Applying Transformers Models to Scanned Documents: An Application in Industry

Kuan-Lin Liu*
Center for Data Science
New York University
New York, NY
kll482@nyu.edu

Zichang Ye*
Center for Data Science
New York University
New York, NY
zy1545@nyu.edu

Duo Jiang*
Center for Data Science
New York University
New York, NY
dj1057@nyu.edu

Meenakshi Jhalani*
Center for Data Science
New York University
New York, NY
mgj265@nyu.edu

Abstract

We apply two transformers models, BERT [1] and LayoutLM [2], on two sets of scanned documents provided by the Zillow Group and obtained promising results. Our best-performing model pushes multi-class macro-average recall to 0.95 (47% absolute percentage improvement compared to the baseline Amazon Textract) and reaches a multi-class macro-average precision of 0.91 (8% lower than Amazon Textract). We also conduct experiments on handling extreme class-imbalanced problem with Weighted Cross-Entropy Loss, and Focal Loss [3]. Based on that, we propose four training strategies and successfully compare the performance of pre-trained models. Throughout the projects, we have built an end-to-end pipeline that preprocesses data, trains models, and visualizes results. This pipeline integrates well with pre-existing pipelines at Zillow.

1 Introduction

Zillow is one of the biggest online real-estate marketplace companies globally. It helps renters and buyers discover homes and neighborhoods all across the United States based on many factors like commute, crime rate, schools, etc. As part of their business, Zillow deals with massive amounts of utility bills. Hiring humans to extract information from these bills is extremely time-consuming and expensive. Therefore, it is beneficial to build a system that automatically parses, processes, and understands these utility bills. This algorithm must successfully capture visual (spatial) and textual (semantic) information from scanned documents and needs to be generalizable to various types of layouts and formats in different regions of the country. Such a system not only improves the efficiency of document processing at companies similar to Zillow but also possesses enormous commercial value when shipped to the greater public as a service.

We approach this problem by framing this as a multi-class classification problem and solve it by fine-tuning BERT and LayoutLM, two pre-trained transformers models from the Natural Language Processing (NLP) domain, on Zillow datasets. In particular, our approach takes in the textual and

*We thank the mentorship provided by our mentors at the Zillow Group, Andreas Rubin-Schwarz, and Jyoti Prakash Maheswari.

Preprint. Under review.

positional information represented as bounding boxes and parsed by Optical Character Recognition (OCR) algorithm and classifies each bounding box into one of the pre-defined classes.

Our main contributions to the academic conversation are three-folded. First, we apply a novel model (LayoutLM) in the community of document understanding on a real-world dataset and validate its effectiveness. Secondly, we compare the performance of LayoutLM with BERT, the state-of-the-art and popular language model in the Named Entity Recognition task, and conclude that 2D positional information is useful in the setting of document understanding. Finally, we accumulate some experience of fine-tuning these pre-trained models on a custom dataset from a series of trials and errors.

2 Related Work

Document Understanding AI has been an active field for a long time, and many researchers have come up with different ways to solve it. One school of thought is to view this as a computer vision problem. Yu Et al. [4] trains a convolutional neural network on scanned historic academic articles and obtained 85% in recall and precision on modern academic articles. However, many of the recent works include both spatial and textual information, more or less. One effective strategy is to treat this more like a Natural Language Processing problem. Chargrid [5] represents the document as a sparse 2D grid of characters or words which are transformed to the character-level 1-hot encoding to formulate it as instance-level semantic segmentation. Wordgrid implemented by Denk Et al. [6] extends the concept of Chargrid and extracts the word-level information. In addition, Denk Et al. [6] introduces BERTgrid, in which gridded texts are embedded with contextual information by BERT [1]. CUTIE [7] applies Convolutional Neural Networks on gridded texts where texts are embedded as features with the spatial relationship.

Compared to CUTIE and BERTgrid, LayoutLM [2] modifies the positional-embedding structure of BERT [1] to include 2D positional embeddings from bounding boxes, and also embeds visual information of each bounding box using a Faster R-CNN [8]. Recent work also attempts to solve the problem of table extraction with Graphical Neural Network [9], which is claimed to be a natural choice of modeling the structural information in such tasks but faces the challenges of scarce training data.

3 Problem Definition

The question we are answering is: Given the post-OCR results of a scanned document, how can we classify each textual information to one of the pre-defined classes? We further formalize the inputs and outputs of our question.

Inputs: n bounding boxes, each of which is uniquely represented by a combination of four coordinates of the box ($\mathbf{R^4}$), the number of the page where the box resides (int), and parsed text within the box (string).
Outputs: n predicted classes from m pre-defined classes.

It is critical to define how big a bounding box is. For example, given the word “Account Number", shall we treat it as one entity-level bounding box that captivates all two words, or as two separate word-level boxes that capture “Account" and “Number" separately? We decide to set the unit of analysis as a word-level bounding box because such boxes are the direct results from the OCR algorithm, and modeling at this level makes our prediction easily integrate with the pre-existing pipeline. However, our word-level predictions will need an extra step to be combined to create an entity-level prediction. One solution to this problem is to label each word using BIO tagging and concatenate the word-level prediction based on the tags.

The workflow of our entire pipeline is presented in Figure 1.

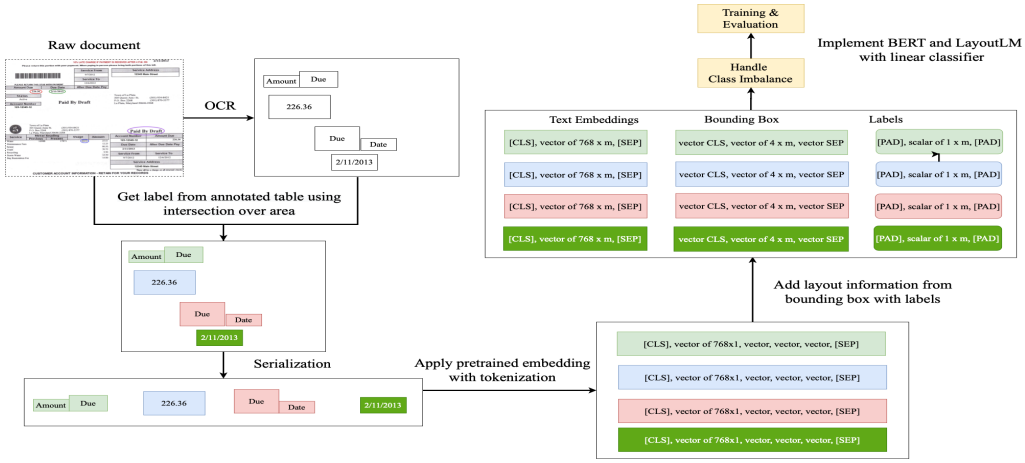


Figure 1: End-to-end workflow

4 Experimental Evaluation

4.1 Data

4.1.1 Source Dataset

We are dealing with two types of datasets: Line-Item and Key-Value. In the Line-Item dataset, the area we are interested in is often within a table, the keys and the associated values on the same line. Our main aim is to extract the value from structures that resemble tables.

The other type of dataset is the Key-Value dataset, where we are interested in a wider range of information such as Service Address, Account Number, etc. For each information, we are also interested in identifying which is the key and which is the value.

The fields in both dataset are summarized in the Table 1.

Line-Item	Key-Value
Electricity	Bill Date
Natural Gas	Due Date
Sewage	Total Amount
Trash	Service Address
Water	Account Number
Tax	Current Charges
Past Due	Service Period
Past Payment	
Balance Forward	

Table 1: Labels in two datasets

For each document, we have an annotations file and post-OCR files. The OCR algorithm bounds a text by a box and parses the text and the coordinates of the bounding box. The human-annotated files have the label, type, and position of the bounding box that we are interested in within a document. An example of both datasets can be found in Figure 2.

4.1.2 Preprocessing

The first task is to find out which post-OCR bounding boxes are annotated, i.e. relevant. We solve it by matching the annotations table to the post-OCR table by how much a bounding box from the annotated tables overlaps with a bounding box from the post-OCR table, given that they are on the

Annotation Dataset:

	source	label	type	top	width	left	height	page_num	ControlNumber	source_file
0	s3://sagemaker.groundtruth.data/document_under...	AmountTotal	key	875	459	1042	60	1	P2B19111800821	Nov
1	s3://sagemaker.groundtruth.data/document_under...	AmountTotal	key	2535	361	937	54	1	P2B19111800821	Nov
2	s3://sagemaker.groundtruth.data/document_under...	AmountTotal	value	881	225	2074	51	1	P2B19111800821	Nov
3	s3://sagemaker.groundtruth.data/document_under...	AmountTotal	value	2595	380	927	162	1	P2B19111800821	Nov
4	s3://sagemaker.groundtruth.data/document_under...	CurrentCharges	key	792	309	1042	49	1	P2B19111800821	Nov

OCR Datasets:

	conf	height	left	line_num	text	top	width	word_num	page_num	block_num	par_num
0	95.724091	35.775203	192.544102	1	MIAMI-DADE	118.324014	260.930204	1	1	1	1
1	93.226509	36.415477	202.158700	2	COUNTY	168.720886	123.269560	1	1	1	1
2	99.189285	41.939413	919.440681	3	Miami-Dade	174.675608	253.549033	1	1	1	1
3	99.863876	40.190017	1180.456880	3	Water	175.608460	128.073540	2	1	1	1
4	99.871437	39.873946	1317.678597	3	and	177.765498	80.582257	3	1	1	1

Figure 2: Annotation and OCR datasets with the important features highlighted in the red boxes

same page. If there is no matching, we label the respective OCR bounding box as “others". [2] We modify the concept of Intersection Over Union (IOU) to Intersection Over Area (IOA) to quantify overlapping. In particular, given a bounding box A and bounding box B,

$$\text{IOA}(A, B) = \frac{(A \cap B)}{B}. \tag{1}$$

Secondly, we define our target label by concatenating the “label" and “type” columns of the annotation dataset.

Label	Type	Final Label
ServiceAddress	key	ServiceAddress-key

Table 2: Label definition

We then split our training and evaluation dataset by time to mimic the fact that we may encounter unseen document layout over time as we enter new markets, change service providers, etc.

Finally, we transform our dataset into the format required by the HuggingFace implementation. In particular, we first tokenize the text within a bounding box using a pre-trained BERT tokenizer. If this particular word is not in the vocabulary of the tokenizer, then the text is broken into several subtokens. All of the subtokens share the same bounding box as their parent word, but only the first subtoken will be given the label of the parent word, and the rest of the subtokens will be labeled as padding tokens, which do not contribute to the loss.

We set the maximum sequence length to be 256. Both LayoutLM and BERT can handle sequence with length up to 512, but in our dataset, 256 is enough to encapsulate all tokens for most documents.[3] If the sequence of tokens after the tokenization step is longer than 256, we break this sequence into several sequences of length 256 with 50 overlapping tokens. If the length of the sequence is shorter than 256, we pad token [PAD] to the end of the sequence until the length meets 256. In addition, each sequence begins with a [CLS] token.

The general steps of preprocessing are summarized in the pseudo-code in the Appendix.

4.2 Class Imbalance

Both Key-Value and Line-Item datasets are facing the same problem: class imbalance. This is because, in a business document, about 90% of the textual information is irrelevant to our purpose. Those irrelevant texts are labeled as class Other in our dataset. Since this is the majority class, the model is likely to predict everything as Other in order to achieve a low loss. Therefore, we have tried

[2] One challenge we faced was that the page number on two tables do not always align. For example, the page number of the post-OCR can be 0-indexed, but the page number of the annotated table can be 1-indexed.

[3] We also tried to set the limit to 50, but it failed to perform well, potentially because it does not utilize the pre-trained model fully.

several techniques to solve this issue, including downsampling, passing weights to the Cross-Entropy loss, or use the Focal Loss function. During fine-tuning, changing the loss function is proven effective in mitigating the issue of class imbalance.

4.3 Model

In this project, we use BERT and LayoutLM, which will be briefly reviewed in this part.

4.3.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a Natural Language Processing Model that applies bidirectional training to Transformers, which uses an attention mechanism to learn contextual relationships between words in a sentence. Previous approaches read text sequentially either from the left or from the right or combine both left-to-right and right-to-left training, which limited contextual learning. The bidirectional training helps the model achieve a higher level of contextual understanding. In this paper, researchers follow the pre-training and fine-tuning approach. In the pre-training, they introduce two training strategies: Masked LM (MLM) and Next Sequence Prediction (NSP), which allows bidirectional training. In the Masked LM approach, some of the words are replaced with [MASK] token, and then the model tries to predict these masked words based on context provided by other non-masked words in the sequence. In the Next Sequence Prediction, the model receives two sentences as input, and it learns to predict if the second sentence is a subsequent sentence in the original document. The input is a sequence of tokens that are embedded into vectors that are the sum of three embedding vectors: positional embedding, sentence embedding, and token embedding. Positional embedding indicates the position of the token in a sequence. In fine-tuning, BERT is trained to adapt to specific tasks. In our case, since our document is 2D and there is no natural ordering of words for the document, we need to serialize words in the document so that it can be fed into BERT. We take a simple way to do serialization: the texts with smaller y come first, and if there is a tie(with some tolerance), texts with smaller x are chosen. After serialization, the document is represented as a 1D sequence of texts, and it can be fed into BERT to do multi-class classification.

4.3.2 LayoutLM

LayoutLM is very similar to BERT but incorporates additional 2D-positional embeddings and image embeddings. The 2D-position embeddings help represent spatial relationships between tokens in a document, and the image embeddings help represent the overall appearance of the document like font, color, etc. LayoutLM uses two pre-training strategies: Masked Visual-Language Model (MVLM) and Multi-label Document Classification (MDC). In the Masked Visual-Language Model approach, some of the words are replaced with [MASK] token, and then the model tries to predict these masked words based on the corresponding 2D position embeddings and context provided by other non-masked words in the document.

5 Training

Since we are fine-tuning the BERT and LayoutLM model with pre-trained parameters, it may potentially lead to overfitting if we unfreeze the layers all at once with too large a network capacity. Therefore, we propose to start by freezing all layers in the backbone of the models and only training the classifiers. There are totally four training strategies introduced below in our experiment. Besides, following the optimizer which was used to fine-tune the pre-trained LayoutLM [2], we also use AdamW [10] to optimize both BERT and LayoutLM in our experiments. Although the model is optimized with two different loss functions to handle the imbalanced classes, we save the best model checkpoint according to the macro-average recall. The learning rate linearly decays if the loss is not decreasing after two patience iteration.

We experiment with four training strategies.

1. *All Layers (Training Strategy A):* We unfreeze all the layers in LayoutLM or BERT during the whole training process and would like to see the performance with such great capacity by exploring the learning curve. The models run for ten epochs with an initial learning rate at $1e{-}4$.

2. *Classifier Only (Training Strategy B):* We only unfreeze layers of the classifier and consider this method to be our baseline because of the lowest model capacity. The models are trained for 25 epochs with an initial learning rate at $1e{-}2$.

3. *All Layers First; Classifier Later (Training Strategy C):* It's also a two-stage training process. We start by training all the layers for five epochs and then only fine-tune the classifier for ten epochs. The initial learning rate is $1e{-}4$ for the first stage and $1e{-}3$ for the second stage.
4. *Classifier First; All Layers Later (Training Strategy D):* The models are trained in two stages. At the first stage, the model only learns from the classifier and freeze all parameters in LayoutLM or BERT for ten epochs. Then at the second stage, it unfreezes all layers, including the model backbone and the classifier, for the rest of the five epochs. The initial learning rate for the first stage is set to $1e{-}2$ and $1e{-}4$ for the second stage.

Instead of using random downsampling or upsampling to handle the imbalanced class problem, we apply two weighted loss functions in our experiments in order not to lose much information because of removing data points. First of all, we change the weights in the Cross-Entropy Loss. The computation of weights follows the formula as below.

$$\text{weight}(c) = \frac{\text{The count of total samples}}{\text{The count of samples in class c}} \tag{2}$$

Secondly, we adopt Focal Loss [3], which was proposed to solve extreme class imbalance in object detection. This method exponentially down-weights the easier and frequent samples:

$$\text{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t) \tag{3}$$

where p_t is the model's estimated probability for the class. γ and α are tunable parameters.

6 Evaluation

6.1 Metrics

Since document understanding is a business application, we choose three metrics to evaluate our model based on the real scenarios where the model will be used.

1. *Document Accuracy*: Out of all the documents in the validation data and test data, how many documents are 100 % correctly predicted, meaning that the model obtains perfect predictions for all bounding box in the document. This is very important because our client will always want to know how likely it is to get intact information from a document.
2. *Multi-Class Macro Average Recall*: Out of all bounding box with a specific class, what is the percentage that is classified correctly.
3. *Multi-Class Macro Average Precision*: Out of all bounding box predicted as a specific class, what is the percentage that is classified correctly.

At this stage of the development, we would like our model to be able to retrieve as much as information as possible from the documents, so we will focus on improving the recall of our model first.

6.2 Results

The baseline result for our is Amazon Textract result. Amazon Textract is an API provided by Amazon so that people can analyze document on it. We use it for our Key-Value dataset and the result is in Table 3.

In this project, we run several experiments to see which configuration gives the best result. In general, we are interested in the comparison of three configurations:

1. BERT VS LayoutLM
2. Focal Loss VS Weighted Cross Entropy Loss

Model	Loss Function	Training Strategy	Precision	Recall	F1-Score
BERTbase	Weighted Cross Entropy Loss	strategy A	0.88	0.85	0.86
		strategy B	0.67	0.54	0.58
		strategy C	0.87	0.84	0.84
		strategy D	0.87	0.83	0.82
	Focal Loss	strategy A	0.85	0.89	0.87
		strategy B	0.69	0.63	0.65
		strategy C	0.84	0.89	0.87
		strategy D	0.86	0.89	0.87
LayoutLMbase	Weighted Cross Entropy Loss	strategy A	0.89	0.91	0.90
		strategy B	0.79	0.65	0.68
		strategy C	0.89	0.90	0.89
		strategy D	**0.91**	0.90	0.90
	Focal Loss	strategy A	0.89	**0.95**	**0.92**
		strategy B	0.77	0.73	0.75
		strategy C	0.87	**0.95**	0.91
		strategy D	0.90	0.93	**0.92**
Amazon Textract			0.99	0.48	0.73

Table 3: Performance of BERT, LayoutLM in different settings and Amazon Textract on the Key-Value dataset. Details of different training strategies can be found in the section 5.

Model	Loss Function	Training Strategy	Precision	Recall	F1-Score
BERTbase	Weighted Cross Entropy Loss	strategy A	0.76	0.82	0.78
		strategy B	0.53	0.58	0.50
		strategy C	0.76	0.82	0.78
		strategy D	0.80	0.81	0.80
	Focal Loss	strategy A	0.79	**0.89**	0.83
		strategy B	0.70	0.52	0.59
		strategy C	0.79	**0.89**	0.83
		strategy D	0.86	0.85	0.85
LayoutLMbase	Weighted Cross Entropy Loss	strategy A	0.83	0.83	0.83
		strategy B	0.57	0.49	0.48
		strategy C	0.81	0.84	0.82
		strategy D	0.79	0.81	0.79
	Focal Loss	strategy A	**0.90**	**0.89**	**0.89**
		strategy B	0.68	0.51	0.56
		strategy C	0.89	**0.89**	0.89
		strategy D	0.85	0.88	0.87

Table 4: Performance of BERT and LayoutLM in different settings on the Line-Item dataset. Details of different training strategies can be found in the section 5.

3. Different training strategies. We have tried four different training strategies, as mentioned in the section 5.

The result can be found in Table 3 for Key-Value dataset and in Table 4 for line item dataset.

6.3 Discussion

We have made a few observations based on the results of our experiments.

1. *BERT performs more evenly on both datasets.* The best precisions on Key-Value and Line-Item are 0.88 and 0.86, while the best recalls are both 0.89.
2. *LayoutLM performs better on the Key-Value dataset than on the Line-Item dataset in general.* The best precisions that LayoutLM achieves on Key-Value and Line-Item are 0.91 and 0.90, respectively. The best recalls are 0.95 and 0.89, respectively. LayoutLM has the better ability to extract all relevant information from the documents, and both models have similar precision, meaning their predictions are comparably relevant.

Dataset	Model	Document Accuracy
Line-Item	LayoutLM	0.29
Line-Item	BERT	0.15
Key-Value	LayoutLM	0.16
Key-Value	BERT	0.09

Table 5: Document accuracy in different settings

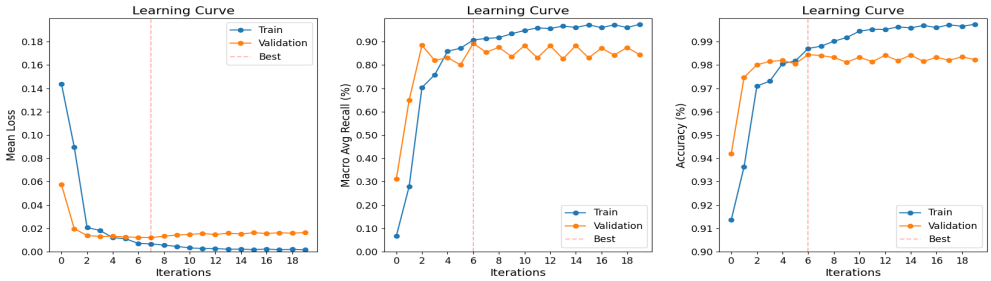


Figure 3: Learning curve for LayoutLM with focal loss when fine-tuning all layers

3. *We need end-to-end fine-tuning.* Strategy B, which is training the linear classifier only, never returns the best performance in any settings.
4. *Focal Loss? It depends on what you want.* On Key-Value, focal loss increases the best recall but unable to increase the best precision for both models. However, on Line-Item, training with focal loss improves both the recall and precision for both models. Recall is defined as a true positive rate, namely (TP/TP+FN). It tends to be negatively influenced if the dataset is highly imbalanced towards irrelevant values because there will be a lot "false irrelevant". Since the focal loss inherently deals with class imbalance, it makes sense that recall is then improved. Meanwhile, precision is defined as (TP/TP+FP), which is not quite sensitive even when there are overwhelmingly many irrelevant samples as long as the classifier is making reliable positive predictions.
5. *Document Accuracy is low.* Even though both BERT and LayoutLM achieve promising results across documents, their document accuracy is still relatively low. It is true that document accuracy is a very strict metric that requires the model to predict perfectly on a document. However, this is a metric that is highly important when pitching to clients or implementing in practice, so we still need to dig deeper into the common mistakes that the model tends to make across the documents.
6. *Few epochs are needed for fine-tuning.* Since both BERT and LayoutLM are pre-trained models, and it only takes about 3 to 4 epochs or 6 to 8 iterations(1 epoch equals two iterations) to converge. One of our learning curves for LayoutLM can be found in Figure 3.

6.4 Error Analysis

1. *Amazon Textract is conservative.* In Figure 4, texts with colored background are the information that we want to extract. Green background means that Amazon Textract gives a correct prediction and red background means that Amazon Textract misses out. We find out that Amazon Textract is very conservative, in that it only predicts relevancy when it is very confident so that its prediction of relevancy is almost always correct. However, as a tradeoff, it is very likely that it misses out on some other more ambiguous but relevant information. In other words, Amazon Textract has high precision and a low recall. It is understandable that Amazon Textract adopts a conservative way because misleading information can be harmful. However, in our project, since our goal is to give the client as complete information as possible, we can afford to sacrifice some precision for recall.
2. *LayoutLM avoids some BERT's mistakes.* LayoutLM outperforms BERT in a lot of settings and we take a closer look into what kind of errors BERT makes but are avoided by LayoutLM. One of the visualization of comparison between BERT and LayoutLM is in Figure 5.

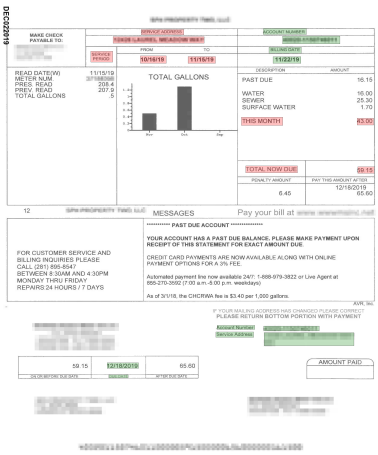
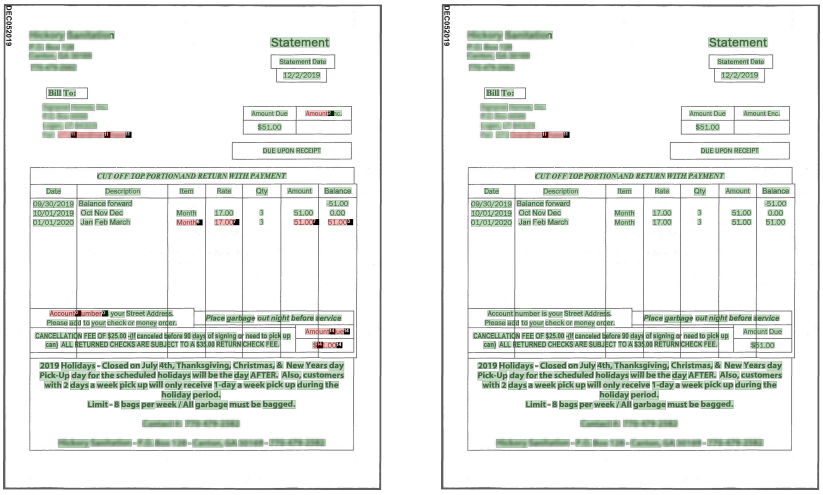


Figure 4: Amazon Textract prediction of document PRA19120203395



(a) BERT Prediction (b) LayoutLM Prediction

Figure 5: Predictions of PRA19120505667 by BERT and LayoutLM. Green background means a correct prediction and red background means an incorrect prediction

a One mistake BERT makes is to use textual information too much when making a prediction since 1D positional embedding is not as useful as in the plain text. For example, in the middle left in Figure 5, there two words "Account Number" with red background in BERT prediction. Although the textual information here is a very strong indication of belonging class "Account Number-Key," it is actually class "other" because it is irrelevant information to our purpose. BERT misclassifies these two words because of the strong textual information, but LayoutLM makes correct predictions because it also uses 2D positional information.

b Another mistake BERT makes is not being able to capture all possible spatial relations between a key and a corresponding value. A value can be to the right of or below the key. LayoutLM can capture this spatial information, while BERT can only capture the left-to-right key-value spatial relation. One example can be seen in the middle right. There is a key-value pair "Amount Due" with red background in BERT prediction. We believe that BERT misclassifies this pair because it cannot link key and value in the document due to all the irrelevant information nearby, and the key and value are very far from each other after 1D serialization. This is not a problem in the LayoutLM framework since this key and value are very closed to each other.

7 Business Impact

Our methods, comparable to the services provided by Amazon Textract, can save the firm $65 every 1,000 pages processed.[4] Supposed our 2000-document labeled dataset is 2-page each and 10% of the entire monthly dataset. Then deploying this model will save a budget of $2,600 monthly and $31,200 yearly for the firm.

This project also has the potential to generate commercial value. According to the Everest Group[5], the market of Intelligent Document Understanding reached US$250-350 million in 2018. We have shown the possibility of customizing an open-source deep learning model and tuning it according to the client-provided dataset, meaning that we have the ability to provide customized solutions to the broader public who has a demand for automatic document processing.

8 Conclusions and Future Work

In this project, we apply results from recent research to real-world problems and obtains results comparable, in some way superior, to existing industry solutions such as Amazon Textract. We can further improve our projects in the following ways. The first is to use BIO tagging so that we can produce an entity-level prediction. Instead of the current word-level prediction, the entity-level prediction will be closer to most use scenarios. Secondly, we can add visual information. The LayoutLM paper uses a convolutional neural network to obtain the visual embeddings of each bounding box and add them on the top of the LayoutLM embeddings before the downstream task. We can also experiment with the inclusion of visual embeddings and see whether it will improve the performance. Thirdly, we can transform our model to be a question answering system to make it interactive for some use cases. Finally, we can wrap our training script into Lightning so that we can be more efficient in tuning hyperparameters and experimenting with a different strategy.

9 Lessons Learned

This work from end to end requires a strong foundation and advanced knowledge in Machine Learning and Deep Learning. All of us are taking DS-GA 1008: Deep Learning this semester. Since the field of document AI is such a broad topic, including Natural Language Processing, Computer Vision, and especially Transfer Learning. We strengthen ourselves as the course goes and basically learn from our mistakes. For example, when it comes to some details of model tuning, we learn to explore the learning curve with error analysis to narrow a wide range of hyper-parameters. In terms of transfer learning, we were struggling with down-fitting in the beginning and over-fitting later, when we start to unfreeze more model parameters. All of us had limited experience in applying the power of transfer learning, so we learn from each other, our mentors, and our mistakes.

Another example is that in industry, data cleaning and preprocessing require a significant amount of time. Understanding that it is hard to get perfect preprocessing right from the beginning, we focused on building an end-to-end pipeline first and iteratively update our preprocessing choices. This method allows us to get proof of concept early and gradually update the results. We also realize that the unit test should have been adopted between each step of preprocessing. During many of our meetings, we spent lots of time organizing and checking the preprocessing scripts back and forth.

This project also allows us to practice our communication skills extensively. The timely communication with Zillow is not only a source of motivation and productivity but also a way of a path-correcting mechanism at some critical moments of this project. We have made six presentations to a different audience and practiced the skills of making different narratives and focal points depending on the audience. As of now, we believe we are more confident in engaging a non-technical person with our data science story.

10 Contributions

1. Zichang Ye: Data preprocessing, LayoutLM training script, Error analysis, Final writeup

[4] https://aws.amazon.com/textract/pricing/

[5] https://www.everestgrp.com/2019-06-intelligent-document-processing-market-is-exploding-expected-to-grow-70-80-over-next-2-years-press-release-50323.html

2. Kuan-Lin Liu: Data preprocessing, Scripts examining and organizing, Training pipeline building, Final writeup
3. Duo Jiang: Data preprocessing, BERT model training, Utility bill visualization, Error analysis, Final writeup
4. Meenakshi Jhalani: Data exploration, Final writeup

11 Appendix

Algorithm 1: Getting Annotation

Result: A labeled OCR dataset.
for *each document* **do**
  for *each OCR bounding box* **do**
    Find all annotations bounding boxes that have large IOA on the particular page;
    if *there is a match* **then**
      labeled as the same label of the matched annotated bounding box;
    else
      labeled as “others";
    end
  end
end

Algorithm 2: Generating Dataset

Result: The Zillow Dataset ready for training
for *each document* **do**
  for *each labeled OCR box* **do**
    Tokenize the word in the box and get a sequence of tokens;
    The tokens shared the same labels and bounding boxes as the parent word;
  end
  if *The length of the token list exceeds 256* **then**
    Break the list into several lists with 50 overlapping tokens;
    Add [PAD] to end if the length of a list is shorter than 256;
  else
    Add [PAD] to end until the length reaches 256;
  end
  Transform the tokens, labels, and bounding boxes information into PyTorch tensors;
  Wrap these tensors as a data instance.
end
Wrap all instances into a dataset class.

Dataset	Model	Mean	Standard Deviation
Line-Item	LayoutLM	0.900	0.007
Line-Item	BERT	0.882	0.008
Key-Value	LayoutLM	0.947	0.004
Key-Value	BERT	0.898	0.004

Table 6: Stability of recall with different seeds

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. *CoRR*, abs/1912.13318, 2019.

[3] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[4] C. Yu, C. C. Levy, and I. Saniee. Convolutional neural networks for figure extraction in historical technical documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 789–795, 2017.

[5] Anoop R Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. Chargrid: Towards understanding 2D documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4459–4469, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[6] Timo I. Denk and C. Reisswig. Bertgrid: Contextualized embedding for 2d document representation and understanding. *ArXiv*, abs/1909.04948, 2019.

[7] Xiaohui Zhao, Zhuo Wu, and Xiaoguang Wang. Cutie: Learning to understand documents with convolutional universal text information extractor. 03 2019.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, June 2017.

[9] Shah Rukh Qasim, Hassan Mahmood, and Faisal Shafait. Rethinking table recognition using graph neural networks, 2019.

[10] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[11] Peng Zhang, Yunlu Xu, Zhanzhan Cheng, Shiliang Pu, Jing Lu, Liang Qiao, Yi Niu, and Fei Wu. Trie: End-to-end text reading and information extraction for document understanding, 2020.