**DOCUMENT LAYOUT EXTRACTION**

* The given task is about Document extraction
* I have referred few papers regarding the document Layout extraction and found the LayoutLM model as the ideal one. I have also discussed the LayoutLM model working principle in the below.
* With the reference of LayoutLM algorithm. I have fine-tune the LayoutLM model weights and parameters by training with the given data set
* The implementation is done in google colab
* The Etraction zip file contains the ‘data’ file, under it we have the training and testing files separately. It also contain extract.ipynb file, layoutlm.pt files
* The python file of implementation is saved in the name of extract.ipynb file and detail description of the code is given in the file at text blocks accordingly.
* **Steps to implement in colab note book**
* Open google drive
* Save the ‘data’ file into the drive
* Open the extract.ipynb file in colab
* Mount the drive
* Start executing the code cell one by one
* **Important note**:
  + Make sure that you have modify the paths of training and testing accordingly
  + Please make sure you are running the code in the colab note book only

**LAYOUTLM MODEL FOR DOCUMENT LAYOUT EXTRACTION**

* LayoutLM is a simple yet effective pretrained method of text and layout for document image understanding tasks. Inspired by the BERT model [1]. Multi-task learning objective for LayoutLM, 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. the LayoutLM is pre-trained on the IIT-CDIP Test Collection.
* In LayoutLM the input textual information is mainly represented by text embeddings and position embeddings, It further adds two types of features
* Document Layout Information: A 2-D position embedding that denotes the relative position of a token within a document.
* Visual Information: An image embedding for scanned token images within a document.
* We add these two input embeddings because the 2-D position embedding can capture the relationship among tokens within a document, meanwhile the image embedding can capture some appearance features such as font directions, types, and colors.
* We get the bounding box of each word from OCR results, we split the image into several pieces, and they have a one-to-one correspondence with the words. We generate the image region features with these pieces of images from the Faster R-CNN [19] model as the token image embeddings. For the [CLS] token, we also use the Faster R-CNN model to produce embeddings using the whole scanned document image as the Region of Interest (ROI) to benefit the downstream tasks which need the representation of the [CLS] token

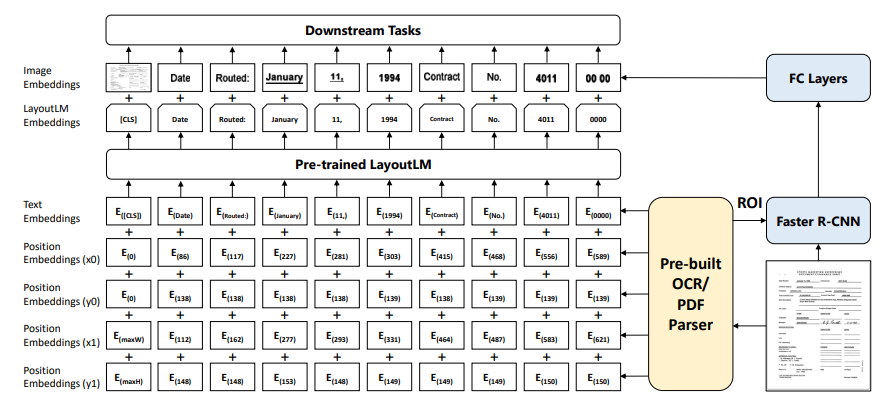


Fig1: An example of LayoutLM, where 2-D layout and image embeddings are integrated into the original BERT architecture. The LayoutLM embeddings and image embeddings from Faster R-CNN work together for downstream tasks.

**IN THE TASK OF DOCUMENT EXTRACTION**

* We improve the LayoutLM model weights by training with the given data set
* This task requires extracting and structuring the textual content of forms. It aims to extract key-value pairs from the scanned form images. In more detail, this task includes Semantic labelling, It is the task of aggregating words as semantic entities and assigning pre-defined labels to them.
* To fine-tune LayoutLM on this task, we treat semantic labelling as a sequence labelling problem. We pass the final representation into a linear layer followed by a SoftMax layer to predict the label of each token.

**OTHER APPROACHES FOR THE DOCUMENT LAYOUT EXTRACTION**

**Generative Grammatical Models for Document Analysis**

* The architecture of a generative grammar model typically consists of several components. First, there is a set of lexical items, which are the basic units of meaning in a language, such as words or morphemes. Second, there are phrase structure rules that determine how lexical items can be combined into phrases and sentences. These rules describe the hierarchical structure of sentences, where phrases are composed of smaller phrases and ultimately of individual lexical items.
* In addition to phrase structure rules, a generative grammar model may also include transformational rules, which describe how one sentence can be transformed into another. For example, a passive sentence can be derived from an active sentence by changing the position of the subject and the object.
* Finally, generative grammar models often incorporate the notion of grammatical categories, such as noun, verb, adjective, and so on. These categories are used to organize the lexical items and to specify their syntactic properties, such as their ability to occur in certain positions in a sentence or to take certain types of complements.

**Non-Generative Grammatical Models for Document Analysis [2]**

* I have referred to the research analysis of Non-Generative Grammatical Models for Document Analysis. Where they have demonstrated the effectiveness and generality of the framework by presenting two applications
* page layout structure extraction, in this case the input to the algorithm is a collection of lines on the page and the output is the section, column, and paragraph structure.
* mathematical expression recognition. In this case the input is a collection of connected components on the page and the output is a set of recognized mathematical symbols and the LaTeX code necessary to reproduce the input. While the final systems are quite different, very few modifications to the learning and parsing process are necessary to produce an accurate recognition system
* Pseudo-code for the training algorithm.
  1. Initialize weights to zero for all productions
  2. Parse a set of training examples using current parameters
  3. For each production in the grammar
     1. Collect all examples from all charts. Examples from the true parse are TRUE. All others are FALSE.
     2. Train a classifier on these examples.
     3. Update production weights. New weights are the cumulative sum.
  4. Repeat Step b.

**Machine Learning classifiers for Document Classification [4]**

**SVM classifier**

* Support Vector Machines are supervised learning models used for classification and regression analysis. Suppose we have a set of training data points {(x1, y1), (x2, y2), ..., (xN, yN )}, where xi ∈ Rd and Yi ∈ {±1}. If these data are linearly separable, we would like to find a linear separating hyperplane classifier H, and two hyperplanes H1 and H2 parallel to H, with the condition that there are no data points between H1 and H2.
* When the data points are not linearly separable, we can use kernel function to transform the data points to some higher dimensional space such that the data points will be linearly separable in that space.
* Another technology is that we can allow some data points, i.e., noise, to be between H1 and H2 and we use the penalty factor C to penalize the situation. we used radial basis function as kernel function and penalty factor C = 1.

**MLP classifier**

* Multi-Layer Perceptron is a widely used type of neural network. It consists of one input layer, one or several hidden layers and one output layer. The input layer distributes the input values to each of the neurons in the hidden layer. For each neuron in the hidden layer(s) and output layer, the output of each neuron in the previous layer is multiplied by a weight and the resulting weighted values and the bias are added together. The summation is transferred by a transfer function. The log-sigmoid function is a frequently used transfer function, which maps the input from negative to positive infinity to the output from 0 to 1. In our experiment, we used the log-sigmoid function as the transfer function for neurons in hidden and output layers
* For the MLP classifier, we used Matlab Neural Network Toolbox4. We adopted the topology of three-layer perceptron, i.e., the topology with only one hidden layer. The weights of the MLP were initialized randomly.

**GMM classifier**

* The Gaussian Mixture Models During training, the Expectation-Maximization (EM) algorithm is used to iteratively refine the component weights, means and variances to monotonically increase the likelihood of the training feature vectors. we can use the EM algorithm to build the models by applying a simple binary splitting procedure after each 10 iterations to increase the number of Gaussian mixtures through the training procedure up to 1024 mixtures. At recognition every pixel line in image is presented by feature vectors, then an alternative Viterbi algorithm uses them and grammars taking into account all possible sequence of models in each pixel line to find the best sequence of hypothesis models. Performances are evaluated in terms of classification rates using an unseen set of historical document images.
* From a practical point of view, GMMs can be seen as one state Hidden Markov Models. We therefore used the HTK toolkit5 to implement the modelling scheme

**REFERENCES**

[1] https://arxiv.org/pdf/1912.13318.pdf

[2] Michael Shilman, Percy Liang, and Paul Viola. 2005. Learning nongenerative

grammatical models for document analysis. In Tenth IEEE International Conference

on Computer Vision (ICCV’05) Volume 1, Vol. 2. IEEE, 962–969.

[3] S. Marinai, M. Gori, and G. Soda. 2005. Artificial neural networks for document

analysis and recognition. IEEE Transactions on Pattern Analysis and Machine

Intelligence 27, 1 (Jan 2005), 23–35. https://doi.org/10.1109/TPAMI.2005.4

[4] H. Wei, M. Baechler, F. Slimane, and R. Ingold. 2013. Evaluation of SVM, MLP

and GMM Classifiers for Layout Analysis of Historical Documents. In 2013 12th

International Conference on Document Analysis and Recognition. 1220–1224. https:

//doi.org/10.1109/ICDAR.2013.247