pre-trained on the BDC, we removed from BDC all documents emitted by a supplier contained in the test set. This means that document layouts contained in the test set have never been seen before by the model at pre-training or training time. This labeled dataset is a bigger and refreshed version of PO-51k, with both datasets sharing a significant proportion of documents.

Long purchase orders are rare but can sometimes be longer than 20 pages. If we wanted to train models and make predictions on such documents, we would have to evaluate the model on dozens of inputs for one document. Instead, we chose to limit documents to one page and crop the remaining. It only concerns roughly 25% of the dataset and sometimes impacts the prediction because labels are missing from the input.

The extraction task consists of 3 fields: *document number*, *delivery date*, and *total amount*. Those fields were chosen because they are mandatory for most customers and thus are well labeled at the word level by the end-user. We controlled the labeling quality at the issuer level and rejected from the dataset some issuers with undesirable labeling practices.

4.5.3 ICDAR 2019 – Scanned Receipts

We also trained and evaluated our model on the public Scanned Receipt OCR and Information Extraction (SROIE) [Huang et al., 2019] dataset that was published for ICDAR 2019. We focus on the third task which consists in extracting information from the documents. SROIE contains Malaysian receipts split into 626 train and 347 test documents. Unfortunately, we do not have control over the composition of the test set, and most of the test layouts also occur in the training set.

We used the OCR text provided with the dataset instead of using our own OCR system. As others have pointed out [Xu et al., 2020], it contains numerous little errors that negatively affect the final performance. For a fair comparison with the leaderboard, we manually fixed them such that the expected string appears in the input, at least. These fixes mostly concern addresses and company names. It almost exclusively involves fixing errors related to white spaces and commas.

4.6 Experiments

All experiments were performed on a single machine equipped with two Nvidia RTX A6000 with 48GB of video memory each. This allowed us to boost the batch size up to 32 per device on a base transformer model. To further increase the batch size, we also aggregated 6 batches together before propagating the gradient for a total batch size of 192. We used the Adam optimizer with a learning rate of $1e - 5$

Post Processing	Dataset	
	SROIE	BDC-PO
Ad-Hoc	93.88 ± 0.59	84.35 ± 0.12
CRF	94.01 ± 0.55	84.40 ± 0.16
ConfOpt	94.94 ± 0.38	84.57 ± 0.10

Table 4.1: Performance comparison on SROIE [Huang et al., 2019] and BDC-PO between multiple post-processing algorithms. The score is computed on the exact match between the prediction and the target string.

and 200 linear warm-up steps as it improved our model’s convergence. We used 1500 training steps for SROIE and 3000 steps for BDC-PO. Finally, we ran each fine-tuning 10 times in each setup to get a precise idea of the performance of the models and the variability of the results. For the different pre-training scenarios, we performed only two runs and the best model was kept.

4.6.1 Post-Processing

This first set of experiments aims at comparing the post-processing used to decode the sequence produced by the model. We want to determine whether our proposed ConfOpt algorithm is competitive with other decoding methods. We decided to use the LayoutLM base model and compare the proposed ConfOpt against two other decoding algorithms as shown in Table 4.1.

We named Ad-Hoc the basic decoding using the label with maximal confidence for each token. When decoding with this method, a B tag starts a new entity, a I tag continues the previous entity, a E closes the previous entity, and a S tag produces a new entity and closes it right away. Ad-Hoc and ConfOpt use the same model weights in this experiment as they do not introduce any trainable parameters.

The second decoding algorithm uses a CRF [Lafferty et al., 2001; Lample et al., 2016] that processes LayoutLM’s predictions. In this particular case, we did not use the classical cross-entropy loss but the score provided by the CRF layer. Because the CRF required specific training and did not optimize the same loss, its weights are different from the two other post-processing methods.

We evaluated these algorithms on both SROIE and BDC-PO. The results in Table 4.1 show a tiny improvement using a CRF instead of the Ad-Hoc post-processing (0.13 and 0.05 F1 points) but those differences are always within one standard deviation range. We would need more evidence for a definitive answer on the effect of adding a CRF layer for the post-processing.

On both datasets, using ConfOpt significantly increases performance (1.06 and 0.22 F1 points) compared to the Ad-Hoc post-processing, even though the model is strictly identical. In light of these results, we decided to use the ConfOpt for the next experiment.

Pre Training		F1 Score	Accuracy per Field		
Task(s)	Dataset		PO Number	Total	Date
MVLM	RVL-CDIP	84.57 ± 0.10	89.98	89.10	93.59
MVLM	BDC	84.77 ± 0.12	90.61	89.33	93.59
MVLM+NO+LI	BDC	84.84 ± 0.08	90.71	89.36	93.83

Table 4.2: Model performance when fine-tuning on BDC-PO.

Pre Training		F1 Score	Accuracy per Field			
Task(s)	Dataset		Com-pany	Address	Total	Date
MVLM	RVL-CDIP	94.94 ± 0.38	92.91	90.81	89.25	99.48
MVLM	BDC	95.18 ± 0.23	93.72	91.00	89.48	99.68
MVLM+NO+LI	BDC	95.50 ± 0.22	93.60	91.41	90.89	99.57

Table 4.3: Model performance when fine-tuning on SROIE [Huang et al., 2019].

4.6.2 Business Document-Specific Pre-training

We conducted another set of experiments in order to study the effects of the new business data-specific pre-training tasks on the model performance. At the same time, we controlled the performance gap obtained by pre-training with the basic MVLM task on the same new dataset. Both comparisons are insightful to decide whether it is useful to pre-train on clients’ data and/or with data-specific pre-training tasks.

For the pre-training part, we always initialize the model’s weights with the base version [Xu et al., 2020]. We pre-train models for 20 epochs on 80% of BDC. When using multiple pre-training tasks at the same time, we chose to provide batches of single tasks to the model. Gradient aggregation over multiple batches helps with smoothing the update between different tasks. We pre-trained 2 models on the BDC, one with MVLM only and another with MVLM+NO+LI.

We evaluated each pre-trained model on both datasets, the results are available in Table 4.2 for BDC-PO and Table 4.3 for SROIE. Each cell contains the means of 10 runs with different seeds and the standard deviation is provided for the F1 score. There are a few interesting things to notice.

The first important remark is the importance of the pre-training dataset. Pre-training on BDC significantly improves performance on both SROIE and BDC-PO, even though the pretext training task is the same as what was used for LayoutLM. BDC is more homogeneous and focuses on invoices and purchase orders. Contrary to our expectations, we observe a greater improvement on SROIE than on BDC-PO (0.24 vs 0.2 F1 points). But the overall improvement by using BDC can be

Model Name	Size	Modalities	F1 Score
LayoutLM _{BASE} (Ours)	113M	T+L	95.50 ± 0.22
RoBERTa _{BASE} [Liu et al., 2019b]	125M	T	91.07
LayoutLM _{BASE} [Xu et al., 2020]	113M	T+L	94.38
LayoutLMv2 _{BASE} [Xu et al., 2021]	200M	T+L+I	96.25
LayoutMask _{BASE} [Tu et al., 2023]‡	182M	T+L	96.87
RoBERTa _{LARGE} [Liu et al., 2019b]	355M	T	92.80
LayoutLM _{LARGE} [Xu et al., 2020]	343M	T+L	95.24
LayoutLMv2 _{LARGE} [Xu et al., 2021]	426M	T+L+I	97.81
ERNIELayout [Peng et al., 2022]†	303M [†]	T+L+I	97.55
LayoutMask _{LARGE} [Tu et al., 2023]‡	404M	T+L	97.27

Table 4.4: Model performance comparison with the literature on SROIE. Model sizes are expressed in terms of their number of trainable parameters and modalities are text (T), layout (L) and image (I). Although our contribution is overthrown by larger and T+L+I models in terms of pure performance, it achieves better results than the original LayoutLM_{LARGE} [Xu et al., 2020] which is 3 times bigger.

(†) Because ERNIELayout [Peng et al., 2022] is not publicly available, we had to estimate its number of parameters with open-source reproductions.

(‡) Those models were published after the first publication of this work.

explained because RVL-CDIP contains a broader panel of document types and is not specialized like BDC. Even though BDC does not contain expense receipts, its global structure is similar to invoices.

Next, we can compare the pre-training tasks. Introducing NO and LI tasks also improves the performance over the previously pre-trained model. We observe a 0.32 F1 point improvement on SROIE but only 0.07 on BDC-PO. We suspect the small improvement introduced by the new tasks can be explained because most useful skills to process purchase orders were learned by pre-training on such documents. The new pre-training tasks help more for generalizing on new types of documents.

We also can look at the results on a field-per-field basis. We observe that using BDC over RVL-CDIP improved the recognition of all fields except for the dates in BDC-PO. If introducing new training tasks did not improve all fields, we notice that some fields were greatly enhanced like the total amount in SROIE (1.41 F1 points difference). We expected to observe a greater improvement in the total field with the new pre-training tasks. But it does not seem to improve performance much on BDC-PO’s total.

Finally, it is interesting to compare on Table 4.4 our results with the literature on the dataset. Our pre-trained model with NO and LI tasks performs better than LayoutLM large which contains 3 times more parameters. However, multimodal models including the visual modality such as LayoutLMv2 [Xu et al., 2021] and ERNIELayout [Peng et al., 2022] achieve performance still unreachable for a textual-

only model. The recent LayoutMask [Tu et al., 2023] reaches similar performance levels as T+L+I models and performs them for smaller models. The difference in the performance of LayoutMask with LayoutLM and our contribution seems to rely on their challenging pre-training and their representation of the input sequence. Further work would be necessary to determine if our proposed pre-training tasks would benefit to LayoutMask as they benefited LayoutLM.

4.7 Conclusion

In this Chapter, we showed significant improvements are accessible without introducing more trainable parameters and computational complexity. Only using the base transformer architecture, we achieved a performance that is comparable to the large version which contains 3 times more parameters. Pre-trained models can be further specialized through in-domain datasets and specific pretext training tasks. We demonstrated that by introducing a new collection of business documents and training tasks focusing on documents' layout and number understanding. We showed that performance improvements can be imputed to both pre-training tasks (NO and LI) and a new pre-training dataset.

Thanks to their pre-training procedure and their capacity to learn complex attention patterns, transformers have proven to be very efficient for document-understanding tasks. Even in a data-constrained environment (see Chapter 3) or with domain-specific languages, transformers achieve unmatched performance. They however suffer a significant drawback as the computation cost quickly grows with long sequences. It limits their application to short, single-page documents with a small number of words, ruling out a large number of potential candidates for BDs processing automation. In the following Chapter, we explore alternative architectures and strategies to include those left-out documents in our applications.

Long-Range Transformer Architectures for Document Understanding

Chapter abstract

Since their release, transformers have revolutionized many fields from Natural Language Processing (NLP) to Computer Vision (CV). Document Understanding (DU) was not left behind with the first transformer-based models for DU dating from late 2019. However, the computational complexity of the self-attention operation limits their capabilities to small sequences. In this paper, we explore multiple strategies to apply transformer-based models to long multipage documents. We introduce 2 new multimodal (text + layout) long-range models for DU. They are based on efficient implementations of transformers for long sequences. Long-range models can process whole documents at once effectively and are less impaired by the document's length. We compare them to LayoutLM, a classical transformer adapted for DU and pre-trained on millions of documents. We further propose a 2D relative attention bias to guide self-attention towards relevant tokens without harming model efficiency. We observe improvements on multipage business documents on Information Extraction (IE) for a small performance cost on smaller sequences. Relative 2D attention was revealed to be effective on dense text for both normal and long-range models.

This work has led to the publication of a paper and an oral presentation at an international workshop:

- Thibault Douzon, Stefan Duffner, Christophe Garcia, and Jérémy Espinas. Long-Range Transformer Architectures for Document Understanding. In *Proceedings of the First Workshop on Machine Vision and NLP for Document Analysis (VINALDO), ICDAR 2023*, San José, September 2023. Springer International Publishing.

Contents

5.1	Introduction	88
5.2	Related Work	89
5.2.1	From Natural Language Processing to Document Understanding	89
5.2.2	Long-Range Transformers	90
5.3	Datasets	92
5.3.1	Business Document Collection - Purchase Orders	92
5.3.2	DocBank	93
5.4	Models	93
5.4.1	LayoutLM	94
5.4.2	LayoutLinformer	95
5.4.3	LayoutCosformer	96
5.4.4	2D Relative attention	97
5.5	Experiments	99
5.5.1	Long-Range	99
5.5.2	Relative Attention	103
5.6	Conclusion	104

5.1 Introduction

In the previous chapters, we demonstrated how transformers significantly improved both data efficiency and performance thanks to their extensive pre-training and self-attention. However, the superior performance of transformers over other sequential models has a cost: contrary to Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), transformers’ computational cost scales quadratically instead of linearly (see Subsection 2.2.3). It limits our capacity to use models based on this architecture for long input sequences such as long texts or multipage documents.

Some Business Document (BD) types can greatly vary in size, for instance, a customer order may include a single line item of several hundred. A model that processes such documents should be able to tackle with maximum performance both short and long examples. Classical approaches involve RNNs combined with several text-reduction mechanisms [Wan et al., 2019]. However, because information propagates through the sequence, RNNs struggle at modeling long-range dependencies, even with Long Short Term Memory (LSTM) cells. The transformer’s self-attention can straightforwardly learn dependencies spanning across the sequence without any loss of information. It makes the transformer architecture a good candidate to learn the appropriate interactions between line items and header information in a BD.

In this Chapter, we explore several approaches and architectures in order to use transformer models on long documents. For simplicity, we limited our study to text and layout modalities, and chose to focus on document length to evaluate the model efficiency. We compare various encoder-only models on sequence tagging tasks with business and academic documents. We also study the impact of relative attention based on document layout instead of a linear token position, and its implementation for long-range transformers.

5.2 Related Work

5.2.1 From Natural Language Processing to Document Understanding

This work derives from both long-range transformers proposed in NLP tasks, trying to process longer sequences at once and Transformer architectures adapted to DU. Before the proposal of transformers, the *de facto* architecture for NLP has been RNNs. Multiple improvements have been proposed, for example, to tackle vanishing gradients like LSTM cells Hochreiter and Schmidhuber [1997]. Coupled with Conditional Random Fields (CRFs) [Lafferty et al., 2001; Lample et al., 2016], bidirectional LSTM encoders were then capable at most text understanding task Lample et al. [2016]. For more complex IE, where target information can span multiple tokens, BIESO tags allow better decoding by precisely locating the beginning and end of the information. Although long sequences can be processed with RNNs, longer input negatively affects the performance of encoder-decoder architectures Bahdanau et al. [2016]. Hence, the attention mechanism was quickly adopted for those architectures as an information highway between the encoder and the decoder. The transformer’s [Vaswani et al., 2017] self-attention generalizes the encoder-decoder attention introduced by Bahdanau et al. [2016] to an encoder or decoder-only model.

First developed for NLP, transformer encoders were quickly transposed to DU [Xu et al., 2020; Garncarek et al., 2021; Tu et al., 2023] as pre-trained models were able to leverage large document collections and outperformed all previous approaches. LayoutLM [Xu et al., 2020], for example, only introduced 2D positional embeddings over BERT [Devlin et al., 2019] and was pre-trained on the RVL-CDIP [Lewis et al., 2006] collection. It opened the way to many other models applying transformers to previous design [Katti et al., 2018; Denk and Reisswig, 2019], leveraging end-to-end capacities of encoder-decoder models [Powalski et al., 2021; Kim et al., 2022], or providing image and text to the models like a visual transformer [Dosovitskiy et al., 2021; Li et al., 2022]. Because the transformer’s output is independent of the sequence order, positional embeddings are classically added to the input. It is also possible to introduce relative bias to the self-attention mechanism to promote local interactions inside the self-attention. When processing Visually-Rich Documents (VRDs) with complex 2D layouts, the input is usually enriched with 2D position and relative bias to self-attention can be replaced with a 2D relative bias to take into account word positions in the document.

Most recent models for DU propose to leverage as much information as possible by using multiple modalities: text, layout and image. Either by combining CNNs with transformers [Xu et al., 2021; Powalski et al., 2021] or by mixing textual and visual sequences inside the transformer’s input Li et al. [2021b]; Huang et al. [2022]. Even though those approaches provide superior results, we chose to not include image modality in our architectures.

5.2.2 Long-Range Transformers

Since the introduction of BERT Devlin et al. [2019] and GPT Radford et al. [2018a], transformers have demonstrated their capacity to understand and model language Wang et al. [2020a]. Their ability to manipulate words can be visualized through the amount of attention each token allows to other tokens. However, as discussed in Subsection 2.2.3, dot-product attention computation involves a $O(N^2)$ time and memory complexity where N is the sequence length. It limits the capacity of transformer-based models in dealing with long sequences as they need too much Graphics Processing Unit (GPU) memory or take too long to process.

Instead of truncating the sequence to the maximum length, many modifications have been proposed to replace the attention layer with some efficient approximation that can be computed in $O(N)$ or $O(N \log(N))$. They have been developed and tested with NLP tasks where long sequences are most likely to be found like long text summarization and translation.

Fixed Attention Patterns

Some models use attention patterns to limit token attention to a fixed number of other tokens. It strictly enforces the locality of attention by sparsifying the similarity matrix, reducing the number of possible interactions between tokens and the complexity of self-attention. For example, splitting the sequence into multiple independent blocks like BlockBERT [Qiu et al., 2020a] is a simple and practical solution. However, because tokens at the edge of a block cannot attend to other blocks, this pattern produces blindspots for the model which hurts performance around those edges. Transformer-XL [Dai et al., 2019] solves this issue by introducing a recurrent connection between blocks. Instead of fixed blocks, sliding windows around each token remove this blindspot and allow each token to attend to its neighborhood. LongFormer [Beltagy et al., 2020] and BigBird [Zaheer et al., 2021] use a combination of sliding windows, global and random attention where the global and random attention allows the model to attend outside of its neighborhood and propagate efficient information through long distances across the sequence. ETC [Ainslie et al., 2020; Lee et al., 2022] splits the input into a small number of global tokens and local tokens. Global tokens can attend (and be attended) to every other token while local tokens are limited to their close neighborhood and global tokens. Overall, the computational complexity of models using attention patterns can be controlled by the proportion of accessible tokens. An architecture using sliding windows of size

K results in a self-attention complexity of $O(NK)$ which is linear in the sequence length N .

Learned Attention Patterns

Fixed attention pattern architectures rely on a human-defined locality definition which dictates beforehand which attention interactions are allowed independently of the input sequence. Architectures relying on learned attention patterns shift the locality definition inside the model which can use the content of the input sequence to decide whether 2 tokens should attend to each other or not. Models learn to assign tokens to different baskets, then tokens can only attend to other tokens inside the same basket. Reformer [Kitaev et al., 2020] uses learnable locality-sensitive hash functions to assign input tokens to their basket. Other proposed strategies involve discovering clusters with K-means like the RoutingTransformer [Roy et al., 2020] or meta-sorting networks like the SinkhornTransformer [Tay et al., 2020]. Because the model learns the attention pattern, those architectures might better adapt to tasks involving long-range dependencies.

Approximation of Attention

Previous architectures focused on restraining attention by limiting the number of attention links between tokens. The dot-product attention however remains identical to Equation 2.37, the similarity matrix is simply sparsified to reduce the cost of computation. Wang et al. [2020b] noticed the low-rank nature of the similarity matrix $\mathbf{Q}\mathbf{K}^\top$ in all their experiments, indicating the contained information can be compressed with little loss. In Linformer, they propose to compress along the input length dimension by projecting both queries and keys into a smaller, fixed length. It acts as a simplification of the sequence by learning to merge together tokens. Performer [Choromanski et al., 2021] and Linear Transformer [Katharopoulos et al., 2020] use kernel approximations to rewrite the similarity matrix as a product of feature matrices $\phi(\mathbf{Q})\phi(\mathbf{K}^\top)$ where $\phi(\mathbf{Q}) \in \mathbb{R}^{N \times D}$ and $\phi(\mathbf{K}) \in \mathbb{R}^{N \times D}$ in which N and D are respectively the sequence length and the feature dimension. It allows the reordering of Equation 2.37 into $\phi(\mathbf{Q})(\phi(\mathbf{K}^\top)\mathbf{V})$ which is more efficient for $N \gg D$. The recently proposed Cosformer [Qin et al., 2022] improves upon Katharopoulos et al. [2020] model by introducing relative attention bias to the attention. Because they never compute the similarity matrix, those models lack interpretability which was provided by self-attention.

Other architectures for efficient transformers have been developed and compared in several surveys [Tay et al., 2022a; Dong et al., 2023]. With the rise of Large Language Models (LLMs), huge efforts have been put into optimizing classical self-attention with close to no compromise on performance. FlashAttention [Dao et al., 2022; Dao, 2023] and multi-query attention [Shazeer, 2019] (instead of multi-head) reduce the memory requirements to improve long-range capabilities of classical transformers. However, long-range transformer architectures have not yet been used on DU tasks, mostly due to datasets not containing lengthy documents.

5.3 Datasets

We used 2 document datasets, where our choice was mainly made based on document length and the task itself. We wanted a NLP task that can be represented as sequence tagging in order to test the whole encoder with long inputs. Both datasets consist of English-only documents with close to perfect Optical Character Recognition (OCR) extraction. They provide word-level axis-aligned bounding boxes in the form that can be fed to the model as layout information. We use the OCR-provided order for the input sequence and do not further analyze documents to extract their structure.

5.3.1 Business Document Collection - Purchase Orders

The first dataset consists of purchase orders submitted to Esker’s platform between 2018 and 2021. This is a modification of Business Documents Collection (BDC)-Purchase Order (PO) from the previous chapter that includes multi-page documents and more fields to extract. Due to privacy concerns, these documents cannot be shared. It contains 80k documents that can be divided into 9000 different issuers with no more than 50 documents from the same issuer. Usually, an issuer only emits documents with the same template for convenience. About 55% of documents can be tokenized into a sequence of 512 tokens which fit into a classical transformer default maximum length. Only 5% of documents are longer than 2048 tokens, following a long tail of distribution. In order to evaluate the models’ generalization abilities, we split into train, validation and test sets such that templates in the test set have not been seen by the model during training.

The task consists of Information Extraction on multiple known classes: *document number*, *date*, *total amount*, *item ID numbers* and *item quantities*. Some information only appears once in the document (e.g., *document number*, *date* and *total amount*) while others are repeated for each line item in the business order. We call *header* fields those that occur only once and *table* fields others as they are most of the time structured in a table layout. There could be between 1 and 50 items present in any document, their number is not known in advance. Figure 5.1 shows the labeling of a multipage document. Even though header fields are sometimes repeated on each page, it is only labeled once to stay consistent across templates. Labels are provided at the word level based on manual customer document extraction. We also controlled labeling quality and rejected from the dataset documents with missing mandatory fields or with the wrong number of line items.

A superset of this dataset was used for pre-training models on business documents. It consists of 300k POs and 100k invoices from the same commercial platform. All documents were submitted and processed by the platform but later rejected due to labeling errors or bad habits. Fortunately, this does not impact the OCR quality and allows us to pre-train our models on a large collection of recent documents. We chose to use it for pre-training instead of RVL-CDIP [Harley et al., 2015] for the OCR quality difference.

JMART SALES ORDER

ORDER # 953487
DATE: JANUARY 15, 2019

JMART
1600 Boston Road
Springfield, MA 01129
401.346.9000

TO: IDES US Inc.
120 Lincoln Avenue
New York, NY 10019
Customer ID 300717

SHP: JMART
TO: 1600 Boston Road
Springfield, MA 01129
Customer ID 300717

SALES PERSON	JOB	SHIPPING METHOD	DELIVERY DATE	PAYMENT TERMS	DUPLICATE
Terry Schmidt	953487	OVERNIGHT SHIPPING	1/15/19	Due on receipt	

QTY	ITEM #	DESCRIPTION	UNIT PRICE	LINE TOTAL
1	0000	Light Bulb 40 Watt clear 220/235V	0000/carton	1314.00
4	0000	Light Bulb 40 Watt frosted 220/235V	0000/carton	1728.00
4	0000	Light Bulb 60 Watt frosted 220/235V	0000/carton	1728.00
3	0000	Light Bulb 60 Watt clear 220/235V	0000/carton	1505.24
2	0000	Light Bulb 80 Watt clear 220/235V	0000/carton	880.70
4	0000	Light Bulb 80 Watt frosted 220/235V	0000/carton	1812.84
2	0000	MAG DL 15P/PC	0000/PC	1710.28
4	0000	MAG DL 15P/PC	0000/PC	1710.28
12	0000	PK-100 Special carton High tech	0000/PC	172.80
12	0000	PK-100 Padlee 120 x 80 x 12.5 Type B	0000/PC	157.80
4	0000	PAQ Monitor, 20", Color	0000/PC	1200.00
2	0000	Monitor C 313 Personal Computer	0000/PC	2194.80
4	0000	Processor Pentium	0000/PC	2100.00
3	0000	SEC Multisync X115	0000/PC	3063.10
4	0000	PAQ Monitor, 20", Color	0000/PC	1200.00
2	0000	MAG DL 15P/PC	0000/PC	1710.28
4	0000	MAG DL 15P/PC	0000/PC	1710.28
TOTAL DISCOUNT				-
SUBTOTAL				9727.68
TAXES TAX				1801.81
TOTAL				11529.49

This is an offer to purchase. Written acceptance of the order or shipment signifies acceptance of the Terms and Conditions of Purchase printed on the back or otherwise available at www.jmart.com. Any additional or different terms proposed by Seller are rejected unless expressly agreed to in writing by the Buyer.

Figure 5.1: Sample pages with colored labels similar to those in the BDC-PO dataset. Both pages come from the same document, the first page is on the left and the last page is on the right. Some information is repeated across pages of a document.

5.3.2 DocBank

DocBank [Li et al., 2020] is a dataset containing 500k public research article pages. It contains English documents spanning various research fields. Documents were obtained on arXiv and were annotated with PDFPlumber¹, a Portable Document Format (PDF) parser that accurately extracts item bounding boxes. Li et al. [2020] provide both pixel and word-level annotations for CV and NLP models. The order of words is defined from top to bottom and left to right, except for multicolumn documents where whole columns are ordered left to right. In this work, we will only use textual information along the word 2D positions.

Docbank segmentation task contains 12 categories (e.g. *title*, *paragraph*, *figure* etc.) representing semantic parts of a research article. Because articles contain dense paragraphs, most pages are longer than 512 tokens once tokenized. In fact, only 11% of the test documents contain less than 512 tokens and 84% contain between 512 and 2048 tokens.

5.4 Models

We compared LayoutLM [Xu et al., 2020], a transformer for DU which is our baseline, with our long-range contributions LayoutLinformer and LayoutCosformer². They only differ by their implementation of self-attention: LayoutLM uses full

¹<https://github.com/jsvine/pdfplumber>

²Models implementation and weights available at <https://github.com/thibaultdouzon/long-range-document-transformer>

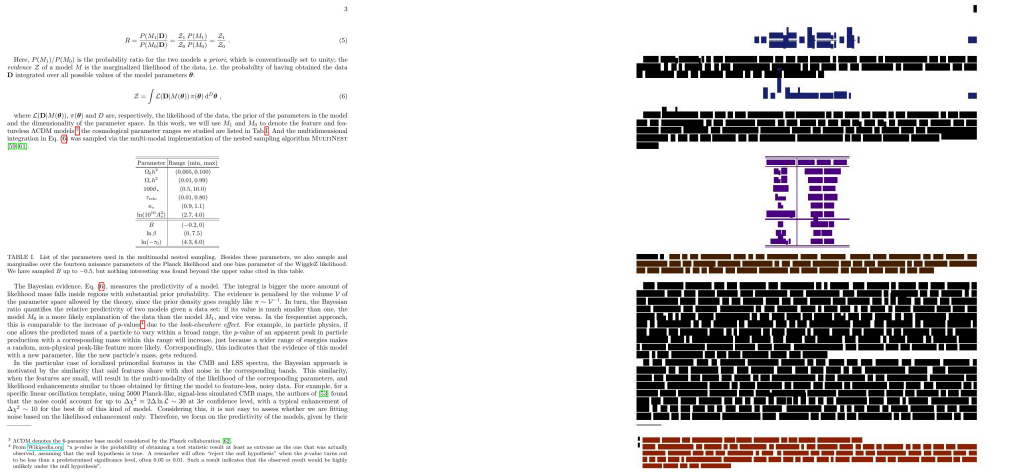


Figure 5.2: DocBank sample image on the left and its corresponding segmentation on the right. Each color represents one class (black for *paragraph*, purple for *equation*, ...).

self-attention like BERT [Devlin et al., 2019], LayoutLinformer uses a low-rank approximation first proposed by [Wang et al., 2020b] and LayoutCosformer uses a kernel-based method introduced in [Qin et al., 2022] as a replacement. We further detail how they work in the subsequent subsections.

We chose those models over other efficient transformers based on the convenience of adapting them from linear text to 2-dimensional documents. Efficient attention based on sliding windows [Beltagy et al., 2020; Zaheer et al., 2021] does not transpose nicely to 2D documents because the sliding window mechanism is deeply linked to the linear order of words. Even though our approach tries to provide words in a natural order, in some documents it does not reflect the human reading order. For example, when parsing table content, the reader often reads top to bottom to associate a column header with its values. To mitigate this issue, we preferred to rely on global attention or 2D local attention.

Similarly to how LayoutLM was adapted from BERT, we adapt Linformer and Cosformer models to process documents by adding a 2D positional embedding and a page embedding to the input. We chose to use learned embeddings to simplify weight transfer from LayoutLM to our long-range models.

5.4.1 LayoutLM

LayoutLM [Xu et al., 2020] has proven its capacities on most tasks related to documents since its release. It reuses BERT [Devlin et al., 2019] encoder and tokenizer but modifies the positional encoding by introducing a 2D encoding for word box boundaries and size. This modification allows the model to leverage layout information provided by the OCR. LayoutLM’s computational bottleneck is the self-

(5)

$$r = \frac{P(M_1|D) \cdot \frac{P_1(M_1)}{P_1(M_2)} \cdot \frac{Z_1}{Z_2}}{P(M_2|D) \cdot \frac{P_2(M_2)}{P_2(M_1)} \cdot \frac{Z_2}{Z_1}}$$

(6)

$$Z = \int \mathcal{L}(D|\theta) \pi(\theta) d^D \theta,$$

where $\mathcal{L}(D|\theta)$, $\pi(\theta)$ and D are, respectively, the likelihood of the data, the prior of the parameter in the model and the dimensionality of the parameter space. In this work, we will use M_1 and M_2 to denote the feature and frequency CRF models. The estimated parameter ranges we studied are listed in Table 5.1 and the reinforcement algorithm in Eq. 6 was sampled via the multi-modal implementation of the nested sampling algorithm `Nestle` [Fong 2018].

Parameter (Range (min, max))	
D_{CRF}	(0, 200, 0, 200)
D_{LF}	(0, 0, 0, 0)
γ_{CRF}	(0.1, 0.0)
γ_{LF}	(0.0, 0.0)
$\ln(\frac{P_1}{P_2})$	(0.9, 1.1)
$\ln(\frac{Z_1}{Z_2})$	(2.7, 4.0)
β	(-0.2, 0)
$\ln(\beta)$	(0, 2.5)
$\ln(-\beta)$	(-3.5, 4.0)

TABLE 1. List of the parameters used in the multimodal nested sampling. Besides these parameters, we also sample and optimize over the feature-usage parameters of the Black likelihood and one bias parameter of the Weight likelihood. We have sampled β up to -0.5 , but nothing interesting was found beyond the upper value cited in this table.

The Bayesian evidence, Eq. 6, measures the probability of a model. The integral is larger the more amount of likelihood mass falls inside regions with non-zero prior probability. The evidence is penalized for the volume β of the parameter space allowed by the theory since the prior density goes roughly like $\pi \sim \beta^{-D}$. In turn, the Bayesian ratio quantifies the relative predictive of two models given a data set. If its value is small smaller than one, the model M_2 is a more likely explanation of the data than the model M_1 , and vice versa. In the frequentist approach, this is comparable to the amount of positive χ^2 due to the best-likelihood effect. For example, in particle physics, if one allows the predicted mass of a particle to vary within a broad range, the position of an apparent peak in particle production with a corresponding mass within this range will increase, just because a wider range of energies makes a random, non-predicted peak-like feature more likely. Correspondingly, this indicates that the evidence of this model is a new parameter, like the one particle mass, gets reduced.

In the particular case of localised primordial features in the CRF and LFN spectra, the Bayesian approach is motivated by the intuition that such features share with what exists in the corresponding bands. This intuition, when the models are small, will result in the distribution of the likelihood of the corresponding parameters, and likelihood enhancements similar to those obtained by fitting the model to frequency, noisy data. For example, for a specific linear correlation template, using PyFRUIT the signal from simulated CRF maps, the authors of [Qin] found that the noise could account for up to $\Delta\chi^2 = 2\Delta \ln \mathcal{L} = 20$ at 3 σ confidence level, with a typical enhancement of $2\Delta \ln \mathcal{L} = 10$ for the best fit of this kind of model. Considering this, it is not easy to assess whether we are fitting noise based on the likelihood enhancement only. Therefore, we focus on the predictivity of the models, given by their

¹ `Nestle` denotes the β parameter has model considered by the Black likelihood [Fong].

² From [Wang2020b] “ β is the probability of obtaining a new statistic result at least as extreme as the one that was actually observed, assuming that the null hypothesis is true.” “ γ measures the ‘weight’ or ‘importance’ when the prior mass goes to be too low than a pre-determined significance level, often 0.05 or 0.01. Such a small indicates that the observed result would be highly unlikely under the null hypothesis.”

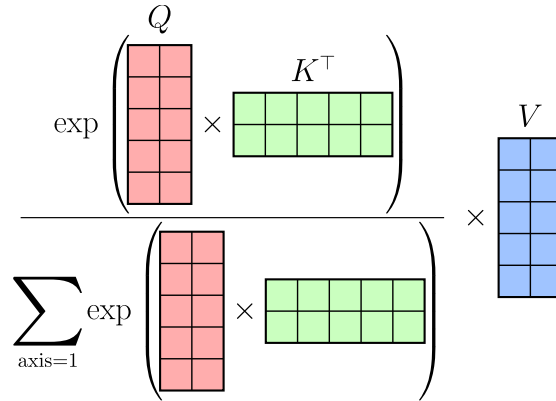


Figure 5.3: Illustration of the attention mechanism used in LayoutLM, normalization and multiple heads aside. In this example, $N = 5$ and $D = 2$. Due to the softmax operator, the product QK^T must be computed, resulting in $O(N^2)$ complexity.

attention layer. In transformers, self-attention is described by Equation 2.37. It can also be visually represented as in Figure 5.3 up to a normalization factor. Xu et al. [2020] pre-trained the model on RVL-CDIP [Harley et al., 2015] which contains 7 million scanned documents released in the 90s from the tobacco industry. Two versions of LayoutLM have been released: base and large, and they outperformed all preceding text-only language models on classification and IE tasks.

In our experiments, we only use the base model with maximum sequence length $N = 512$ and hidden size $d = 768$. For longer documents, we split the tokenized sequence into chunks of maximum length and process them separately.

5.4.2 LayoutLinformer

Our first contribution, LayoutLinformer, is based on the Linformer architecture [Wang et al., 2020b] and adapted to document processing by adding 2D positional encodings and using LayoutLM pre-trained weights. Although true self-attention can only be computed in $O(N^2)$, it can be approximated very efficiently by leveraging the low-rank property of the attention matrix QK^T . In Figure 5.4, we illustrate LayoutLinformer’s attention mechanism. Keys and values sequence length dimensions are projected on a smaller space of size M through a linear transformation: $K' = P_K K$ where $P_K \in \mathbb{R}^{M \times N}$ is the learned projection matrix (respectively $V' = P_V V$ where $P_V \in \mathbb{R}^{M \times N}$). This means the size of the new attention matrix $Q(P_K K)^T$ is $N \times M$, reducing the complexity of self-attention to $O(NM)$.

An immediate drawback of this projection is the loss of ability to visualize the attention matrix in order to explain the model. It is also no longer possible to implement causal attention or any specific attention pattern. On the other hand, Linformer provides a simple modification to the transformer to make it manage longer sequences with global attention. Most model weights are identical between the two architectures, allowing us to transfer LayoutLM pre-trained weights into

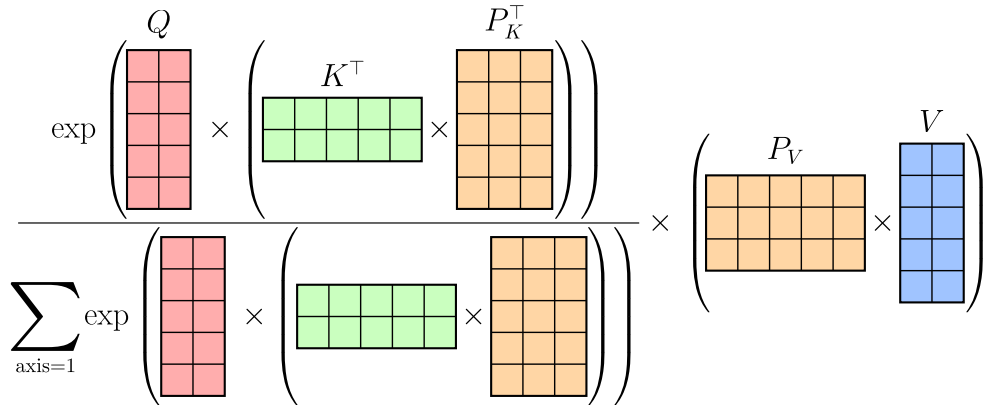


Figure 5.4: LayoutLinformer attention mechanism. In this example, $N = 5$, $D = 2$ and $M = 3$. Efficient matrix multiplication ordering reduces the complexity to $O(NM)$.

DocumentLinformer before further pre-training.

Wang et al. [2020b] showed that it can obtain a performance comparable to Roberta [Liu et al., 2019b] on multiple NLP benchmarks. They showed evidence that its performance is mostly determined by the projection dimension M and that increasing the sequence length N did not degrade results. Therefore, we chose to apply LayoutLinformer with $N = 2048$ and $M = 512$ to compare its performances with LayoutLM.

5.4.3 LayoutCosformer

Our second contribution, called LayoutCosformer, is based on the Cosformer [Qin et al., 2022] model which is another efficient alternative to the original transformer. Similarly to LayoutLinformer, we transferred pre-trained weights from LayoutLM to DocumentCosformer thanks to the similarities between architectures. It achieves linear complexity by replacing the non-linear similarity computation between Q and K with a linear operation. More specifically, Qin et al. [2022] proposed to replace $\exp(QK^T)$ with $\phi(Q)\phi(K)^T$ where ϕ is a nonlinear feature function. Figure 5.5 illustrates in more detail how LayoutCosformer attention works. In order to keep values of the similarity matrix positive, a good choice is $\phi = \text{ReLU}$. Computations can then be reordered to decrease the complexity to $O(N)$.

In addition to its linear self-attention complexity, Qin et al. [2022] include a relative self-attention bias towards nearby tokens. They cannot simply add the bias to the $N \times N$ similarity matrix before multiplying with values because it would mean a quadratic complexity. Their solution is to use functions that can be decomposed into a sum of products: $f(x, y) = \sum_n g_n(x) \times h_n(y)$. If we call B the bias matrix where $B_{i,j} = f(i, j)$, their biased similarity matrix can be written $\phi(Q)\phi(K^T) \odot B$ where \odot is the element-wise product. Then when looking at the attention from token i to token j we obtain:

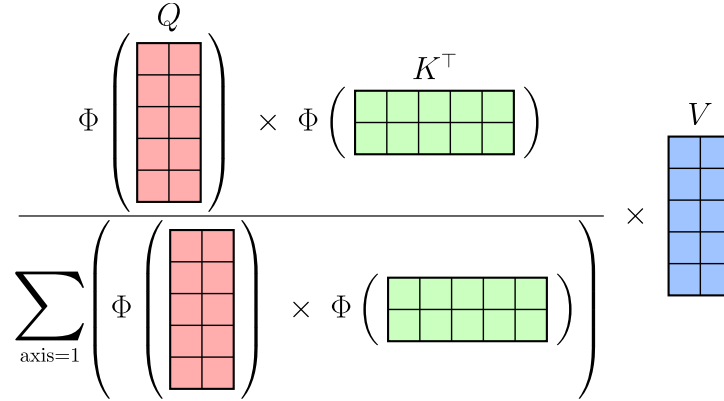


Figure 5.5: LayoutCosformer efficient attention mechanism with $N = 5$ and $D = 2$. The linear similarity enables computing first $\phi(\mathbf{K}^\top)\mathbf{V}$ and factorize $\phi(\mathbf{Q})$ out of the summation of the normalization factor.

$$\begin{aligned}
 \mathbf{s}_{i,j} &= \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j^\top)\mathbf{B}_{i,j} \\
 &= \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j^\top) \sum_n g_n(i) \times h_n(j) \\
 &= \sum_n \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j^\top)g_n(i)h_n(j) \\
 &= \sum_n (\phi(\mathbf{Q}_i)g_n(i)) \times (\phi(\mathbf{K}_j^\top)h_n(j))
 \end{aligned} \tag{5.1}$$

Using this trick, they proposed to use a cosine bias $\mathbf{B}_{i,j} = \cos(\frac{\pi}{2M}(i-j))$ which can be decomposed into $\mathbf{B}_{i,j} = \cos(\frac{\pi}{2M}i)\cos(\frac{\pi}{2M}j) + \sin(\frac{\pi}{2M}i)\sin(\frac{\pi}{2M}j)$. With the normalization constant M set to the maximum sequence length, they ensure $0 < \mathbf{B}_{i,j} < 1$ with a maximum when $i = j$. In the next subsection, we demonstrate how it can also be applied to 2D relative attention.

5.4.4 2D Relative attention

Global self-attention is a powerful tool for capturing long-range dependencies. However, although distant dependencies can be relevant, most attention should be on close neighbors. Relative attention [Shaw et al. \[2018\]](#); [Powalski et al. \[2021\]](#) selectively focuses on specific parts of the input by biasing the base self-attention. This was proven useful in text that can be represented as a linear sequence, but due to complex layouts, the sequence order is suboptimal to determine locality. To better capture the local context in documents, we introduced 2D relative attention based on the token positions inside the document.

In LayoutLM, we pre-compute for each document an attention bias matrix B and modify the self-attention formula to take it into account. More precisely, we use a modified self-attention, similarly to Equation 2.37:

$$\text{RelativeAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{B}) = \left(\text{softmax}(\mathbf{Q}\mathbf{K}^\top) \odot \mathbf{B} \right) \mathbf{V} \quad (5.2)$$

Where \odot denotes element-wise multiplication. Directly multiplying the attention matrix by some bias is very flexible and allows for any bias matrix to be chosen. It also matches the way LayoutCosformer applies a relative bias to its self-attention, thus allowing us to compare them.

On the other hand, it is nontrivial to implement relative attention for global long-range transformers. Because LayoutLinformer compresses the sequence dimension of \mathbf{K} , it is not possible to apply custom 2D attention bias to LayoutLinformer. For LayoutCosformer it is possible to reuse the same trick as in the 1D version with another bias function.

Because the function must remain separable into a sum of products, a good choice is to use exponentials and trigonometric functions. We first prove that the product of two separable functions is also itself separable. Let $f^1 = \sum_n g_n^1(x) \times h_n^1(y)$ and $f^2 = \sum_m g_m^2(x) \times h_m^2(y)$ be two functions separable into a sum of products, then:

$$\begin{aligned} f^1(x, y) \times f^2(x, y) &= \left(\sum_n g_n^1(x) \times h_n^1(y) \right) \times \left(\sum_m g_m^2(x) \times h_m^2(y) \right) \\ &= \sum_n \sum_m (g_n^1(x) \times h_n^1(y) \times g_m^2(x) \times h_m^2(y)) \\ &= \sum_{n,m} (g_n^1(x)g_m^2(x)) \times (h_n^1(y)h_m^2(y)) \end{aligned} \quad (5.3)$$

Which can also be separated into a sum of products.

We chose to compare 2 different attention biases. The first one is simply the product cosine bias along both X and Y axis. It captures the local context in every direction with variations close to the Euclidean distance. We define B^{squircle} ³ the following:

$$\mathbf{B}_{i,j}^{\text{squircle}} = \cos\left(\frac{\pi}{2M}(x_i - x_j)\right) \times \cos\left(\frac{\pi}{2M}(y_i - y_j)\right) \quad (5.4)$$

Where x_i and y_i (resp. x_j and y_j) are positions of token i (resp. j) along X and Y axis. In practice, we used the coordinates of the center of each token bounding box.

Although this bias correctly captures 2D locality, documents complex layout sometimes implicitly calls for other definitions of proximity to understand it. For instance, Figure 5.6 shows a table from a purchase order.

In this configuration, in order to grasp correctly the meaning of a cell in the table, the model needs to make the connection with the table header positioned at the

³Squircle are intermediate shape between square and circle, see <https://en.wikipedia.org/wiki/Squircle>. Contours of the surface described by B^{squircle} is not a squircle but also range from square to circle.

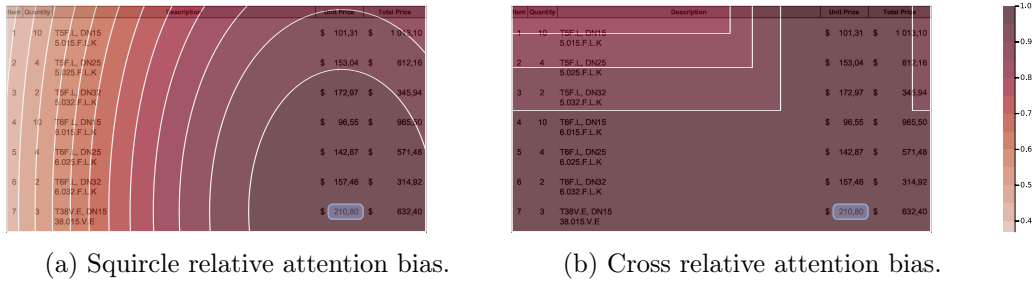


Figure 5.6: Contour plots for squirrel and cross relative attention bias applied to token “210,80” (bottom-right corner). Because token positions are normalized between 0 and 1000, tokens along the same line cannot fully attend to each other on the left while they are unaffected on the right.

beginning of the page. When multiple line items span the whole page, we hypothesize that this relative attention might hurt the performance due to the long-distance separating tokens. To deal with this issue, we propose another bias pattern. Its objective is to allow attention to tokens that are aligned with each other along the X or Y axis. To this end, we define $B_{i,j}^{\text{cross}} = \max\{\cos(\frac{\pi}{2M}(x_i - x_j)), \cos(\frac{\pi}{2M}(y_i - y_j))\}$. We illustrate the differences with an example shown in Figure 5.6. With cross-shaped relative attention bias, the highlighted token (the price of an item) can better attend to the column header “Unit Price” and its related line. In general, tokens inside a table can fully attend to their corresponding column header and line. This should prove helpful for understanding tables by guiding the model’s attention toward semantically related tokens.

5.5 Experiments

Our models are pre-trained on the BDC for 200k steps using Masked Visual Language Modeling (MVLN) Xu et al. [2020]. They are then finetuned on each dataset. For both tasks, we use BIESO tags to help the model decode predictions spanning multiple tokens. We performed our experiments on two RTX A6000 for pre-trainings and a single RTX A6000 for fine tunings. LayoutLM models run with a batch size of 48 and a sequence length of 512 while long-range models (LayoutLinformer and LayoutCosformer) can only get to a batch size of 16 with a sequence length of 2048 on a single device. We accumulate the gradient for 96 data samples before updating the model’s weights. We use Adam with learning rate $lr = 2 \cdot 10^{-5}$ and linear warmup for 5% of the training steps followed by a linear decrease.

5.5.1 Long-Range

Theoretical results on model architectures hint towards LayoutLinformer and LayoutCosformer being much more efficient the longer the sequence. We use a dummy inference task with increasing sequence lengths and compare our 2 models with

Model Name	Time (s) / Memory (GB)					
	Sequence Length					
	512	1024	2048	4096	8192	16384
LayoutLM [Xu et al., 2020]	1.41/1.25	2.83/2.50	7.39/5.01	23.43/13.69	-	-
LayoutLinformer	1.18/1.35	1.92/2.26	3.54/3.28	6.90/5.19	13.08/8.96	25.65/16.78
LayoutCosformer	2.03/1.36	2.50/2.37	4.68/3.38	9.00/5.38	17.23/9.59	33.96/17.59

Table 5.1: Duration and memory consumption of the 3 models for various sequence lengths on an inference task.

LayoutLM base architecture. The results are available in table 5.1. They reveal how the computational complexity of full self-attention disables LayoutLM when dealing with a sequence longer than 1024. Its memory consumption limits our tests with LayoutLM up to a sequence length of 4096, longer sequences couldn’t fit into a single GPU. On the other hand, LayoutLinformer and LayoutCosformer performed as predicted, with LayoutCosformer being slightly slower and more memory-hungry than LayoutLinformer.

It turns out the document’s length also greatly impacts model metrics performance on BDC-PO. For better visualization, we group documents into 3 length categories: short (document fits into 512 tokens), medium (between 513 and 2048) and long (2049 or more tokens). LayoutLM models can process short documents in a single sequence but need to split other documents into multiple independent sequences. Short and medium documents fit into LayoutLinformer and LayoutCosformer sequence lengths but not long documents. When a model cannot process a document in a single sequence, we split the document into multiple sequences and process them separately.

In Figure 5.7, we compare our pre-trained LayoutLM models with LayoutLinformer and LayoutCosformer. First, we discovered LayoutLM is very sensitive to the split position for medium and long documents. Introducing a sequence split when a new page is started greatly improves performance, we call this model LayoutLM SplitPage. It performs better on *total amount* (from 53.7% to 70%), *item ID number* (from 62.7% to 75.6%) and *quantity* (from 77.0% to 90.1%) recognition for medium and long documents. The repetitive structure of multipage documents combined with the fact that most pages fit in a 512 tokens sequence allow the model to not get lost. *Document number* and *date* are mostly not affected because they almost always occur at the beginning of the document, which is not affected by the splitting strategy.

Although LayoutLinformer and LayoutCosformer perform slightly worse than LayoutLM for short documents on all classes (around 74% F1 score on *item ID number* versus 81% for LayoutLMs), their performance decreases less than LayoutLM’s on medium documents. On those medium documents, even LayoutLM SplitPage drops from 88.2% to 70.1% F1 score on the *total amount* while both long-range

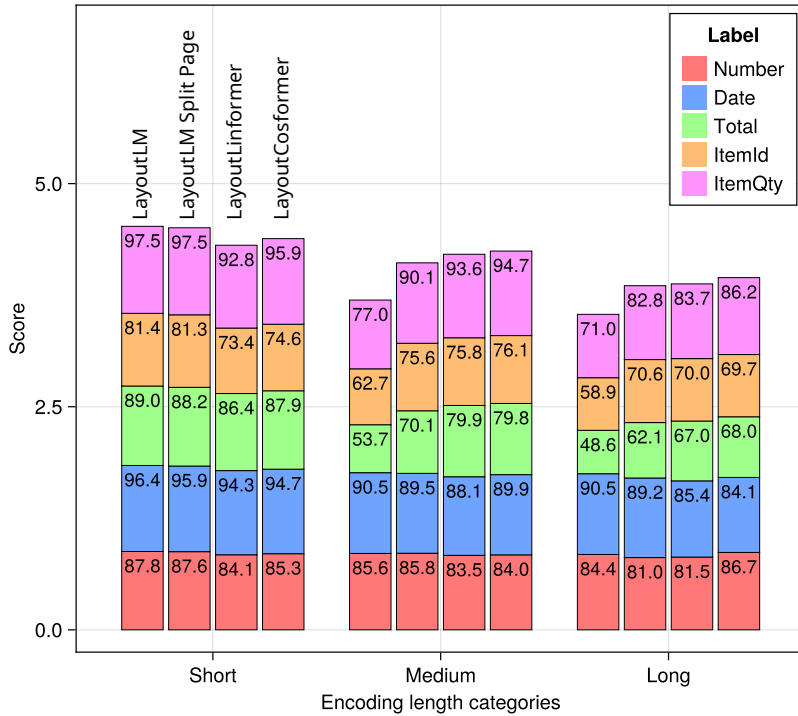


Figure 5.7: F1-score stacked bar plot of multiple models on BDC-PO. In each document length category, models are in the same order.

models only reduce performance from roughly 87% to 80%. We also noticed date recognition performance degrades across all models with longer documents which is not expected because *dates* are usually at the top of the first page. The same can be noted for the *order number* at a smaller scale. It might be due to a correlation between the document’s length and layout: short and medium/long documents do not share layouts. And because there are twice more short documents than longer ones, it is harder to generalize to new layouts. Overall, the performance of long-range models is more consistent across a wide variety of document lengths.

We performed the same experiments on the Docbank dataset, except for the page-splitting part as all documents are single-paged. At first, we compared the models’ performances for each document length category in Table 5.2. It contains the average F1 score across all labels weighted by the support of each label. It turns out length categories introduce bias in the composition of pages, with labels being very sparsely represented in some categories. This bias implicitly selects the first page of documents in shorter pages (with lower text density), while medium-sized pages contain a lot of paragraphs.

We observe the same drop in performance for long-range models on short documents, with LayoutLinformer providing better results across the board than LayoutCosformer. But we notice LayoutLM performs slightly better on medium documents than on short. Long-range models follow the same pattern with a greater difference between short and medium pages, LayoutCosformer almost gaining 2 average F1 percentage points. There are almost 20 times fewer long documents than medium,

Model Name	F1 Score		
	Document Length		
	Short	Medium	Long
LayoutLM	95.36	95.84	91.42
LayoutLinformer	95.20	96.49	91.41
LayoutCosformer	94.03	95.91	91.40

Table 5.2: F1 weighted average for each model and document length categories. All models were first pre-trained on BDC.

Model Name	F1 Score										Macro Average
	Categories										
	Abst.	Auth.	Capt.	Equa.	Footer	List	Para.	Sect.	Table	Title	
LayoutLM ([Li et al., 2020])	98.1	85.9	95.9	89.4	89.5	89.4	97.8	95.9	86.3	95.7	93.1
LayoutLM ([Xu et al., 2020])	98.3	89.6	96.0	89.0	91.6	88.2	97.5	94.3	87.4	90.4	93.0
LayoutLM †	97.8	87.5	94.9	87.2	90.5	84.0	97.1	92.8	85.7	88.6	91.6
LayoutLM _{SQUIRCLE} †	98.4	90.2	96.1	89.7	92.0	88.9	97.6	94.6	87.7	90.3	93.2
LayoutLM _{CROSS} †	98.4	90.3	96.0	89.6	92.1	88.7	97.6	94.6	87.5	90.7	93.2
LayoutLinformer †	97.9	88.9	93.7	90.0	91.1	87.9	97.5	91.3	87.6	88.7	92.3
LayoutCosformer †	97.2	87.2	91.0	88.1	90.6	87.4	97.1	81.4	87.0	88.3	90.7
LayoutCosformer _{SQUIRCLE} †	97.0	85.4	92.4	89.2	90.7	84.2	97.2	85.6	87.9	86.8	90.7
LayoutCosformer _{CROSS} †	97.4	86.9	93.8	91.2	91.7	87.5	97.5	87.4	89.0	88.1	91.9

Table 5.3: Results on Docbank dataset for LayoutLMs and long-range models. † Those models were pre-trained on BDC before finetuning on Docbank.

which could explain part of the global performance loss. Unfortunately, due to those biases, it is difficult to conclude on the models’ performances.

Table 5.3 compiles results for LayoutLM and long-range models for all labels. First, we can make sure our training pipeline performs on par with what Docbank authors reported for the LayoutLM base model by comparing their results and the ones we obtained by using public LayoutLM weights. Except for *author* and *title* labels, both results are very close, and the macro average is almost identical. Secondly, pre-training on business documents negatively impacts LayoutLM performances on all labels, losing 1.4 F1 percentage points on average. This advocates for pre-training data crucial role in later model finetuning results and its composition. Finally, long-range models performed on the same level as LayoutLM. LayoutLinformer even being more performant than our pre-trained LayoutLM. Overall, even though LayoutCosformer seems less performant on this task, both long-range mod-

Model Name	Macro Average F1 Score		
	Short	Medium	Long
LayoutLM Split Page	90.0	82.2	77.2
LayoutLM _{SQUIRCLE} Split Page	90.4	83.0	77.8
LayoutLM _{CROSS} Split Page	90.0	82.0	77.6
LayoutCosformer	87.6	85.0	79.0
LayoutCosformer _{SQUIRCLE}	85.8	82.2	73.4
LayoutCosformer _{CROSS}	87.6	85.2	77.2

Table 5.4: Macro average F1 score on the Business Orders dataset with 2D relative attention.

els performed better than our pre-trained LayoutLM on *table* and *equation*. Those two labels might benefit from long-range references, giving the model hints of their presence in the current sequence.

5.5.2 Relative Attention

We conduct the same experiments on models with 2D relative attention and compare their performance with their flat attention counterparts. On the BDC-PO dataset, Table 5.4 shows slight gains when using squircle attention with LayoutLM. For all document lengths, IE is improved by a few percentage points of F1 score over our previous LayoutLM Split Page implementation. However, we do not observe the same improvement with the cross-shaped attention pattern. This might indicate focusing on very local neighbors helps LayoutLM make the right decision. Overall, relative attention improves results in some circumstances but not as much as splitting every page did. However, when combined with LayoutCosformer, we observe a significant degradation in performance for all labels with the squircle attention while the cross pattern provides similar results as the raw LayoutCosformer.

On Docbank task, relative attention provides noticeable performance gains for both LayoutLM and LayoutCosformer. We provide all results in Table 5.3. LayoutLM with relative attention is standing out, going from 91.6% F1 score to 93.2% for both squircle and cross patterns. Most improvements are made on *author*, *equation* and *list*, each gaining at least 2 F1 score points. Both resulting models even beat Docbank’s authors’ version by a thin margin. This is impressive knowing those models were pre-trained on the same BDC-PO dataset as our base LayoutLM which suffered a 1.5 F1 score performance drop as a consequence. It turns out *author*, *equation* and *list* were also the fields where our LayoutLM performance dropped the most compared to stock LayoutLM. Applying cross-shaped relative attention to LayoutCosformer also improves performance across most labels. It even outperforms all other models on *equation* and *table* fields which benefit most from very long attention.

5.6 Conclusion

In this Chapter, we showed the impact of document length on transformer-based models applied to DU. Depending on the document’s type and the task, the model’s performance on longer documents can be negatively impacted with the F1 score dropping 20% for the most impacted. We explored several alternatives including another sequence split strategy and long-range layout-aware models based on Linformer and Cosformer architectures. They all proved to successfully reduce the performance gap between short and long documents (down to only a 10% performance drop), sometimes at a small cost on short document metrics. We also introduce relative attention based on 2D textual layout instead of the classical sequence order. It produces better results on dense text, significantly improving both LayoutLM and LayoutCosformer on the Docbank layout segmentation task.

In addition to other efficient transformer architectures, we plan to investigate other ways to use longer sequences for DU. For example, in multimodal models, this may allow the fitting of the whole text and visual patches of a document in a single sequence without needing more computing capabilities.

Conclusion

Contents

6.1 Contributions	105
6.2 Future Work and Perspectives	106

Throughout this thesis, we’ve shown the evolution of tools and algorithms available when performing any task related to document understanding. From hand-crafted rules and regular expressions to models able to discover and learn complex patterns when provided with labeled data examples. The last decade revealed the potential of models based on Artificial Neural Networks (ANNs) at the origin of the deep learning revolution. On several occasions, researchers working on document understanding adapted architectures first proposed for Natural Language Processing (NLP) tasks to the world of documents: first with Recurrent Neural Networks (RNNs) and Bidirectional Long Short Term Memorys (BiLSTMs), and more recently with models based on the transformer architecture in combination with extensive pre-training.

From the point of view of a company like Esker with business applications in mind, those technologies enable the progressive automation of the document processing pipeline. The choice of technology is driven by multiple factors like the performance at a given task which is directly correlated with the satisfaction of the client. Other factors such as the development and running costs, the time to market and the data requirements should not be overlooked as they also play a role in the adoption of a new feature. If statistical models provide better performances than rule-based algorithms, they require labeled datasets and skilled engineers to train and push models to production. Our contributions focus on the recent transformer architecture and its application to Visually-Rich Document (VRD) understanding.

6.1 Contributions

Data-efficiency of transformers

In Chapter 3, we studied how pre-training affects the amount of labeled data required to train a model. By comparing a pre-trained model to freshly initialized models, we demonstrated how pre-trained versions were significantly more data-efficient than their newly initialized counterparts on a sequence labeling task. The model pre-trained with Masked Language Modeling (MLM) was able to reach close to maximal

performance with a fraction of the training dataset, outperforming other models with 10 times fewer examples. We also showed public datasets could be leveraged to further improve a model on a downstream task with a very limited amount of labeled examples. This work strongly advocates the use of pre-trained models, both for their performance and their efficiency. They drastically reduce the entry barrier of deep learning models in terms of the required size of the training dataset, which greatly reduces the cost of finetuning a model.

Specific pre-training for business documents

Chapter 4 shifts the focus toward pre-training tasks for Business Document (BD) understanding. To pre-train a model specialized for BD, we first collected a large collection of BDs. However the distribution of BDs is quite different from general VRDs, BDs include several numeric values across pages and their content is structured by tables and blocks instead of long paragraphs of text. To guide the model learning better representations of BDs, we introduced new pre-training tasks inspired by MLM directed toward better numeric values and layout understanding. The resulting pre-trained model significantly outperformed comparable models in the literature and challenged the performances of models 3 times bigger. This work incentivizes to spend more time designing adequate pre-training tasks for the data distribution as it significantly helps the model when finetuning. Besides, as long as pre-training is performed with self-supervision, it also reduces the need for labeled examples during finetuning for the same performance.

Processing long documents

In the last Chapter, we evaluated several modified transformer architectures aimed at processing longer sequences. Because the original transformer uses self-attention with quadratic complexity with respect to the sequence length, transformer-based models are usually not able to process long sequences without driving costs high. Our baseline strategy was to split the sequence when reaching the size limit or the end of the page. We proposed to adapt to VRD understanding known long-range alternatives to self-attention and comparing their performance. Those alternatives compared favorably to the baseline when dealing with long documents but suffered from the self-attention replacement for short documents.

6.2 Future Work and Perspectives

The following section is a short discussion on the future of document understanding.

Integrating the workflow into the model

Currently, models are deeply integrated into a workflow that first pre-processes the image to extract words with an Optical Character Recognition (OCR) and post-processes the results afterward. Post-processing BIESO tags can involve complex

steps to take into account field occurrences spanning over multiple tokens as seen in Chapter 4. One perspective would be to reduce the model’s dependence on such systems by actually outputting structured, parsed results.

End-to-end models like TILT [Powalski et al., 2021] and Donut [Kim et al., 2022] have demonstrated the importance of a decoder network to generate a textual output. It allows the model to tackle a broader range of Document Understanding (DU) tasks, including Question Answering (QA). Kim et al. [2022] even resorted to integrating the OCR system into the model to improve overall speed.

Multimodality

The transformer architecture has proven its performance with multiple modalities for VRD understanding. The addition of the visual modality significantly improves metrics provided that the model learns how to combine information from several modalities together. Those interactions are currently learned during the pre-training of the model through the pre-training tasks. As shown by Tay et al. [2022b] with UL2 and LayoutMask [Tu et al., 2023], difficult pre-training tasks benefit the model as it learns more complex patterns. This might inspire future multimodal pre-training task designs that will hopefully better align the modalities provided to the model.

On the other hand, multimodal architectures such as StrucTexT [Li et al., 2021c], LayoutLMv3 [Huang et al., 2022] and ERNIELayout [Peng et al., 2022] use multiple sequences to encode separately the different modalities. It substantially lengthens the input sequence which is already heavily constrained by the quadratic complexity cost associated with self-attention. Efficient transformer architectures might be able to fully unlock the potential of multimodal transformers by processing at the same time a complete page of dense tokens with highly detailed visual patches.

Large language models

The recent breakthrough of Large Language Models (LLMs) in language generation [Touvron et al., 2023] demonstrated their ability to understand text, follow instructions and produce an answer. They however rely on text only although some have announced the release in the future of multimodal LLMs. Such a model with a long enough context sequence length could drastically improve on current standards. Most extreme models have already extended the context length to 128k tokens [Peng et al., 2023] thanks to FlashAttention [Dao et al., 2022]. Their ability to use tools and reason through a chain of thoughts [Wei et al., 2023] enables more complex tasks like browsing a web portal to search for information and submit previously extracted document details. LLMs have the potential to automate a significant number of tasks by connecting systems that were not able to interact previously.

However, because of their size, LLMs are incredibly expensive to train and use for inference. The challenge is to reduce the costs associated with them by reducing their size through quantization [Dettmers et al., 2022], distillation [Sanh et al., 2020] or a better training procedure.

Bibliography

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. (Cited on page 28.)
- Milan Aggarwal, Hires Gupta, Mausoom Sarkar, and Balaji Krishnamurthy. Form2Seq: A framework for higher-order form structure extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3830–3840, 2020. (Cited on page 40.)
- Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding Long and Structured Inputs in Transformers. *arXiv:2004.08483 [cs, stat]*, October 2020. (Cited on pages 46 and 90.)
- Srikanth Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. DocFormer: End-to-End Transformer for Document Understanding, September 2021. (Cited on pages xv, 54, 55 and 74.)
- Douglas E Appelt, Jerry R Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS: A Finite-state Processor for Information Extraction from Real-world Text. 1993. (Cited on page 15.)
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. (Cited on pages 32 and 44.)
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate, May 2016. (Cited on pages xiv, 40, 41, 42, 43 and 89.)
- Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training, February 2020. (Cited on page 49.)
- Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers, September 2022. (Cited on page 55.)

- Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6): 1554–1563, 1966. ISSN 0003-4851. (Cited on page 15.)
- Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]*, December 2020. (Cited on pages 46, 61, 90 and 94.)
- 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 28.)
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March 2003. ISSN 1532-4435. (Cited on page 38.)
- Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep Learning*, volume 1. MIT press Massachusetts, USA:, 2017. (Cited on pages 19 and 20.)
- James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, and Arnaud Bergeron. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011. (Cited on page 28.)
- Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. (Cited on page 20.)
- John S. Bridle. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In Françoise Fogelman Soulié and Jeanny Hérault, editors, *Neurocomputing*, NATO ASI Series, pages 227–236, Berlin, Heidelberg, 1990. Springer. ISBN 978-3-642-76153-9. doi: 10.1007/978-3-642-76153-9_28. (Cited on page 27.)
- Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks?, January 2022. (Cited on page 53.)
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. arXiv, July 2020. (Cited on pages 48, 50, 59 and 60.)

- Arthur Bryson, Y.-C Ho, and George Siouris. Applied Optimal Control: Optimization, Estimation, and Control. *Systems, Man and Cybernetics, IEEE Transactions on*, 9:366–367, July 1979. doi: 10.1109/TSMC.1979.4310229. (Cited on page 27.)
- Yungcheol Byun and Yillbyung Lee. Form classification using DP matching. In *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1, SAC '00*, pages 1–4, New York, NY, USA, March 2000. Association for Computing Machinery. ISBN 978-1-58113-240-3. doi: 10.1145/335603.335611. (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 59.)
- Francesca Cesarini, Enrico Francesconi, Marco Gori, and Giovanni Soda. Analysis and understanding of multi-class invoices. *International Journal on Document Analysis and Recognition*, 6(2):102–114, October 2003. ISSN 1433-2833, 1433-2825. doi: 10.1007/s10032-002-0084-6. (Cited on page 15.)
- Nawei Chen and Dorothea Blostein. A survey of document image classification: Problem statement, classifier architecture and performance evaluation. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(1):1–16, June 2007. ISSN 1433-2833, 1433-2825. doi: 10.1007/s10032-006-0020-2. (Cited on page 15.)
- Zhiyu Chen, Harini Eavani, Wenhui 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. (Cited on page 59.)
- Zhanzhan Cheng, Peng Zhang, Can Li, Qiao Liang, Yunlu Xu, Pengfei Li, Shiliang Pu, Yi Niu, and Fei Wu. TRIE++: Towards End-to-End Information Extraction from Visually Rich Documents, July 2022. (Cited on pages 3 and 42.)
- Nancy A. Chinchor. Overview of MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, 1998. (Cited on page 15.)
- Laura Chiticariu, Yunyao Li, and Frederick R. 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, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. (Cited on pages 15 and 59.)
- 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, September 2014. (Cited on pages 36 and 40.)
- Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains, April 2020. (Cited on page 34.)
- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking Attention with Performers. *arXiv:2009.14794 [cs, stat]*, March 2021. (Cited on page 91.)
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways, October 2022. (Cited on pages xv, 50 and 56.)
- Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, January 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00448. (Cited on page 46.)
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv:2003.10555 [cs]*, March 2020. (Cited on page 49.)
- Bob Cohen and Andrew Bartolini. ArdentPartners-AP-MTM2023-Basware, 2023. (Cited on page 2.)
- 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, ICML '08*, pages 160–167, New York, NY, USA, July 2008. Association for Computing Machinery. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. (Cited on page 38.)

- Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks, June 2016. (Cited on page 54.)
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, June 2019. (Cited on page 90.)
- Tuan Anh Nguyen Dang and Dat Nguyen Thanh. End-to-end information extraction by character-level embedding and multi-stage attentional U-net. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 96. BMVA Press, 2019. (Cited on page 60.)
- Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning, July 2023. (Cited on page 91.)
- Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, June 2022. (Cited on pages 45, 91 and 107.)
- Brian Davis, Bryan Morse, Scott Cohen, Brian Price, and Chris Tensmeyer. Deep Visual Template-Free Form Parsing, September 2019. (Cited on page 32.)
- Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. ISSN 1097-4571. doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASII>3.0.CO;2-9. (Cited on page 38.)
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848. (Cited on pages 28, 32, 34 and 47.)
- Andreas R. Dengel and Bertin Klein. smartFIX: A Requirements-Driven System for Document Analysis and Understanding. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Daniel Lopresti, Jianying Hu, and Ramanujan Kashi, editors, *Document Analysis Systems V*, volume 2423, pages 433–444. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN 978-3-540-44068-0 978-3-540-45869-2. doi: 10.1007/3-540-45869-7_47. (Cited on page 15.)
- Timo I. Denk and Christian Reisswig. BERTgrid: Contextualized Embedding for 2D Document Representation and Understanding. *arXiv:1909.04948 [cs]*, October 2019. (Cited on pages 52, 58, 60, 73 and 89.)
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, November 2022. (Cited on pages 45 and 107.)

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. (Cited on pages xv, 48, 49, 50, 51, 58, 60, 74, 75, 77, 89, 90 and 94.)
- Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified Language Model Pre-training for Natural Language Understanding and Generation, October 2019. (Cited on page 49.)
- Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. A Survey on Long Text Modeling with Transformers, February 2023. (Cited on page 91.)
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]*, June 2021. (Cited on pages 55 and 89.)
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. 2011. (Cited on page 21.)
- V. Egin and S. Bres. Document page similarity based on layout visual saliency: Application to query by example and document classification. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 1208–1212, August 2003. doi: 10.1109/ICDAR.2003.1227849. (Cited on page 15.)
- Floriana Esposito, Donato Malerba, and Francesca A. Lisi. Machine Learning for Intelligent Processing of Printed Documents. *Journal of Intelligent Information Systems*, 14(2):175–198, March 2000. ISSN 1573-7675. doi: 10.1023/A:1008735902918. (Cited on page 15.)
- Philip Gage. A New Algorithm for Data Compression. *The C User Journal*, 1994. doi: DOI10.5555/177910.177914. (Cited on page 37.)
- 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, Virtual Event CA USA, September 2020. ACM. ISBN 978-1-4503-8000-3. doi: 10.1145/3395027.3419584. (Cited on pages 53, 60 and 73.)
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, December 2020. (Cited on page 50.)

- Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. Question Answering is a Format; When is it Useful?, September 2019. (Cited on pages 13, 40 and 74.)
- Lukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. LAMBERT: Layout-Aware (Language) Modeling for information extraction. volume 12821, pages 532–547. 2021. doi: 10.1007/978-3-030-86549-8_34. (Cited on pages 3, 47, 52, 74 and 89.)
- Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, Sebastopol, CA, 2017. ISBN 978-1-4919-6229-9. (Cited on pages xiii and 30.)
- F.A. Gers and J. 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 vol.3, July 2000. doi: 10.1109/IJCNN.2000.861302. (Cited on page 36.)
- Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry, June 2017. (Cited on pages xv and 53.)
- 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 62.)
- Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Journal of Machine Learning Research*, volume 15, January 2010. (Cited on page 27.)
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks, June 2014. (Cited on page 33.)
- M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, July 2005. doi: 10.1109/IJCNN.2005.1555942. (Cited on page 53.)
- Alex Graves. Sequence Transduction with Recurrent Neural Networks, November 2012. (Cited on pages xiv and 41.)
- Alex Graves and Jürgen Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. (Cited on page 28.)

- Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Olomouc, Czech Republic, December 2013. IEEE. ISBN 978-1-4799-2756-2. doi: 10.1109/ASRU.2013.6707742. (Cited on page 28.)
- Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval. February 2015. (Cited on pages xi, 10, 31, 32, 34, 59, 92 and 95.)
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. (Cited on pages 33, 43 and 54.)
- Benjamin Heinzlerling and Michael Strube. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). (Cited on page 58.)
- Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), July 2020. (Cited on page 27.)
- Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude, 2013. (Cited on page 21.)
- Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. (Cited on pages 35, 36 and 89.)
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models, March 2022. (Cited on page 50.)
- Steven C.H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, October 2021. ISSN 09252312. doi: 10.1016/j.neucom.2021.04.112. (Cited on page 11.)
- Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. BROS: A Layout-Aware Pre-trained Language Model for Understanding Documents. *arXiv:2108.04539 [cs]*, August 2021. (Cited on pages 54, 58, 60 and 61.)

- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. (Cited on page 26.)
- Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. May 2018. (Cited on pages 48, 59 and 65.)
- Jiaying Hu, Ramanujan Kashi, and G. Wilfong. Document classification using layout analysis. pages 556–560, February 1999. ISBN 978-0-7695-0281-6. doi: 10.1109/DEXA.1999.795245. (Cited on page 15.)
- Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. April 2022. (Cited on pages xv, 56, 74, 90 and 107.)
- Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shjian Lu, and C. V. Jawahar. ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520, September 2019. doi: 10.1109/ICDAR.2019.00244. (Cited on pages xvi, xix, 12, 59, 61, 64, 66, 81, 82 and 83.)
- Wonseok Hwang, Jinyeong Yim, Seunghyun Park, Sohee Yang, and Minjoon Seo. Spatial Dependency Parsing for Semi-Structured Document Information Extraction, July 2021. (Cited on page 54.)
- Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, March 2015. (Cited on page 32.)
- Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. FUNSD: A dataset for form understanding in noisy scanned documents. In *2nd International Workshop on Open Services and Tools for Document Analysis, OST@ICDAR 2019, Sydney, Australia, September 22-25, 2019*, pages 1–6. IEEE, 2019. doi: 10.1109/ICDARW.2019.10029. (Cited on pages 12, 59 and 61.)
- Jian Liang, D. DeMenthon, and D. Doermann. Flattening Curved Documents in Images. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 338–345, San Diego, CA, USA, 2005. IEEE. ISBN 978-0-7695-2372-9. doi: 10.1109/CVPR.2005.163. (Cited on page 7.)
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving Pre-training by Representing and Predicting Spans, January 2020. (Cited on page 49.)

- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2342–2350, Lille, France, July 2015. PMLR. (Cited on page 36.)
- John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. (Cited on page 29.)
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. (Cited on page 50.)
- Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *arXiv:2006.16236 [cs, stat]*, August 2020. (Cited on pages 46 and 91.)
- Anoop Raveendra Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. Chargrid: Towards Understanding 2D Documents. In *arXiv:1809.08799 [Cs]*, September 2018. (Cited on pages 13, 33, 52, 60, 73, 74 and 89.)
- Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The Impact of Positional Encoding on Length Generalization in Transformers, May 2023. (Cited on page 47.)
- Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. OCR-free Document Understanding Transformer, October 2022. (Cited on pages 56, 74, 89 and 107.)
- Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. (Cited on page 21.)
- Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks, February 2017. (Cited on page 53.)
- David G. Kirkpatrick and John D. Radke. A Framework for Computational Morphology. In Godfried T. Toussaint, editor, *Machine Intelligence and Pattern*

- Recognition*, volume 2 of *Computational Geometry*, pages 217–248. North-Holland, January 1985. doi: 10.1016/B978-0-444-87806-9.50013-X. (Cited on page 53.)
- Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. *arXiv:2001.04451 [cs, stat]*, February 2020. (Cited on pages 46 and 91.)
- Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The Stack: 3 TB of permissively licensed source code, November 2022. (Cited on page 50.)
- Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning, May 2020. (Cited on page 34.)
- Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, April 2018. (Cited on page 37.)
- Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, August 2018. (Cited on page 37.)
- 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, June 2001. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-778-1. (Cited on pages 15, 73, 82 and 89.)
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural Architectures for Named Entity Recognition, April 2016. (Cited on pages 39, 73, 82 and 89.)
- Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gerard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Alexandra Luccioni, and Yacine Jernite. The BigScience ROOTS Corpus: A 1.6TB Composite Multilingual Dataset, March 2023. (Cited on page 50.)

- Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219. doi: 10.1109/5.726791. (Cited on pages 28 and 31.)
- Yann Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. ISSN 0899-7667. (Cited on pages 28 and 29.)
- Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas Pfister. FormNet: Structural Encoding beyond Sequential Modeling in Form Document Information Extraction, March 2022. (Cited on pages xv, 47, 52, 53 and 90.)
- Chen-Yu Lee, Chun-Liang Li, Hao Zhang, Timothy Dozat, Vincent Perot, Guolong Su, Xiang Zhang, Kihyuk Sohn, Nikolai Glushnev, Renshen Wang, Joshua Ainslie, Shangbang Long, Siyang Qin, Yasuhisa Fujii, Nan Hua, and Tomas Pfister. FormNetV2: Multimodal Graph Contrastive Learning for Form Document Information Extraction, May 2023. (Cited on pages 53, 54 and 74.)
- 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, Seattle Washington USA, August 2006. ACM. ISBN 978-1-59593-369-0. doi: 10.1145/1148170.1148307. (Cited on pages 10, 34, 51, 55, 60, 80 and 89.)
- Chenliang Li, Bin Bi, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. StructuralLM: Structural Pre-training for Form Understanding, May 2021a. (Cited on pages 47, 51, 52 and 54.)
- Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. DiT: Self-supervised Pre-training for Document Image Transformer. In *ACM Multimedia 2022*, July 2022. (Cited on pages 55 and 89.)
- Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. DocBank: A Benchmark Dataset for Document Layout Analysis, November 2020. (Cited on pages 9, 93 and 102.)
- Peizhao Li, Jiuxiang Gu, Jason Kuen, Vlad I. Morariu, Handong Zhao, Rajiv Jain, Varun Manjunatha, and Hongfu Liu. SelfDoc: Self-Supervised Document Representation Learning, June 2021b. (Cited on pages 55 and 90.)
- Y. H. Li and A. K. Jain. Classification of Text Documents. *The Computer Journal*, 41(8):537–546, January 1998. ISSN 0010-4620. doi: 10.1093/comjnl/41.8.537. (Cited on page 15.)

- Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph Matching Networks for Learning the Similarity of Graph Structured Objects, May 2019. (Cited on page 54.)
- Yulin Li, Yuxi Qian, Yuchen Yu, Xiameng Qin, Chengquan Zhang, Yan Liu, Kun Yao, Junyu Han, Jingtuo Liu, and Errui Ding. StrucText: Structured Text Understanding with Multi-Modal Transformers. *arXiv:2108.02923 [cs]*, August 2021c. (Cited on pages 55 and 107.)
- 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. (Cited on pages 59 and 73.)
- Weihong Lin, Qifang Gao, Lei Sun, Zhuoyao Zhong, Kai Hu, Qin Ren, and Qiang Huo. ViBERTgrid: A Jointly Trained Multi-modal 2D Document Representation for Key Information Extraction from Documents. In Josep Lladós, Daniel Lopresti, and Seiichi Uchida, editors, *Document Analysis and Recognition – ICDAR 2021*, Lecture Notes in Computer Science, pages 548–563, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86549-8. doi: 10.1007/978-3-030-86549-8_35. (Cited on pages xv, 52, 53, 54 and 73.)
- Xiaoqing 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. (Cited on pages 3, 60 and 73.)
- 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, July 2019b. (Cited on pages 49, 84 and 96.)
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. August 2021. (Cited on page 55.)
- D. Lohani, A. Belaïd, and Y. Belaïd. An Invoice Reading System Using a Graph Convolutional Network. In Gustavo Carneiro and Shaodi You, editors, *Computer Vision – ACCV 2018 Workshops*, volume 11367, pages 144–158. Springer International Publishing, Cham, 2019. ISBN 978-3-030-21073-1 978-3-030-21074-8. doi: 10.1007/978-3-030-21074-8_12. (Cited on pages 53, 58 and 73.)

- Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. Icml*, volume 30, page 3. Atlanta, Georgia, USA, 2013. (Cited on page 27.)
- Larry M Manevitz and Malik Yousef. One-Class SVMs for Document Classification. (Cited on page 15.)
- Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A Dataset for VQA on Document Images, January 2021. (Cited on page 13.)
- Andrew McCallum. Efficiently Inducing Features of Conditional Random Fields. 2012. doi: 10.48550/arXiv.1212.2504. (Cited on page 39.)
- Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. 2000. (Cited on page 15.)
- Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In I. Guyon, U. Von 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. (Cited on page 39.)
- Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR). *IEEE Access*, 8:142642–142668, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.3012542. (Cited on page 8.)
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, September 2013a. (Cited on pages 38, 39, 47, 58, 73 and 74.)
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013b. Association for Computational Linguistics. (Cited on page 38.)
- M. Minsky. *Semantic Information Processing*. MIT Press, 1968. ISBN 978-0-262-13044-8. (Cited on page 16.)
- M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969. (Cited on page 24.)
- Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the Number of Linear Regions of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. (Cited on page 26.)

- Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada. *Optical Character Recognition*. Wiley, April 1999. ISBN 978-0-471-30819-5. (Cited on page 4.)
- Harold Mouchere, Christian Viard-Gaudin, Richard Zanibbi, and U. Garain. ICFHR2016 CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 607–612, Shenzhen, China, October 2016. IEEE. ISBN 978-1-5090-0981-7. doi: 10.1109/ICFHR.2016.0116. (Cited on page 8.)
- Kevin Murphy. *Probabilistic Machine Learning An Introduction*. 2022. (Cited on page 22.)
- Christopher Olah. Understanding LSTM Networks – colah’s blog. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>, 2015. (Cited on pages xiii and 34.)
- Rasmus Berg Palm. *End-to-End Information Extraction from Business Documents*. PhD thesis, DTU Compute / Technical University of Denmark, 2019. (Cited on page 74.)
- 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, Kyoto, November 2017. IEEE. ISBN 978-1-5386-3586-5. doi: 10.1109/ICDAR.2017.74. (Cited on pages 13, 40, 51, 52, 60, 62 and 73.)
- 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, December 2018. (Cited on pages 11, 42 and 60.)
- Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. CORD: A Consolidated Receipt Dataset for Post-OCR Parsing. *Document Intelligence Workshop at Neural Information Processing Systems*, 2019. (Cited on pages 12, 59 and 61.)
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. (Cited on page 28.)
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient Context Window Extension of Large Language Models, August 2023. (Cited on page 107.)

- Qiming Peng, Yinxu Pan, Wenjin Wang, Bin Luo, Zhenyu Zhang, Zhengjie Huang, Teng Hu, Weichong Yin, Yongfeng Chen, Yin Zhang, Shikun Feng, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE-Layout: Layout Knowledge Enhanced Pre-training for Visually-rich Document Understanding, October 2022. (Cited on pages [xix](#), [55](#), [84](#) and [107](#).)
- Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. (Cited on pages [39](#), [47](#), [73](#) and [74](#).)
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv:1802.05365 [cs]*, March 2018. (Cited on pages [39](#), [48](#) and [74](#).)
- Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter W. J. Staar. DocLayNet: A Large Human-Annotated Dataset for Document-Layout Analysis. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3743–3751, August 2022. doi: 10.1145/3534678.3539043. (Cited on pages [xi](#) and [9](#).)
- 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, July 2021. (Cited on pages [47](#), [54](#), [56](#), [74](#), [89](#), [90](#), [97](#) and [107](#).)
- 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 [58](#), [60](#) and [61](#).)
- Ofir Press, Noah A. Smith, and Mike Lewis. Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation, April 2022. (Cited on page [46](#).)
- 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, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1082. (Cited on pages [53](#) and [60](#).)
- Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosFormer: Rethinking Softmax in Attention. *arXiv:2202.08791 [cs]*, February 2022. (Cited on pages [46](#), [91](#), [94](#) and [96](#).)

- Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise Self-Attention for Long Document Understanding. *arXiv:1911.02972 [cs]*, November 2020a. (Cited on page 90.)
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26, 2020b. (Cited on page 60.)
- J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986. ISSN 1573-0565. doi: 10.1007/BF00116251. (Cited on page 20.)
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. page 12, 2018a. (Cited on pages 48, 74 and 90.)
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2018b. (Cited on pages 50 and 58.)
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. February 2021. (Cited on page 55.)
- 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. *arXiv:1910.10683 [cs, stat]*, July 2020. (Cited on pages 46, 49 and 50.)
- Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, August 2017. (Cited on page 26.)
- Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for Activation Functions, October 2017. (Cited on page 27.)
- Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation, February 2021. (Cited on page 29.)
- 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 39 and 63.)

- Lev Ratinov and Dan Roth. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics. (Cited on page 39.)
- Sangeeth Reddy, Minesh Mathew, Lluís Gomez, Marçal Rusinol, Dimosthenis Karatzas., and C. V. Jawahar. RoadText-1K: Text Detection & Recognition Dataset for Driving Videos, May 2020. (Cited on page 8.)
- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, May 2016. (Cited on page 29.)
- Alexander M. Robertson and Peter Willett. Applications of -grams in textual information systems. *Journal of Documentation*, 54(1):48–67, January 1998. ISSN 0022-0418. doi: 10.1108/EUM0000000007161. (Cited on page 37.)
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. (Cited on pages 33 and 54.)
- Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. (Cited on pages xii, 19 and 24.)
- Max Roser, Hannah Ritchie, and Edouard Mathieu. Technological Change. *Our World in Data*, May 2013. (Cited on page 1.)
- Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient Content-Based Sparse Attention with Routing Transformers. *arXiv:2003.05997 [cs, eess, stat]*, October 2020. (Cited on page 91.)
- D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representations by Error Propagation. In *Readings in Cognitive Science*, pages 399–421. Elsevier, 1988. ISBN 978-1-4832-1446-7. doi: 10.1016/B978-1-4832-1446-7.50035-2. (Cited on page 27.)
- Clement Sage, Alexandre Aussem, Haytham Elghazel, Veronique Eglin, and Jeremy 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, Australia, September 2019. IEEE. ISBN 978-1-72813-014-9. doi: 10.1109/ICDAR.2019.00211. (Cited on pages 13, 40, 51, 60, 62 and 73.)
- 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 *EMNLP 2020 Workshop on Structured Prediction for NLP*, Punta Cana (online), Dominican Republic, November 2020. (Cited on pages 40, 41 and 74.)

- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter, February 2020. (Cited on pages 45 and 107.)
- Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M. Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, Ofir Press, Colin Raffel, Victor Sanh, Sheng Shen, Lintang Sutawika, Jaesung Tae, Zheng Xin Yong, Julien Launay, and Iz Beltagy. What Language Model to Train if You Have One Million GPU Hours?, November 2022. (Cited on page 50.)
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools, February 2023. (Cited on page 51.)
- M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. In *IEEE Transactions on Signal Processing*, volume 45, pages 2673–2681, 1997. doi: 10.1109/78.650093. (Cited on page 36.)
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units, June 2016. (Cited on page 37.)
- K. Seymore and R. Rosenfeld. Learning Hidden Markov Model Structure for Information Extraction. 1999. (Cited on page 15.)
- Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-Attention with Relative Position Representations, April 2018. (Cited on pages 46 and 97.)
- Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need, November 2019. (Cited on page 91.)
- E.M. Shehab, M.W. Sharp, L. Supramaniam, and T.A. Spedding. Enterprise resource planning: An integrative review. *Business Process Management Journal*, 10(4):359–386, August 2004. ISSN 1463-7154. doi: 10.1108/14637150410548056. (Cited on page 2.)
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, December 2017. (Cited on page 29.)
- Štěpán Šimsa, Milan Šulc, Michal Uříčář, Yash Patel, Ahmed Hamdi, Matěj Kocián, Matyáš Skalický, Jiří Matas, Antoine Doucet, Mickaël Coustaty, and Dimosthenis Karatzas. DocILE Benchmark for Document Information Localization and Extraction, February 2023. (Cited on pages 12 and 13.)
- Stephen Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1):233–272, February 1999. ISSN 1573-0565. doi: 10.1023/A:1007562322031. (Cited on page 15.)

- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. (Cited on pages [xiii](#), [32](#), [33](#) and [44](#).)
- Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. Kleister: Key Information Extraction Datasets Involving Long Documents with Complex Layouts. volume 12821, pages 564–579. 2021. doi: 10.1007/978-3-030-86549-8_36. (Cited on page [12](#).)
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding, August 2022. (Cited on page [46](#).)
- Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A Survey of Optimization Methods from a Machine Learning Perspective, October 2019. (Cited on page [21](#).)
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. 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. (Cited on page [40](#).)
- Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse Sinkhorn Attention, February 2020. (Cited on page [91](#).)
- Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient Transformers: A Survey. *arXiv:2009.06732 [cs]*, March 2022a. (Cited on pages [xiv](#), [45](#) and [91](#).)
- Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying Language Learning Paradigms, October 2022b. (Cited on pages [49](#), [50](#), [51](#), [74](#) and [107](#).)
- Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, January 2021. (Cited on page [55](#).)
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models. February 2023. (Cited on pages [50](#), [56](#) and [107](#).)
- Yi Tu, Ya Guo, Huan Chen, and Jinyang Tang. LayoutMask: Enhance Text-Layout Interaction in Multi-modal Pre-training for Document Understanding, May 2023. (Cited on pages [xv](#), [47](#), [51](#), [52](#), [74](#), [84](#), [85](#), [89](#) and [107](#).)

- Vladimir Vapnik, Steven Golowich, and Alex Smola. Support Vector Method for Function Approximation, Regression Estimation and Signal Processing. In *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. (Cited on pages 20 and 25.)
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA., June 2017. (Cited on pages xiv, 13, 42, 43, 44, 46, 51, 73, 74, 75 and 89.)
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks, February 2018. (Cited on page 53.)
- Lulu Wan, George Papageorgiou, Michael Seddon, and Mirko Bernardoni. Long-length Legal Document Classification, December 2019. (Cited on page 88.)
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding, February 2019. (Cited on page 74.)
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems, February 2020a. (Cited on pages 74 and 90.)
- Jiapeng Wang, Lianwen Jin, and Kai Ding. LiLT: A Simple yet Effective Language-Independent Layout Transformer for Structured Document Understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7747–7757, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.534. (Cited on page 47.)
- Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv:2006.04768 [cs, stat]*, June 2020b. (Cited on pages 46, 91, 94, 95 and 96.)
- Zilong Wang, Mingjie Zhan, Xuebo Liu, and Ding Liang. DocStruct: A Multimodal Method to Extract Hierarchy Structure in Document for General Form Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 898–908, Online, November 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.80. (Cited on page 54.)
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023. (Cited on pages 51 and 107.)

- Mengxi Wei, Yifan He, and Qiong Zhang. Robust Layout-aware IE for Visually Rich Documents with Pre-trained Language Models, May 2020. (Cited on pages 53 and 61.)
- Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560, 1990. (Cited on page 35.)
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace’s Transformers: State-of-the-art Natural Language Processing, July 2020. (Cited on page 74.)
- BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Froberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rhea Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Deba-

gyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobel, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M. Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névél, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Njadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynek, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourier, Daniel León Perrián, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrmann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabc, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A. Castillo, Marianna Nezhurina, Mario Sängler, Matthias Samwald, Michael Cul-

- lan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S. Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aaroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model, June 2023. (Cited on pages 50 and 56.)
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, October 2016. (Cited on pages 37 and 75.)
- Yuxin Wu and Kaiming He. Group Normalization, June 2018. (Cited on pages xiii and 32.)
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2020.2978386. (Cited on page 53.)
- Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On Layer Normalization in the Transformer Architecture, June 2020. (Cited on page 44.)
- Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding. *arXiv:2012.14740 [cs]*, May 2021. (Cited on pages 55, 58, 60, 61, 74, 84 and 90.)
- 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, August 2020. doi: 10.1145/3394486.3403172. (Cited on pages xv, xvi, xix, 47, 51, 52, 53, 54, 58, 59, 60, 61, 62, 65, 66, 67, 69, 73, 74, 75, 77, 81, 83, 84, 89, 93, 94, 95, 99, 100 and 102.)

- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models, March 2022. (Cited on page 46.)
- Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao, and Furao Shen. Image Data Augmentation for Deep Learning: A Survey, April 2022. (Cited on page 33.)
- 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 33.)
- Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. (Cited on page 47.)
- Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. PICK: Processing Key Information Extraction from Documents using Improved Graph Learning-Convolutional Networks. July 2020. (Cited on pages 53, 54, 60 and 73.)
- Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. *arXiv:2007.14062 [cs, stat]*, January 2021. (Cited on pages 46, 90 and 94.)
- Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization, October 2019. (Cited on pages 32 and 44.)
- Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond Empirical Risk Minimization, April 2018. (Cited on page 34.)
- Kaixuan Zhang, Zejiang Shen, and Jie Zhou. Information Extraction from Text Regions with Complex Tabular Structure. 2020. (Cited on page 32.)
- 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. (Cited on page 66.)
- Xiaohui Zhao, Endi Niu, Zhuo Wu, and Xiaoguang Wang. CUTIE: Learning to Understand Documents with Convolutional Universal Text Information Extractor. *arXiv:1903.12363 [cs]*, March 2019. (Cited on pages 52 and 60.)
- Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. PubLayNet: Largest dataset ever for document layout analysis, August 2019. (Cited on page 9.)



FOLIO ADMINISTRATIF

THESE DE L'INSA LYON, MEMBRE DE L'UNIVERSITE DE LYON.

NOM : Douzon

DATE de SOUTENANCE : 24/10/2023

Prénoms : Thibault, Pierre, Sébastien

TITRE : Language Models for Document Understanding

NATURE : Doctorat

Numéro d'ordre : 2023ISAL0075

Ecole doctorale : InfoMaths

Spécialité : Informatique

RESUME : Every day, an uncountable amount of documents are received and processed by companies worldwide. In an effort to reduce the cost of processing each document, the largest companies have resorted to document automation technologies. In an ideal world, a document can be automatically processed without any human intervention: its content is read, and information is extracted and forwarded to the relevant service. The state-of-the-art techniques have quickly evolved in the last decades, from rule-based algorithms to statistical models. This thesis focuses on machine learning models for document information extraction.

Recent advances in model architecture for natural language processing have shown the importance of the attention mechanism. Transformers have revolutionized the field by generalizing the use of attention and by pushing self-supervised pre-training to the next level. In the first part, we confirm that transformers with appropriate pre-training were able to perform document understanding tasks with high performance. We show that, when used as a token classifier for information extraction, transformers are able to exceptionally efficiently learn the task compared to recurrent networks. Transformers only need a small proportion of the training data to reach close to maximum performance. This highlights the importance of self-supervised pre-training for future fine-tuning.

In the following part, we design specialized pre-training tasks, to better prepare the model for specific data distributions such as business documents. By acknowledging the specificities of business documents such as their table structure and their over-representation of numeric figures, we can target specific skills useful for the model in its future tasks. We show that those new tasks improve the model's downstream performances, even with small models. Using this pre-training approach, we can reach the performances of significantly bigger models without any additional cost during finetuning or inference.

Finally, in the last part, we address one drawback of the transformer architecture which is its computational cost when used on long sequences. We show that efficient architectures derived from the classic transformer require fewer resources and perform better on long sequences. However, due to how they approximate the attention computation, efficient models suffer from a small but significant performance drop on short sequences compared to classical architectures. This incentivizes the use of different models depending on the input length and enables concatenating multimodal inputs into a single sequence.

MOTS-CLÉS : Document Understanding, Machine Learning, Language Models, Pre-Training, Transformers

Laboratoire (s) de recherche : LIRIS

Directeur de thèse : Garcia Christophe
Co-directeur de thèse : Duffner Stefan

Président de jury : Ogier Jean-Marc

Composition du jury : Lemaitre Aurélie (rapporteuse), Paquet Thierry (rapporteur), Tabbone Salvatore-Antoine (examinateur), Ogier Jean-Marc (examinateur)