Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Carthage

Ecole Polytechnique de Tunisie

وزارة التعليم العالي و البحث العلمي

جامعة قرطاج

المدرسة التونسية للتقنيات

Engineering Internship Report

Named Entity Recognition from scanned images

From July 1 to August 31

Within

Centre de Recherche
en Numérique de Sfax
مركز البحث في الرقميات بصفاقس

The Digital Research Center of Sfax
DeepVision teap

Elaborated by: Mohamed BOUZID
3rd year engineering student

Supervised by: Mr. Yousri KESSENTINI
Assistant professor

Academic/University Year :
2020/2021

Abstract

Many pre-trained NLP models are used today for document image understanding. However, they only focus on text while they neglect the visual and layout information which are crucial for better understanding. Hence, we used the pre-trained model LayoutLM to model the interaction between text and layout. The main goal of this project was to extract key information from scanned images. Therefore, in this report we will present methods and techniques we used to achieve this goal. We started by an image processing part, after that we preprocessed the text. In the modeling part, we used pytorch to fine-tune the pre-trained LayoutLM model. Finally, we evaluated our model and we obtained excellent results that were verified by visualizing the predictions.

Key Words: OCR, Deep Learning, NLP, LayoutLM, Layout information, visual information, Pre-training, Fine-tuning.

Résumé

De nombreux modèles en NLP sont utilisés aujourd'hui pour la compréhension des images de documents. Cependant, ils ne se concentrent que sur le texte et négligent les informations visuelles et la mise en page qui sont cruciales pour une meilleure compréhension. C'est pourquoi nous avons utilisé le modèle LayoutLM pour modéliser l'interaction entre le texte et la mise en page. L'objectif principal de ce projet était d'extraire les informations clés des images scannées. Par conséquent, dans ce rapport, nous présenterons les méthodes et techniques que nous avons utilisées pour atteindre cet objectif. Nous avons commencé par une partie de traitement d'image, puis nous avons prétraité le texte. Dans la partie modélisation, nous avons utilisé le pytorch pour fine-tuner le modèle LayoutLM. Enfin, nous avons évalué notre modèle et nous avons obtenu d'excellents résultats qui ont été vérifiés en visualisant les prédictions.

Mots clés : OCR, aprentissage profond, NLP, LayoutLM, positions spatiales, information visuelle, Pre-training, Fine-tuning.

Acknowledgements

I would like to express my gratitude and thanks to the persons named below who supported me, supervised me and helped me during the different stages of my internship. I wish to express the supreme gratitude to Mr. Yousri KESSENTINI, my supervisor and the head of DeepVision research team in the Digital Research Center of Sfax, for all the advice given throughout the internship as well as his expertise in the field of engineering and project management. I would also like to thank all the DeepVision team at CRNS.

Our thanks also go to the entire scientific, academic and administrative body of the Ecole Polytechnique de Tunisie for knowledge, know-how and interpersonal skills that they have constantly taught us during our academic curriculum.

May all those who, from near or far, in any way whatsoever, have contributed to the realization of this work do not feel forgotten, and find here our deepest feelings of gratitude. Doing my internship at CNRS was a great pleasure. I have learned a lot of things.

List of Tables

List of Figures

Contents

Introduction

Nowadays, information is increasingly important, and its flow is becoming larger and quicker. In fact, Documents are available in different forms and structures and are coming from a myriad of sources. This led to a major problem for companies: they cannot exploit the data contained in scanned images, so they need to re-enter it manually. This can become a bottleneck since companies look to save time in their daily tasks.

Because of this problematic and the fact that world documents are composed of rich information, understanding documents automatically has become a hot topic. It refers to the techniques for automatically reading, understanding and analyzing documents like invoices, purchase orders, tax forms, insurance quotes, etc.

As part of my summer internship required by Tunisia Polytechnic School, I worked with the DeepVision research team at the Centre de Recherche Numérique de Sfax (CRNS). I chose CRNS because they offer a diversity of projects in Artificial Intelligence, a perfect ground of a polytechnic engineer student having SIgnal and SYstem as an option. In addition, I wanted to develop my skills in Deep learning.

Currently, receipt digitization plays a critical role in office automation in many financial, accounting and taxation areas. Therefore, the purpose of the project was to extract key information from scanned receipt images. During this internship, I was under the supervision of Mr. Youssri KESSENTINI, an assistant professor in the center.

- In the first chapter, we will talk about the general framework of the project
- In the second chapter, we will present the description of the model architecture
- In the third chapter, we will talk about the steps of the works as well as experimental results

Chapter 1

General Framework of the Project

1.1 Introduction

In this chapter, we will start by presenting the host company **Digital Research Center of Sfax,** where the project has been completed. After that, we will state the problem to resolve and the main goals of the project.

1.2 Host Company Presentation

Figure 1.1: CNRS Logo

The Digital Research Center of Sfax was created in July 2012. It is a public institution whose mission is to conduct research and development activities in the field of information and communication technologies and promote innovation and technology transfer in this field by promoting research results, resources, and expertise. It is also expected to develop international cooperation in science and

communication information technology. It aims to be at the forefront of technology by conducting applied research focused on innovation. Its staff is composed of highly skilled senior researchers (university professors) and PhD. students in computing and telecommunication. They work to elaborate sophisticated solutions and to develop techniques and tools covering recent topics like, e-health, smart transportation, e-government, Image processing, etc..

The research team that I worked with was DeepVision, who is head is my supervisor Mr. Youssri KESSENTINI. The team is dedicated to automatically understanding multimedia data (images, video, digital documents) using deep learning techniques. They focus on acquisition, indexing, modeling, classification or automatic content recognition.

1.3 Problem statement

As we know, business documents are critical to a company's efficiency and productivity. Among different types of documents, Visually Rich Documents (VRDs) have gained more attention. They are known of their composed modality of text, layout, and vision that contains important information in addition to plain text like tabular structure, font, font color, etc. Hence, we can divide them into 4 categories based on text type and layout as shown in table 1.1 .

Table 1.1: Example of documents and their categories.

Layout \ Text type	Structured	Semi-structured
Fixed	invoice, passport, ID card	business email, sales contract
Variable	purchase receipt, business card	resume, financial report

The layout here is described as the texts' relative locations.Precisely, Structured data symblizes data that is arranged in a predetermined schema invoices and receipts. Semi-structured data denotes data that has one or more ids, but each data section is not necessarily arranged into predetermined fields , e.g. resumes.

Some examples are shown in figure 1.2 of documents and entities to be extracted.

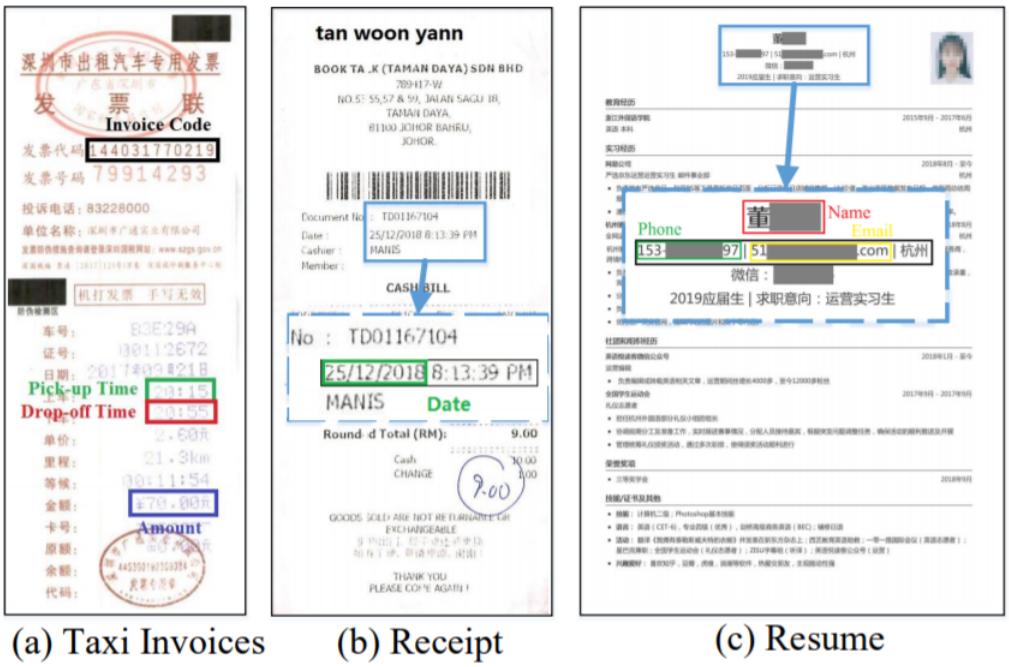


Figure 1.2: Examples of VRDs and example entities to extract.

Due to the diversty of templates and formats, the low quality of scanned document files, as well as the complexity of template structures, understanding VRDs have become difficult task.

Nowadays, several companies extract data from business records using manual efforts that take a lot of time and cost a lot of money. Also, they need manual cusomization and configuration. Moreover, Rules and workflows for each type of text need to be hardcoded and updated to the particular format or when dealing with different formats.

Document AI models have been developed to accelerate the automation of document processing to resolve these issues. Even though these models have made great progress with deep neural networks in the document AI area, most of these methods face two limitations:

- They usually proceed in two stages, An OCR phase followed by an NLP stage, fully discarding the image visual features and layout detail. There

is much more information than plain text when it comes to VRDs. This additional information can be encoded into the model.

- They focus on a few human-labeled training samples without fully exploring the possibility of using large-scale unlabeled training samples.

1.4 Proposed solution

The main purpose of the project is to solve the problem stated above when processing the receipt. We want a faster, easier and efficient process that avoids potential data entry errors.

So, unlike classic approaches, we use an end-to-end architecture that allows to jointly extract and recognize named entities from scanned images of receipts.

This architecture is LayoutLM. We talk about it in detail in the next chapter. It is inspired by recent work in vision and automatic natural language processing. Mainly, it consists of exploiting the attention mechanism with sequence-to-sequence architectures that have shown interesting performances in these fields of research.

1.5 Conclusion

In this chapter, we introduced the host company, the problem statement and the goals of the project. In the next chapter, we will get into detail and present the pre-trained model LayoutLM and how it can jointly model text and layout information.

Chapter 2

The LayoutLM model

2.1 Introduction

In this chapter, we will talk about LayoutLM framework and its architecture. The model outperformed several SOTA pre-trained models on benchmark datasets, and it was the first time that textual and layout information from the scanned document images are pre-trained in a single framework. Since LayoutLM is inspired by the BERT model, we will dedicate the first part of this chapter for a brief summary of BERT.

2.2 Summary of BERT model

BERT, which stands for **Bidirectional Encoder Representations from Transformers**, was introduced by researchers at Google AI in 2018. It is known to be one of the most important updates to the searching algorithms in recent years. It is a language representation model for many NLP tasks with remarkable accuracy.

BERT's key technical contribution was applying the bidirectional training to language modeling. The Transformer encoder reads the whole sequence of words at once, as opposed to directional models which read the text input sequentially (left-to-right or right-to-left). That's why it's called bidirectional. This part makes it possible for the model to learn a word's context based on all its surroundings (left and right of the word).

The architecture of BERT is essentially a multi-layer bidirectional Transformer encoder. To generate final representations, it takes a sequence of tokens and

stacks multiple layers. In depth, the input is a sequence of tokens that are firstly embeddded using WordPiece into vectors.

The input embedding are calculated by summing the corresponding word embeddings, location embeddings, and segment embeddings (figure 2.1). Then, they will be transmitted to a multi-layer bidirectional transformer that can generate context (for a detailed overview of Transformers, see Appendix).

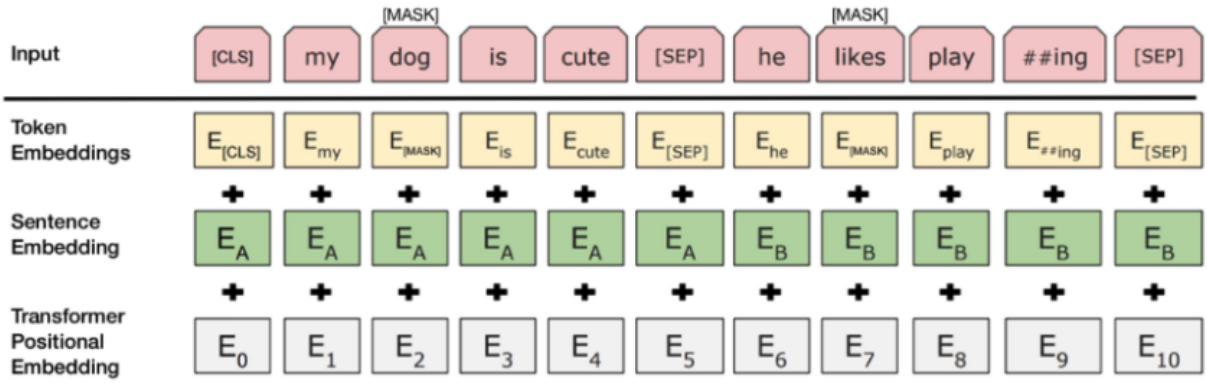


Figure 2.1: Input embeddings

BERT framework works mainly in two steps: pre-training and fine-tuning. During the pre-training, the model make use of two objectives to learn language representation:

- **Masked Language Modeling (MLM)**: 15% of the words in each sequence are masked (by [MASK] token, random token, or keep the original word) before feeding word sequences into BERT, as shown in figure 2.2. The model then tries to predict the original value of the masked words, based on the context given by the other, non-masked, words in that sequence.

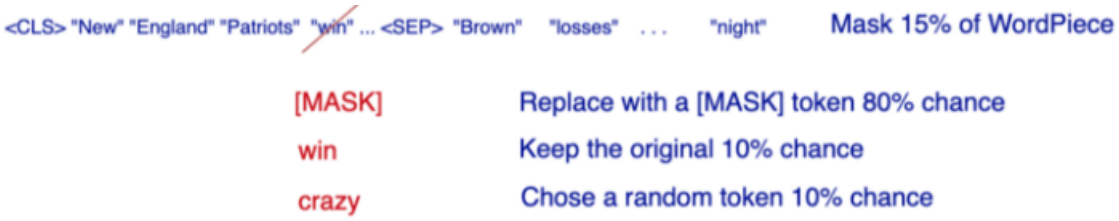


Figure 2.2: Example of Masked Language modelling

- **Next Sentence Prediction (NSP):** is a binary classification task taking a pair of sentences as inputs and classifying whether they are two consec-

utive sentences. To help the model make the distinction between the two sentences in training, a [CLS] token is going to be inserted at the beginning of the first sentence, and a [SEP] token is going to be inserted at the end of every sentence. We can see an example of NSP in figure 2.3

Figure 2.3: Example of Next Sentence Prediction

Finally, in the fine-tuning, task-specific datasets are used to update all parameters in an end-to-end way. The BERT model has been successfully applied in a set of NLP tasks such as Question & Answering, Named Entity Recognition, classification tasks, etc.

2.3 LayoutLM

2.3.1 Main Idea

Even though BERT-like models became the state-of-the-art techniques on many NLP tasks, they only leverage text information for any kind of inputs. Hence, LayoutLM proposed to align visually rich information with text embeddings. Mainly, there are two types of features which improved the language representation of a visually rich document

- **Document Layout Information:** It is clear that the relative positions of words in a document (different from the BERT relative position in a sequence) contribute a lot to the semantic representation.

  Let us take Passport as an example, its corresponding value is expected to be on its right or below instead of on the left or above. Thus, this position information can embedded as 2-D position representation.

- **Visual Information:** It is another important feature for document understanding. Usually, documents contain some visual signals that show the importance and priority of segments.

For document-level visual features, the whole image can indicate the document layout, an essential feature for document image classification. For word-level visual features, styles like bold, underline, and italic, can be very helpful for the sequence labeling tasks like in our project.

2.3.2 Model Architecture

The LayoutLM architecture (figure 2.4) used BERT as a core and two atdditional input embeddings were introduced: a 2-D position embedding and image embedding.

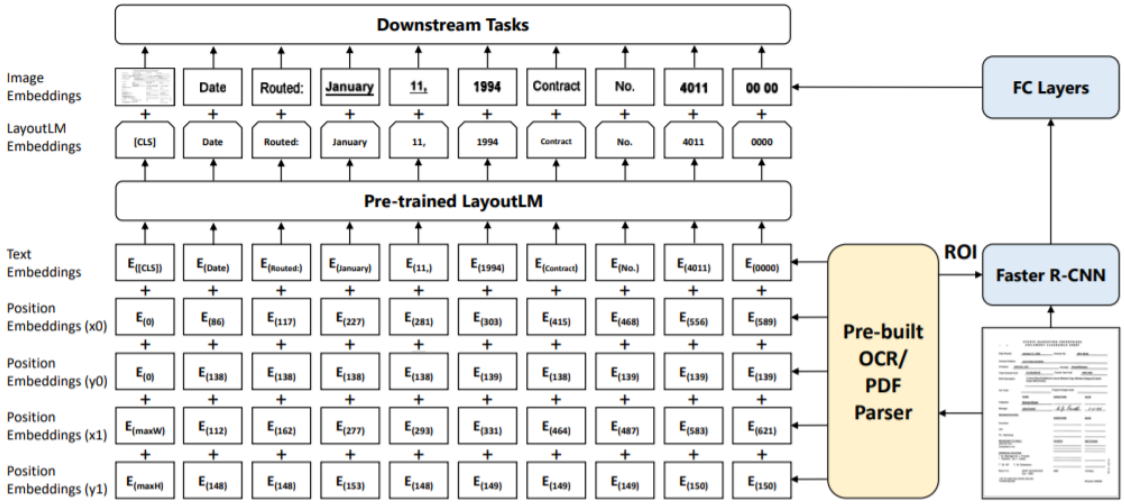


Figure 2.4: LayoutLM Architecture

2.3.2.1 2D Position Embeddings

The relative spatial location in a document is given by 2-D positions embeddings. It's different from the position embedding that describes the word position in a sequence. The paper page is regarded as a coordination system to represent the spatial location of components, with the origin being the top left.

Unlike the position embedding that describes the word position in a sequence, 2-D position embedding gives the relative spatial position in a document. For representing the spatial position of elements, the document page is considered as a coordinate system with the origin being the top left.

Then we set a bounding box whose coordinates are (x_0, y_0, x_1, y_1), where (x_0, y_0)

is the position of the upper left of the bounding box, and (x_1, y_1) is the position of the lower right (figure 2.5).

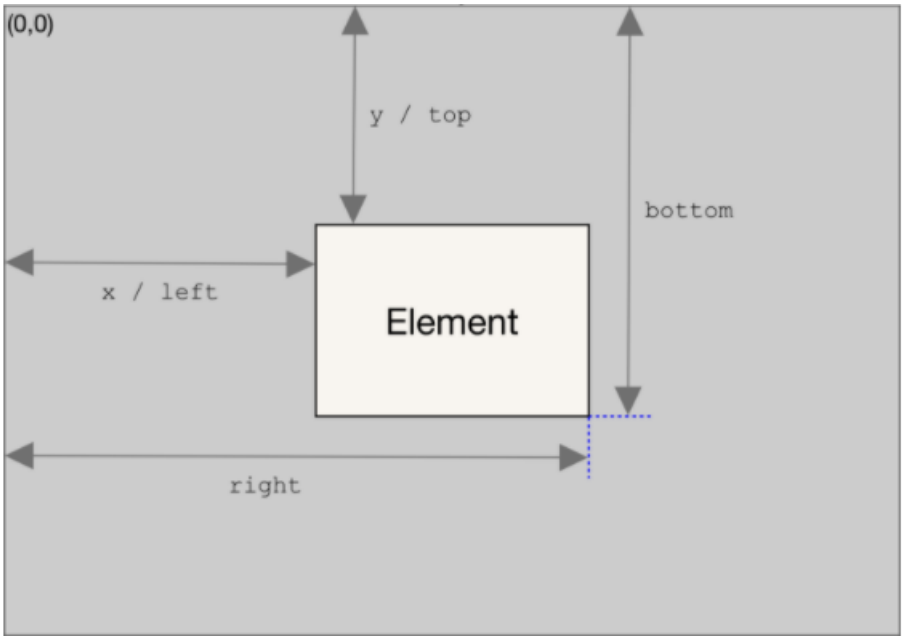


Figure 2.5: Bounding Box coordinates

2.3.2.2 Image embeddings

In order to benefit from the image feature of a document alongside the text, an image embedding layer is added for a better representation. In fact, given the bounding box of each word obtained from OCR results, the image is split into multiple pieces. They have a one-to-one correspondence with the words.

These fragments of image are used to generate the image region features of the images from the Faster R-CNN model. These features are the image embeddings. For the [CLS] token, the whole scanned document is used as the Region of Interest (ROI).

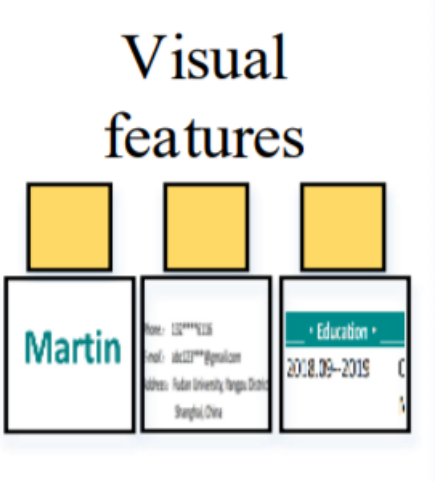


Figure 2.6: Image embeddings

2.3.3 Pre-training LayoutLM

In addition, the LayoutLM is pre-trained on the IIT-CDIP Test Collection 1.02, a dataset of 11 million scanned document images. They are from different categories: letter, email, handwritten, invoice, news, articles, scientific publication, resume, scientific report, specification, and many others. This is ideal for large-scale self-supervised pre-training.

The pre-training phase of LayoutLM, like BERT, uses a multi-task learning objective to learn the language representation. The first one including a Masked Visual-Language Model (MVLM) loss and a Multi-label Document Classification (MDC) loss, which further enforces joint pre-training for text and layout.

2.3.3.1 Task #1: Masked Visual-Language Model (MVLM)

This task is inspired by BERT masked language model, the MVLM learn language representation with the help of 2-D position embeddings and text embeddings. In fact, during this phase, some of the input tokens are randomly masked. However, the corresponding 2-D position embeddings are kept. The model is trained to predict the masked tokens given the contexts. These masked tokens are replaced with the [MASK] token 80% of the time, a random token 10% of the time, and the original token 10% of the time.

Also, given that the document layout may vary in different page sizes, we scale the actual coordinate to a "virtual" coordinate: the actual coordinate is scaled

to have a value from 0 to 1,000.

2.3.3.2 Task #2: Multi-label Document Classification (MDC)

Since the IIT-CDIP dataset included many tags for each document image, a Multi-label Document Classification loss was also used in the pre-training phase.

The idea behind it is that given a set of scanned documents, we utilize the document tags in supervising the pre-training process. The model will be able to cluster the knowledge from different domains and generate better document-level representation.

2.4 Fine-Tuning LayoutLM

After pre-training LayoutLM, it can be fine-tuned on many document images understanding tasks, including form understanding task, document image classification task and receipt understanding task, which is our project goal. In this phase, LayoutLM uses sequential labeling to detect each type of entity. The receipt understanding task can be resumed as follows : given a set of receipts, we need to fill specific slots (e.g., company, address, date, and total). Therefore, using the sequence labeling process, the model only needs to predict the corresponding values. We will talk more about this part in chapter three since it is the goal of our project.

2.5 Conclusion

In this chapter, we talked about LayoutLM, a simple yet effective pre-training technique with text and layout information in a single framework. Based on the Transformer architecture as the backbone, LayoutLM takes advantage of multimodal inputs, including token embeddings, layout embeddings, and image embeddings. In the next chapter, we will present the different steps of our project and how we used LayoutLM to obtain good results in receipt understanding.

Chapter 3

Modeling and Results

3.1 Introduction

In this chapter, we will explain the different steps we used to solve the problem starting from document preprocessing. We will also explain how we used LayoutLM model to fine-tune it on our task. Finally, we are going to explore the results of our project.

3.2 General Desrcription

3.2.1 Problem Statement & Project Goals Recall

Receipts contain the information that's needed for trade to occur between companies. For managing information effectively, companies need to extract and store the relevant information contained in these documents, which was done manually earlier. As mentioned in the first chapter, the goal of this project is to extract texts of a number of key fields from given receipts.

3.2.2 Dataset and Annotations

During this work, we used **SROIE** (Scanned receipts OCR and information extraction) dataset provided by ICDAR 2019 (Robust Reading Challenge on Scanned Receipts OCR and Information Extraction). The data have 1000 whole scanned receipt images split into labeled training and validation set and unlabeled test set. An example of a scanned receipt is show below in figure 3.1.

Figure 3.1: Example of a scanned receipt

To make things easier, the dataset source also provided OCR results. In fact, each image in the dataset is annotated with text bounding boxes and the text itself.

Coordinates form a rectangle with 4 vertices in clockwise order starting from the top. As stated in the previous chapter, we only need the upper left and the lower of the rectangles. The annotations of each image are stored in a text file sharing the same name with the image. The format of the annotation is shown below figure 3.2

```
x1_1, y1_1,x2_1,y2_1,x3_1,y3_1,x4_1,y4_1, transcript_1
x1_2,y1_2,x2_2,y2_2,x3_2,y3_2,x4_2,y4_2, transcript_2
x1_3,y1_3,x2_3,y2_3,x3_3,y3_3,x4_3,y4_3, transcript_3
...
```

Figure 3.2: Example of boxes annotations for an image

Plus, for the information extraction task, the dataset has another annotation file for each image that contains the labels. The model is going to be trained to predict them. Mainly, it is a dictionary (JSON file) with the keys being the entities (Company, date, address, total), and the values are the extracted information related to the entity. An example of the format is shown below figure 3.3


```
{"company": "STARBUCKS STORE #10208",
"date": "14/03/2015",
"address": "11302 EUCLID AVENUE, CLEVELAND, OH (216) 229-0749",
"total": "4.95",
}
```


Figure 3.3: Example of keys annotations for an image

3.3 Document Preprocessing

3.3.1 Image Preprocessing

Receipt understanding is a difficult task since receipts have a lot of diversity and are sometimes scanned poorly for the following reasons: Noisy images, faded images, camera motion and shake, watermarking, faded text, etc. As a result, OCR results may not be correct. Hopefully, for the fine-tuning dataset (SROIE), OCR results are provided in the annotation file. However, we still need this step when we want to try our model in a new image in order to create the annotation from OCR.

Hence, it is necessary to process images, and there are many techniques to do that. The most important preprocessing methods can be resumed as follow :

- **Re-scaling:** Resizing or scaling is a transformation applicable to a digital image that consists of changing the size, whether to enlarge it or to shrink, as a zoom would do. The whole purpose of this is because OCR works best on 300 ppi (pixels per inch) or more.
- **Noise removal:** Noise is the change of color or brightness between pixels of an image. It decreases the readability of the text from the image. There most known types of noise are salt and pepper noise and Gaussian noise. The purpose of the Noise removal step is to make the image smoother by removing small dots/patches which have high intensity than the rest of the image.
- **Binarization:** also known as Thresholding. It is the process of converting an image to two only two colors, which are black and white because OCR engines work better in that case. There are many types of thresholding like simple thresholding, Adaptive thresholding, Otsu's Binarization. The crucial part of thresholding is determining the threshold otherwise, we will not get good results. We can see an example in figure 3.4.

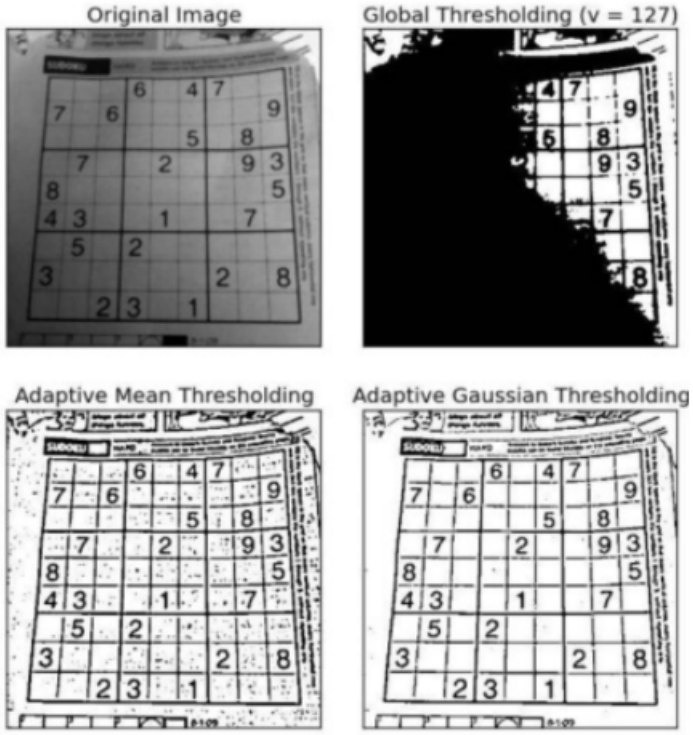


Figure 3.4: Example of thresholding

3.3.2 Annotations Preprocessing

In order to use LayoutLM, we need to convert our data into the format expected by LayoutLM. For example :

- SROIE provides the labels for OCR lines, but not tokens. We need to map the given labels to each token in the current dataset so that the LayoutLM can be applied. The same thing applies to bounding boxes that's why we need to get word-level annotations. That is why we defined two functions (**get_label** and **get_word_bbox)** to help us get the bounding box and the label of each word
- Key annotations and OCR results are stored in separate files. We need to create a single annotation file for each image.

3.3.2.1 Getting the bounding box of each word

Since we only have the bounding box of the whole line, the first solution that may come in mind is to reapply OCR to get the bounding box of each word alone. However, we could have some mismatches from the provided OCR results. My intuition was to apply some approximation heuristic rule.

The idea was that given that we have the bounding box of the whole line of length **L,** which contains a known number of words. Having the length **l** of each word, we can measure the proportion of that words in the whole sequence (l/L) and use that fraction to deduce the limits of the bounding box of each word. This solution remains approximate but somehow correct as we verified that by visualizing the bounding box in the image as shown below (figure 3.5).

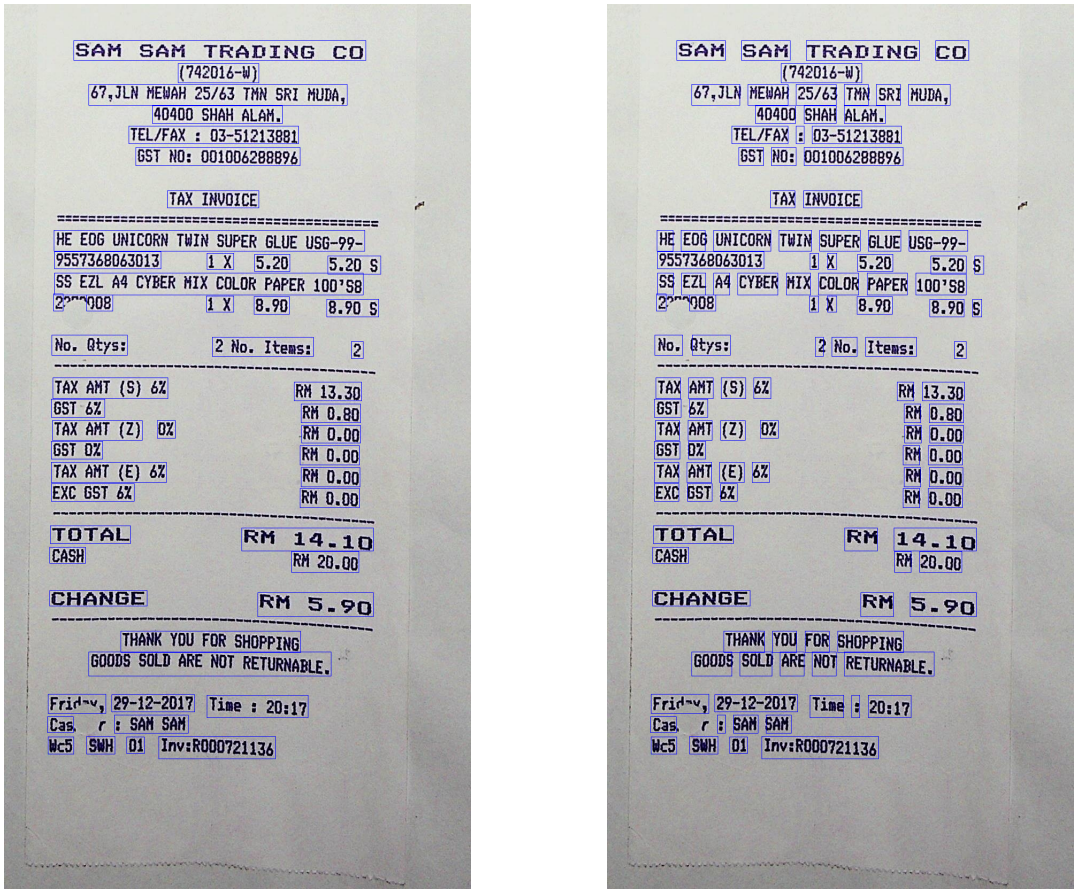


Figure 3.5: Example of getting bounding box of each word

3.3.2.2 Getting the label of each word

These are the labels that the model is going to try predicting them. In our task, a word can have 5 labels: Company, Date, Address, Total, Other. As mentioned above, we need to match the given labels to each token in the current dataset. If the word is a part of the provided label, we map it to the corresponding entity, otherwise we label it as other.

To do this, we used regular expression (Regex): A package provided by python where you define rules to specify a match with a desired set of strings; these strings can be in our project the date, the total, etc. The regular expression engine compares this regular expression to a string to find it in a text block. It returns a boolean that is true if the text matches the proposed regular expression, and false if not the case. Here is a table resuming the most known regular expression (table 3.1)

Symbol	*Meaning*	*Example*
`.`	any character character `.` itself is represented as `\.`	`a.c` matches `abc`
`*`	zero or more repetition of the previous character	`a*` matches `aaa`
`+`	one or more repetition of the previous character	`a+ = aa*`
`?`	zero or one repetition of the previous character	`a?` matches `a` or null character
|	or connective	`a|b` matches `a` or `b`
`[]`	any character within the bracket	`[abc] = a|b|c`
`\a`	any alphabetic characters	`\a = [abc...z]`
`\d`	any numeric characters	`\d = [012...9]`
`[^abc]`	any character other than `a`, `b` or `c`	`[^abc]` matches `e`

Table 3.1: Regular expressions

3.3.2.3 Creating a new annotation file

After we defined the two previous functions, we used them to create a new annotation file containing all information (words, boxes, labels) of each image in one file. In fact, each image information is encoded in a JSON file. Each form is described by an id, a label (company, date, total, address, other), a bounding box of the entire sequence, and a list of words. Each word in the list is described by its textual content and its bounding box. An example of the annotation is shown in figure 3.6.

```
{
    "form": [
        {
            "id": 0,
            "box": [
                72,
                25,
                326,
                64
            ],
            "text": "TAN WOON YANN",
            "label": "other",
            "words": [
                {
                    "text": "TAN",
                    "box": [
                        72,
                        25,
                        130,
                        64
                    ]
                },
                {
                    "text": "WOON",
                    "box": [
                        150,
                        25,
                        228,
                        64
                    ]
                },
                {
                    "text": "YANN",
                    "box": [
                        247,
                        25,
                        326,
                        64
                    ]
                }
            ],
        },
```


Figure 3.6: Example of the full annotation of an image

Finally, we need to do a final step of preprocessing which is preprocessing this JSON file into txt. Precisely, each image has its own annotations. We will need to put all information of all images in just 3 files because that is what LayoutLM demand. We keep a blank line between two different receipt information so that will help us later in the post-processing step. We also used **Transformers Tokenizer** (a word can also be split in token if it does not belong to the vocabulary of the model) to keep track of the maximum sequence length that can be accepted by the model which is fixed at 512. These 3 files are respectively:

Train_box.txt: or test_box.txt (depends if we are working on training or testing). This file contains the words with their respective **scaled** bounding boxes. This what we need for the text and 2d embeddings we talked about. Example: "JOHOR 408 172 444 185".

Train_image.txt: or test_image.txt. This file contains the words with their **actual** bounding boxes alongside their respective image name. This what we need for the image embeddings.

Train.txt: or test.txt. This file contains the ground-truth labels. The model is going to be trained to predict these labels.
Unlike other files, there's something new we added in these files which is known as tags. In fact, we need to keep track of word position in the label sequence (at the beginning, inside the sequence, not in the sequence). To do that, we used the BIO tags for each category. For example, B-DATE, I-DATE, B-Com, I-Com and O for others. B stands for the beginning, I stands for insider and O stands for outside. We can see an example below in figure 3.7

```
TAN     O
WOON    O
YANN    O
BOOK    B-COMPANY
TA      I-COMPANY
.K      I-COMPANY
(TAMAN  I-COMPANY
DAYA)   I-COMPANY
SDN     I-COMPANY
BHD     I-COMPANY
789417-W        O
NO.53   B-ADDRESS
55,57   I-ADDRESS
&       I-ADDRESS
59,     I-ADDRESS
JALAN   I-ADDRESS
SAGU    I-ADDRESS
18,     I-ADDRESS
TAMAN   I-ADDRESS
DAYA,   I-ADDRESS
81100   I-ADDRESS
JOHOR   I-ADDRESS
BAHRU,  I-ADDRESS
JOHOR.  I-ADDRESS
```

Figure 3.7: Example of tags used for train.txt

3.4 Modeling

In this part we used LayoutLM and fine-tuned it on a sequence labeling task. I trained the model for 100 epochs with a batch size of 16 and a learning rate of 5e-5.

The model outputs 2 files :

test_predictions: similar to test.txt but instead of ground-truth labels it contains the predicted label in the same format as figure 3.7.

test.txt: which contains the score and the classification report. Keep in mind that the score here is by comparing test.txt to test_predictions.txt. Later in this chapter we will use another metric to compare two dictionaries and give us the score.

3.5 Output Postprocessing

Since our model provides predictions in one file (test_predictions.txt). the post-processing step aims to regroup the individual predictions: each image will have its corresponding predictions in a txt file sharing the same name with image.

The second step of postprocessing to put those predictions into a dictionary in the same format as the ground truth labels in figure 3.3. To do that, we used the BIO tags we defined previously. In fact, if a word has a B tag, we extract the corresponding entity and any following word having an I tag and the same entity until we reach an O tag, which corresponds to the end of that entity. We repeat the same process for the four entities (company, date, address, total). Hence, we obtained our final dictionary containing the predicted labels.

3.6 Metrics & Results

3.6.1 Chosen Metric

For the scoring metrics, Since we are working on a classification task, the first metrics that come to mind are Precision, Recall and hmean (F1-score). We used these metrics to evaluate our model and they can be defined as follows:

Precision: It is defined as the number of true positive divided by the total of predicted positive.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall: It is defined as the number of true positive divided by the total of actual positive.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

F1-Score: It is a function of precision and recall defined as follows

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

3.6.2 Results

After all the preprocessing, modeling and postprocessing steps , we used those metrics to evaluate our model. Hopefully, we obtained an excellent score of 95.04% shown below in figure 3.8.

```
precision = 95.65
recall = 94.58
f1_score = 95.04
```

Figure 3.8: Score of the model

For more details, we **classification report** function from **seqeval** package to show us the performance of our model by entity, as shown below in 3.9. As anticipated, the best performance was scored by the date entity since all dates have a fixed format.

	precision	recall	f1-score	support
ADDRESS	0.92	0.96	0.94	228
COMPANY	0.89	0.93	0.91	235
TOTAL	0.90	0.97	0.93	564
DATE	0.97	0.99	0.98	273
micro avg	0.92	0.97	0.94	1300
macro avg	0.92	0.97	0.94	1300

Figure 3.9: Score by entity

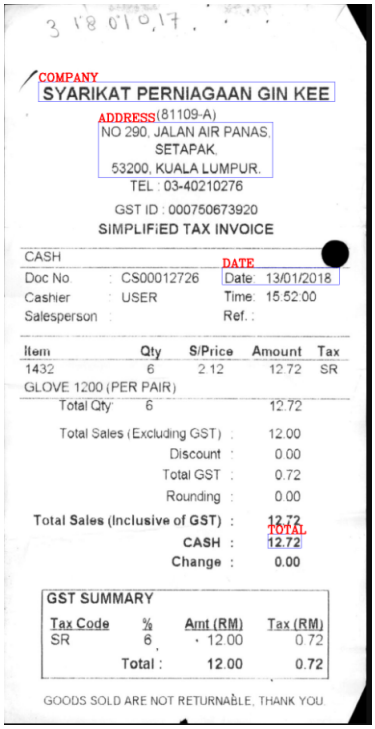
We can also get the score per sample. In fact, for each sample, it shows us the number of ground-truth categories, the number of determined categories, the number of correct labels, the scores, and the log. Here is an example below in 3.10

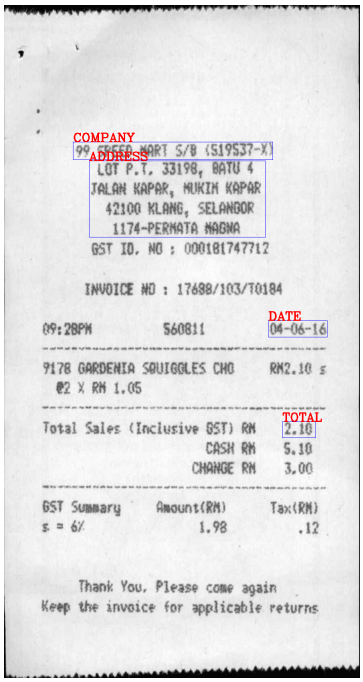
precision = 0.75
recall = 0.75
hmean = 0.75
gtCategories = 4
detCategories = 4
correct = 3
log =
det category company = **CROSS CHANNEL NETWORK SDN. BHD.** correct
det category address = **47, JALAN MERANTI 1, SEK. 3, BANDAR UTAMA BATANG KALI, 44300 BATANG KALI, SELANGOR.** correct
det category total = **6.36** incorrect
det category date = **29/01/2018** correct

Figure 3.10: Score Per sample

3.7 Visualization of the predictions

In the final step of our project, we used **opencv** to draw a bounding box of the predicted labels over the corresponding image. Here are some examples of model performance.





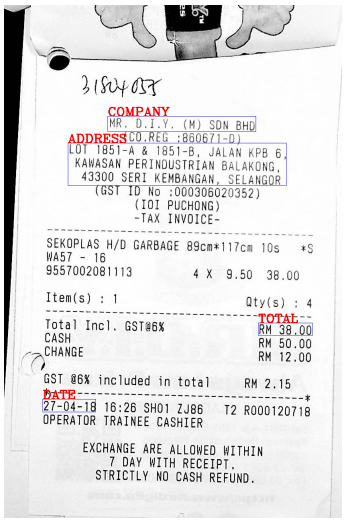


Figure 3.11: Visualization of some results

3.8 Conclusion

In this last chapter, we have presented how we have preprocessed the data, applied LayoutLM and how we extracted information from receipts with excellent results.

Chapter 4

Conclusion and Discussion

In this project, we extracted the Company name, Address, Date and Total from a scanned receipt by fine-tuning the pre-trained LayoutLM.

To achieve this, both our receipt image and annotations have been preprocessed. After that, we trained our model to recognize key information from receipts. Similarly, for post-processing, we regrouped entities from predictions. The results were good.

Even though the main objectives of our project have been achieved, the system we developed could be improved by other enhancements:

- Our project only processes scanned receipts with length limitations. Long receipts were not considered, and that is something we could improve.
- Text reading and information extraction are two seperated tasks in this model architecture, and the model focuses on enhancing the task of extracting information. We may suggest a new architecture in which the two tasks can improve each other.

Bibliography

[1] https://www.linkedin.com/company/research-center-on-ict-of-sfax/about/

[2] http://www.crns.rnrt.tn/research-team/deepvision

[3] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2019. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. arXiv preprint arXiv:1912.13318 (2019).

[4] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[5] https://medium.com/@jonathan_hui/nlp-bert-transformer-7f0ac397f524

[6] https://nanonets.com/blog/receipt-ocr/

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Appendix A

Sequence to sequence learning and Attention Mechanism

A.1 Sequence to sequence learning

Sequence to sequence (Seq2Seq) is a neural network that is transforming a sequence (of words for example) into another sequence. Seq2Seq model are basically an Encoder and a Decoder: The encoder will map the sequence into a higher dimensional space. This abstract vector is going to be fed into the Decoder which turns it into another output sequence that can be in another language, symbols,etc.

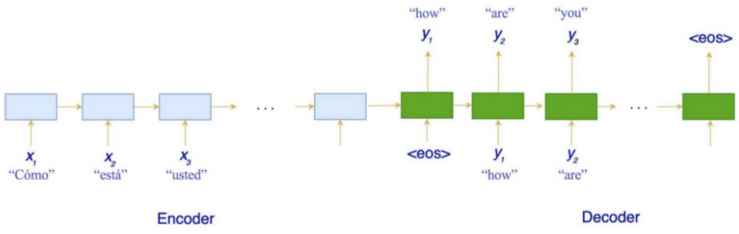


Figure A.1: Sequence to Sequence

Usually, it was used with RNN, LSTM, GRU but they share some drawbacks like the difficulty of learning long range context as well as being directional. Currently, it's used in bidirectional RNN which looks at a sentence from both sides. Another important concept is the context of a word. for every word, we should check over all the words in a sequence in order to know what the context

may refer to. However this could be like finding a needle in a haystack. To solve this problem "**Attention**" was introduced.

A.2 Attention Mechanism

The concept of attention mechanism is very simple. Suppose we are translating a sentence, we are going to pay special attention to the word we are presently translating. The same thing apply for a recording, you listen carefully to the segment you're writing down. Neural networks the same behavior using attention. For example in RNNs we used to encode the whole sentence in a hidden state. Using attention, each word has its corresponding hidden state that is passed to the decoding stage.

Attention mechanism do this by allowing the decoder to look back at the encoder's hidden states based on its current state. Hence, the decoder will extract only important information from the input at each decoding, which will help it learn more complicated dependencies between input and output.

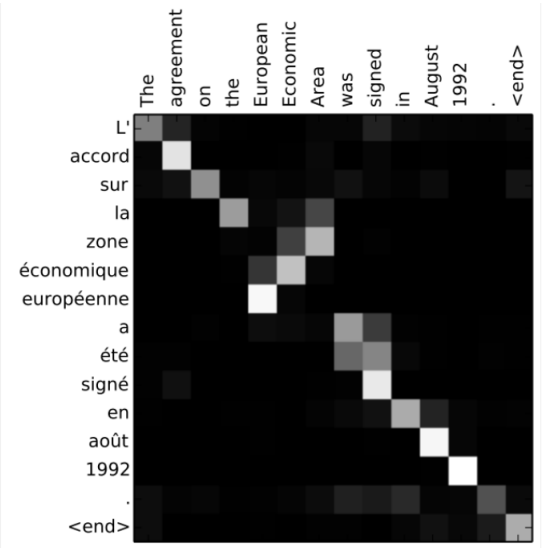


Figure A.2: Example of how a model can pay attention

Appendix B

Transformers

The paper 'Attention Is All You Need' [7] introduced a new architecture called Transformers. It used the attention mechanism presented earlier. It's an architecture that transforms one sequence into another one with the help of two parts (Encoder and Decoder). The difference from the previous sequence-to-sequence models is that it does not imply any RNNs (LSTM,GRU, etc). We can see in the figure below the model architecture.

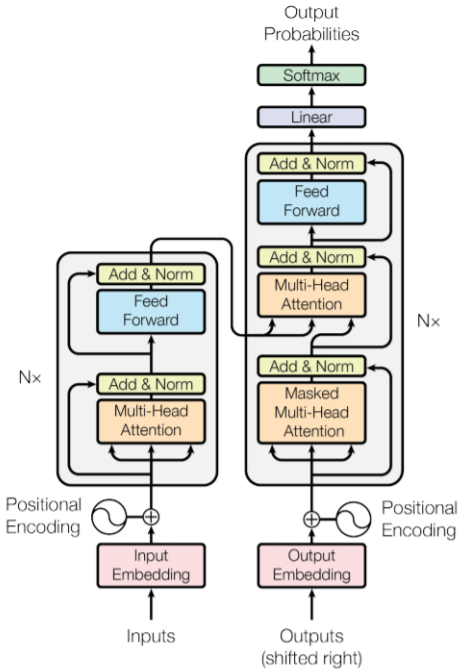


Figure B.1: The Transformer - model architecture

We have the Encoder on the left and the Decoder on the right. They both contains a number of modules that can be stacked on top of each other multiple times, which is what **Nx** means in the figure. The model uses mainly Multi-Head Attention and Feed Forward layers. The inputs and outputs are embedded into a an n-dimensional space and combined with their relative position embeddings in the sequence.

The shifting part after the the embeddings is because we don't want the model learns to simply 'copy' the decoder input, since the target for the i^{th} position would be the i^{th} word in the decoder input. Thus, this shifting will make our model train to predict the i^{th} target word given words in position 1, ..., i-1 in the decoder sequence.

Now, Let's get more into detail by describing the Multi-Head Attention module.

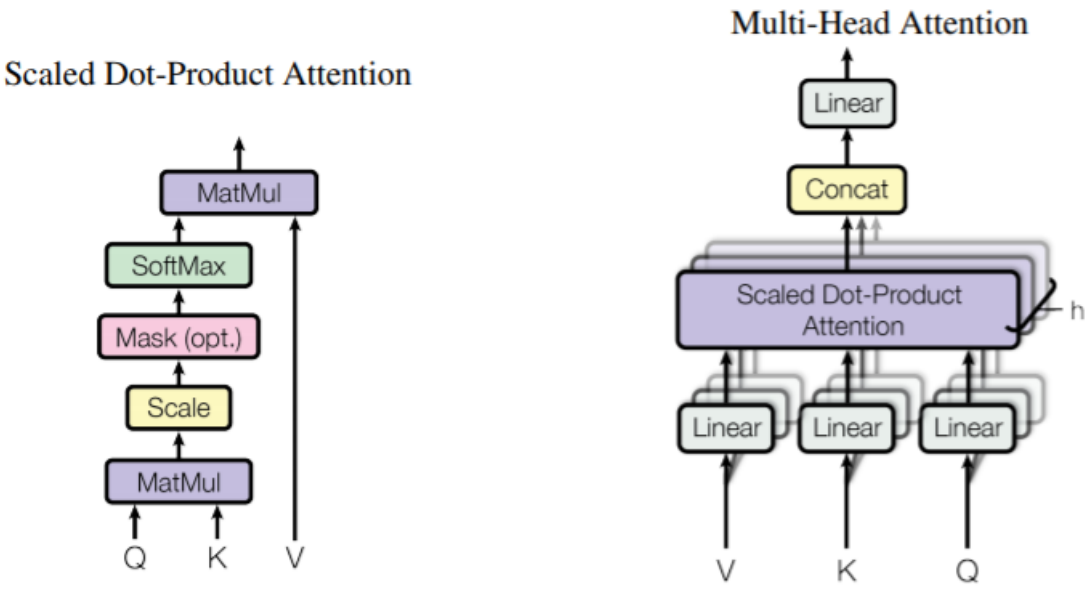


Figure B.2: (left) Scaled Dot-product Attention. (right) Multi-Head Attention consisting of several attention layers running in parallel.

The mathematical description of the scaled Dot-product Attention is simple. It can be resumed by the the following equation

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Q is the matrix containing the query (vector representation of one in the sequence), K are all the keys(vector representations of all the words in the sequence) and V are the values while d_k is a scaling factor.

The right figure show that the attention-mechanism can be parallelized by repeating the scaled dot-product attention mechanism multiple times with linear projections of Q, K and V. This allows the system to learn from different representations of Q, K and V, which will be beneficial to the model. These linear representations are done by multiplying Q, K and V by weight matrices W that are learned during the training. The second multi-head attention module will assure that the inputs of encoder is taken with the inputs of the decoder. After the multi-attention heads, we have a feed-forward layer in both encoder and decoder. This layer has identical parameters for each position. It can be described as a separate linear transformation of every element from the given sequence.

Finally, the linear and the softmax layer role is to get the decoder output and turn it into a word.

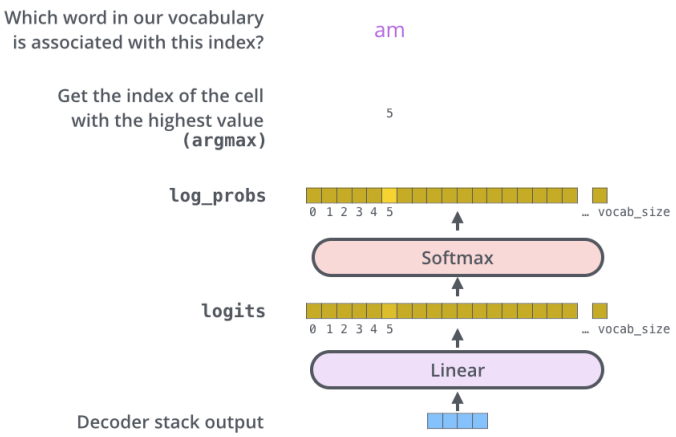


Figure B.3: Linear and softmax layer in the Transformer

This can be done using first the linear layer which is a simple fully connected neural network that projects the input vector into larger vector called logits vector. For example, if the model have 10,000 unique words (vocabulary). The linear layer will make the logits vector 10,000 cells wide with each cell corresponding to the score of a unique word. Then the softmax layer will turn scores into probability where the cell having the highest probability is chosen and the corresponding word is the final output for this time step.