Learning To Cluster: A Comparison of Document Vector Representations for Layout Identification

Stanford CS224N Custom Project

Pooja Sethi
Department of Computer Science
Stanford University
pjasethi@stanford.edu

Bryan Chia
Department of Statistics
Stanford University
bryancws@stanford.edu

Abstract

In this work, we present a new document understanding task: unsupervised ***layout identification***. The objective is to cluster documents together that have similar layout, which we define to jointly include physical appearance and semantic meaning. For example, given an unsorted collection of documents, we may want to cluster all bank statements together, with further sub-clusters dividing statements from Chase Bank vs. Bank of America. Towards this goal, we investigated how effectively LayoutLM and variants of it perform when used to produce document embeddings. When tested on two of our own benchmark datasets, sourced from SROIE2019 and RVL-CDIP, we found that LayoutLM performed 5 to 10 times better (as measured by the Silhouette coefficient and the Calinski-Harbasx index) than simple but effective baselines such as Bag-of-Words and TF-IDF. Our code and experiment results are available at `https://github.com/poojasethi/doc-clustering.git`

1 Key Information

Our TA mentor is Kamil Ali. We have two external mentors, Richard Stebbing & Spencer de Mars (Impira Inc.), and no external collaborators. We are not sharing this project with another class.

2 Introduction

While much research has been done on obtaining representations of individual words [1, 2] and sentences [3, 4, 5], comparatively little has been done on obtaining representations of entire documents. Good document representations, however, could be helpful in a variety of real-world applications. A common use case is the problem of supervised *document classification*, i.e., predicting which category a document belongs to (e.g., invoice, email, bank statement, or scientific report) [6].

Unfortunately, pre-defined labels are often not granular enough. For example, "scientific report" does not discern between an ACL, ICLR, and Nature paper, "bank statement" does not distinguish between Chase Bank versus the Bank of America, and so on. For a model to make these more fine-grained distinctions, it is important it understands a notion of *layout*. Loosely defined, two documents have the same layout if they have similar physical appearance and semantic meaning. Examples of documents with the same layout are given in Figure 1; examples with different layouts are given in Figure 2. Documents that originate from the same template are likely to have the same layout.

In this work, we introduce the new task of *layout identification*. The goal is to cluster documents together that are likely to have the same layout. In contrast to document classification, which is supervised, we explicitly choose to frame layout identification as an unsupervised problem for two reasons. First, the notion of different layouts is relative and thus difficult to annotate. All bank statements, even if coming from different banks, have a similar layout when contrasted against scientific papers. Second, we want our model to be able to distinguish between different types of layouts even when the train data and test data do not overlap.

Figure 1: Bank statements with the *same* layout should be clustered together.

Figure 2: Bank statements with *different* layouts should be in separate but neighboring clusters.

We investigate: how effectively can existing encoders, such as LayoutLM and its variants [7, 8], be used to form document embeddings for layout identification?

3 Related Work

There are several notable approaches to obtaining document embeddings. One well-known approach, Doc2Vec [9], learns a vector per paragraph in the document using methods called PV-DM and PV-DBOW, which are analogous to CBOW and skip-gram, respectively, from word2vec [1]. These paragraph vectors can be concatenated or averaged with word embeddings to get a final document embedding. In addition to Doc2Vec, there have been numerous works in obtaining sentence embeddings, such as Universal Sentence Encoder (USE), Sentence BERT (SBERT), and Simple Contrastive Learning (SimCSE) [4, 3, 5]. Sentence embeddings for a document could also be combined to obtain a document embedding..

The downside of the aforementioned approaches is that they only leverage the text in the document while ignoring information about its physical layout. For example, knowing that a word is at the top-center of a page could convey it is a title and has importance. LayoutLM [7] is a notable model that aims to address this gap. When a document is processed using optical character recognition (OCR), the word's bounding box (position) on the page is generally returned and is represented by a top-left coordinate (x_0, y_0) and bottom-right coordinate (x_1, y_1). The LayoutLM encoder, which builds on BERT [10], capitalizes this positional information in the following way: instead of using BERT's traditional 1-D position embeddings, it adds and trains four additional input layers, one for each of x_0, y_0, x_1, and y_1. The authors refer to these layers as 2-D position embeddings.

Once pretrained, the LayoutLM model is then finetuned and tested on supervised tasks such as document classification and entity extraction, where it outperforms BERT and RoBERTa [11]. However, it is not evaluated in the context of obtaining document embeddings, nor is it tested on any task similar to ours (i.e., unsupervised layout identification). Thus, in our work, we extend its analysis to these use cases.

Finally, it is also worth noting LayoutLMv2 [8]. The authors extend the inputs to the original LayoutLM model, adding visual token embeddings obtained from patches of the document image. Figure 5 in the Appendix highlights the architecture changes. We also test LayoutLMv2 for our task.

4 Approach

From a modeling perspective, the task of layout identification can be broken into two key steps.

1. **Document Encoding** In this step, each document is converted into an *embedding*, using models we discuss in further detail in Sections 4.1 and 4.2.
2. **Document Clustering** Once document embeddings have been obtained, they are each softly assigned to one of k clusters (k is a hyperparameter) using Gaussian Mixture Models (GMMs), which we discuss in further detail in Section 4.3

4.1 Baseline Models

We implemented two baseline methods for obtaining document embeddings: Bag of Words (BoW) and Term frequency-inverse document frequency (TF-IDF), leveraging `CountVectorizer` and `TFIDFVectorizer`, respectively, from the `sckitlearn` library [12]. We then applied dimensionality reduction (PCA) to reduce the embeddings to $\mathbb{R}^{768}$ vectors. We chose this size for consistency with the document embedding size obtained from our advanced models.

4.2 Advanced Models

In addition to the baseline, we used four additional models to obtain document embeddings: a pretrained LayoutLMv1[1] model, two finetuned variants of LayoutLMv1, and a pretrained LayoutLMv2 model. We chose to focus on LayoutLM specifically because its success on other document understanding tasks and accessibility of pretrained models.

To test LayoutLMv1, we leveraged a pretrained model from HuggingFace [13]. We wrote a wrapper around this model that allows us to pass in a preprocessed document as input and obtain the hidden states of the model as output, which in turn we used to create a final document embedding. The approaches we tested to combine the model hidden states into a document embedding are discussed in in Section 5.3.1.

Next, we finetuned the base model twice, separately: once for sequence (document) and once for token (word) classification. We hypothesized that finetuning the model for a sequence classification task, using in-domain data, might help it perform better on layout identification. However, we also worried finetuning could cause the embeddings to lose generalizability. Therefore, we also tested finetuning on an unrelated task, token classification, and with out-of-domain data, to see if it could improve the robustness of the document embeddings. Further details on the datasets we used for finetuning are in Section 5.1.2 and the results are in Section 5.3.2. We referenced code from two examples of finetuning LayoutLM for sequence classification [14, 15].

Finally, we also tested the pretrained LayoutLMv2 model from HuggingFace [16] and finetuned it.

4.3 Clustering Algorithm

For clustering, we implemented a GMM, leveraging `GaussianMixture` [12]. We chose GMM as opposed to K-Means because of its ability to handle non-circular clusters and softly assign documents to clusters. We also implemented interactive visualizations of the cluster assignments, shown here.

5 Experiments

5.1 Data

We prepared datasets both for the core layout identification task and for finetuning of our pretrained models. Our layout identification datasets are available for download here.

5.1.1 Layout Identification Datasets

We created two datasets using documents from SROIE [17] and RVL-CDIP [6], respectively. The SROIE training set consists of 626 scanned receipts and the RVL-CDIP training set consists of

[1]We use LayoutLMv1 to refer to the model in the original LayoutLM paper [7]

320,000 documents from 16 distinct categories, such as invoices, emails, scientific reports, etc., of which we randomly sampled 1,000. While these datasets are traditionally used for benchmarking entity extraction and document classification, respectively, we prepared them for our own task of layout identification. For each document, we ran OCR pre-processing using Impira [18]. This returns all of the tokens in the document and their bounding box positions in a file called `rivlets.json`. An example is provided in Figure 6 in the Appendix. In addition, Impira splits each document into images of each of its constituent pages.

The rivlets and images (for LayoutLMv2) are passed as input to the document encoders, which return their hidden states. The hidden states are then squashed into document embeddings. Finally, the document embeddings are passed as input to the GMM, which returns a cluster assignment per document.

5.1.2 Finetuning Datasets

As described in Section 4.2, we first finetuned using in-domain data, i.e., the full RVL-CDIP [6] test dataset.[2] of 40,000 labeled documents. During finetuning, we used the sequence classification objective, where the model has to assign the document to one of 16 classes. Next, we finetuned using out-of-domain data, i.e., the full FUNSD [19] dataset, which consists of 200 annotated documents. During finetuning, we used the token classification objective, where the model has to tag each token with one of the following labels: "question", "answer", "header" or "other". We used off-the-shelf LayoutLM tokenizers available on HuggingFace to preprocess each of the datasets before finetuning.

5.2 Evaluation method

We evaluated the quality of the clusters obtained using two metrics.[3] These metrics capture how *tight* each cluster is and how *well-separated* the clusters are from each other. The advantage of these metrics is that they allow us to measure cluster quality without labeled data.

The `Silhouette coefficient` evaluates clustering performance based on the pairwise distance of between- and within-cluster distances. The coefficient ranges from $[-1, 1]$. A high score is attained when $b(x)$, the distance of a point to the nearest cluster's points, is large and $a(x)$, the proximity to other points within the same cluster, is small [20]. The coefficient is negative when the point has been classified in the wrong cluster.

The `Calinski-Harabasx (CH) index` evaluates clusters based on the ratio of the between-cluster and within-cluster sum of squares. The index ranges from $[0, \infty]$. A higher value indicates that there is more between-cluster variation and less within-cluster variation in terms of distance.

The key difference between these metrics is that, in measuring between-cluster distances, the CH index compares all cluster centers to the overall center, while the Silhouette coefficient compares them to their closest neighboring clusters. Thus, CH index rewards embeddings that produce clusters far from all other clusters, and not just the nearest neighboring cluster.

5.3 Experimental details

5.3.1 Squashing Hidden States

The hidden states returned by the LayoutLM encoders are a matrix $\in \mathbb{R}^{512\times768}$, where 512 is the maximum sequence (document) length and 768 is the dimension of the final hidden state embedding for each word.[4] Padding or truncation are used to standardize different lengths to 512. We tested the following methods to process the hidden states matrix into a document embedding $\in \mathbb{R}^{d}$.

Method 1: Average all words Take the mean on the sequence dimension (`axis=0`). This averages each column to produce a vector $\in \mathbb{R}^{768}$.

[2]Because we created our layout identification test set by sampling the *training* split of RVL-CDIP, we used the RVL-CDIP *test* split for finetuning to avoid overlap.

[3]See Appendix Equations 1 and 2 for the full Silhoutte Coefficient and Calinski-Harabasx calculations.

[4]We use the terminology "word" in this section for ease-of-understanding. This is also not incorrect because we do use word-level tokenization. However, more precisely, these methods are being applied over tokens.

Method 2: Average all words, mask pads We observed that many of the rows in the hidden states matrix can correspond to pad tokens. For example, if the sequence length is 50, then the last 462 rows in the matrix will be the corresponding hidden state per pad. We therefore used the attention mask to zero-out all rows corresponding to mask tokens, and then take the average as above to get a vector $\in \mathbb{R}^{768}$.

Method 3: Average all words, mask pads, append length We realized one downside of Method 2 is that it loses information about the length of the sequence. So next, we appended the true sequence length (or max length minus number of padded tokens) to the vector produced by Method 2 to create a vector $\in \mathbb{R}^{769}$.

Method 4: Average all words, mask pads, append & normalize length Upon further inspection of Method 3, we realized that the sequence length feature tended to be much higher in value than the other elements of the hidden state vector. So next, we tried to normalize the sequence length by dividing it by the max sequence length before appending it. This still produced a vector $\in \mathbb{R}^{769}$.

Method 5: Last word, append length In addition to averaging the hidden states from all words in the sequence, we also tested using the hidden state from only the last word. To do this, we zero all rows corresponding to words that are not the last word. We then take the sum across the sequence dimension. Similarly to Method 3, we finally append the sequence length. This produced a vector $\in \mathbb{R}^{769}$.

Method 6: PCA on all words, mask pads, append length Finally, we tested performing PCA over the masked matrix $\in \mathbb{R}^{512\times768}$ to get a matrix $\in \mathbb{R}^{768}$. Intuitively, the procedure chooses the principle component of the words that provide the most variance in each embedding dimension. Finally, we appended the sequence length to create a vector $\in \mathbb{R}^{769}$.

5.3.2 Finetuned Model Details

Table 2 in the Appendix shows the train and test accuracy of finetuned LayoutLMv1 and LayoutLMv2 on their respective tasks. Though we finetuned LayoutLMv2, we ultimately did not use its finetuned variants for document clustering because we were seeing better performance with LayoutLMv1.

5.3.3 Clustering Details

We set k equal to 10 based on our prior, rough estimate of the number of layouts present in each dataset. We discuss the choice of k further in Section 6.4.

5.4 Results

Our results are shown in Table 1. One immediate observation is that all of the LayoutLM models performed significantly better than the baseline encoding methods, achieving Silhouette coefficients approximately 5 times higher and a CH index 200 times higher than the baselines, on both the SROIE and RVL-CDIP layout identification datasets. Among the variants of LayoutLM we tested, Vanilla LayoutLMv1 (the base pretrained model) performed the best the most often, achieving a Silhouette score of 0.536 on SROIE and 0.665 on RVL-CDIP. Close behind it was was the model finetuned on an unrelated task, which tied Vanilla LayoutLMv1 on SROIE and did only marginally worse on RVL-CDIP. Finally, and surprisingly, Vanilla LayoutLMv2 mostly under-performed Vanilla LayoutLMv1. However, it did achieve the highest CH index on RVL-CDIP of 20779.4.

We discuss possible explanations for these results in Section 6.

6 Analysis

6.1 Effects of Encoding Model

There were two notable takeaways in the choice of document encoding model.

First, all of the LayoutLM models and variants performed significantly better than the baselines. This can be better understood by examining Figure 3. Qualitatively, we can see that the embeddings

Table 1: Layout Identification Results (Silhouette Coefficient / Calinski-Harabasx (CH) Index)

Method	SROIE ($n = 626$)	RVL-CDIP ($n = 1000$)
Baselines		
Bag of Words (BoW)	0.093 / **27.5**	**0.134 / 90.3**
TF-IDF	**0.103** / 16.9	0.003 / 5.7
Vanilla LayoutLMv1		
Average all words	0.186 / 406.4	0.371 / 1214.2
Average all words, mask pads	0.436 / 2262.7	0.497 / 8525.4
Average all words, mask pads, append length	**0.536** / 3172.3	0.664 / **20513.6**
Average all words, mask pads, append & normalize length	0.436 / 2265.0	0.497 / 8542.2
Last word, append length	**0.536 / 3172.9**	**0.665** / 20442.5
PCA on all words, mask pads, append length	0.460 / 2412.6	0.576 / 18493.6
Finetuned LayoutLMv1 on Related Task (RVL-CDIP)		
Average all words	0.253 / 272.0	0.229 / 388.2
Average all words, mask pads, append length	**0.534 / 3165.2**	**0.660 / 20405.5**
Last word, append length	0.524 / 3094.6	0.654 / 20093.0
Finetuned LayoutLMv1 on Unrelated Task (FUNSD)		
Average all words	0.162 / 308.5	0.308 / 923.4
Average all words, mask pads, append length	**0.536 / 3172.5**	**0.663 / 20503.8**
Last word, append length	**0.536** / 3172.4	0.659 / 19882.2
Vanilla LayoutLMv2		
Average all words	0.163 / 115.83	0.113 / 177.1
Average all words, mask pads, append length	**0.524 / 2887.8**	**0.652** / **20779.4**
Last word, append length	0.517 / 2858.7	0.646 / 20615.7

***Bold** numbers indicate the best performance in each model category.*
***Green** numbers indicate the best performance on each dataset. There can be ties.*

from the LayoutLM models lead to progressively tighter and better separated clusters. We also implemented an animation that allows us to inspect the contents of each cluster, and saw that related documents, such as hand-written notes, were being placed more closely together.

Second, and surprisingly, LayoutLMv2 generally did *not* outperform LayoutLMv1. This went against our expectations given that LayoutLMv2 takes in a richer feature set: it appends 49 additional visual embeddings in addition to the 512 original text embeddings used by LayoutLMv1. One possible explanation for the negligible impact of the visual embeddings is that that the documents image are resized to 224 x 224 before being embedded. We hypothesize that the small image size, and the transformation of a rectangular document to a square, could lead to loss of important layout-related features.

6.2 Effects of Hidden State Squash

The next most impactful design decision was the strategy we used for squashing hidden states. Masking the hidden states of pad tokens alone increased the CH index by a factor of $\sim$5-10 for both SROIE and RVL-CDIP. Appending the length led to further improvement. Surprisingly, normalizing the length (by dividing the maximum sequence length) was worse using the original length. We think this is because normalizing the sequence length “dilutes" this useful feature too much. Another interesting takeaway was that using the last word’s hidden state was comparable to using the average of all words’ hidden states, which might indicate the LayoutLM model does a good job of retaining long-range dependencies.

6.3 Effects of Finetuning

Overall, we had hoped finetuning would increase performance. Instead, we were initially surprised that finetuning on the related task (RVL-CDIP) led to *worse* performance than finetuning on the

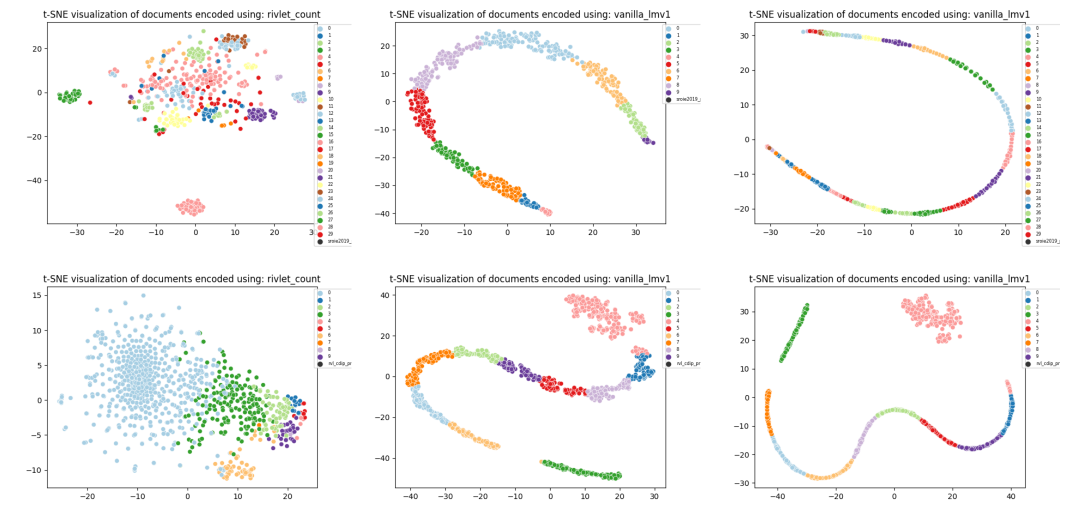

Figure 3: These plots show document embeddings, compressed into two dimensions using t-SNE. Each point corresponds to a document. The top row shows documents from the SROIE layout identification dataset, and the bottom from RVL-CDIP. Each of the three columns represent a different encoding technique: BoW; Vanilla LayoutLMv1 with averaging and masking pads; and Vanilla LayoutLMv1 with averaging, masking pads, and appending length.

unrelated task (FUNSD). And finetuning on the unrelated task was generally no better than not finetuning at all.

However, in hindsight, these results seem congruent. Finetuning on the related task may have caused the document encoder to overfit to that task, thus producing less universally generalizable document embeddings. While the same statement could perhaps be made of the model finetuned on FUNSD, the fact that this task is more *unrelated* to layout identification could have enabled the model keep its generalizability with respect to layout identification. Finally, it's worth noting that for the unrelated task, we only had 200 documents available (compared to 40,000 for related), so it's possible that we would have also seen performance degrade for unrelated as well if we had finetuned with more data.

6.4 Effects of Cluster Size

Finally, we were curious how various cluster sizes k could impact performance. We tested this in Figure 4.

The CH index increases nearly monotonically as k increases, irrespective of the choice of encoding model. In contrast, we see a steady decrease in the Silhouette score. These results are expected. The CH index increases because as there are more clusters, the variance within a given cluster decreases and the cross-cluster variance increases – both of which the CH index rewards. In contrast, the Sihouette score decreases because its measure of cross-cluster variance goes up. This difference is explained by the fact that the CH index compares all cluster centers to the overall center, while the Silhouette score compares all cluster centers to their nearest neighbor's cluster center. With more clusters, the nearest neighbors also become closer.

The most distinctive part of Figure 4 is a sharp discrepancy between LayoutLM and LayoutLMv2 beyond 60 clusters, which repeats both on RVL-CDIP and SROIE. LayoutLM maintains its prior performance, but LayoutLMv2 quickly degrades. One possible explanation is that the the smaller, tail clusters formed using LayoutLMv2 embeddings are noisier than the tail clusters formed using LayoutLMv1. That is, because LayoutLMv2 document embeddings are relatively poor for layout identification, the clusters formed at marginal values of k are also likely to be poor.

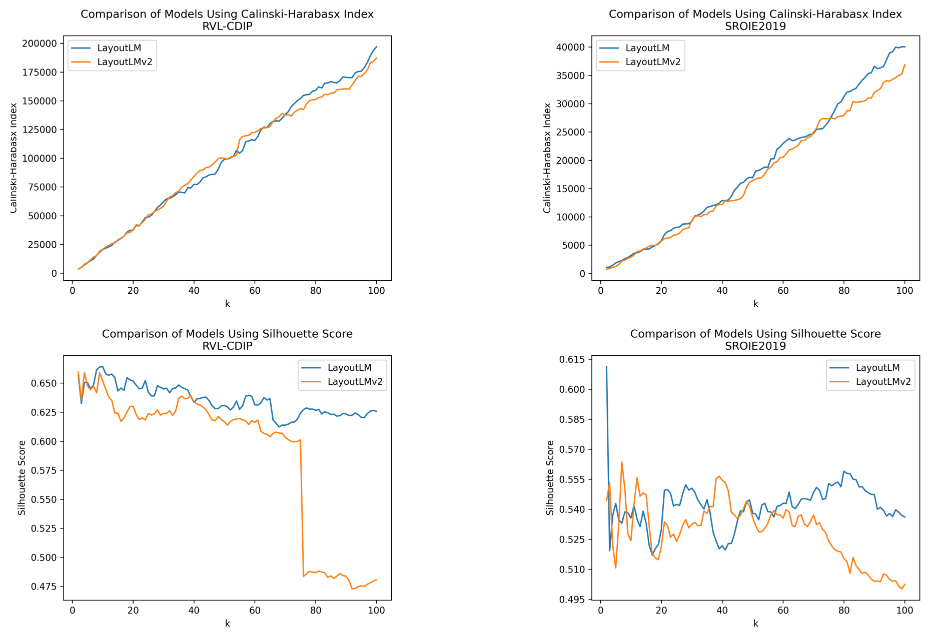

Figure 4: Plots showing changes in clustering performance for increasing values of k. We test vanilla LayoutLM v1 (blue line) and v2 (orange line) on both RVL-CDIP (left column) and SROIE2019 (right column).

7 Conclusion

In this work, we introduced a new task of unsupervised layout identification. The objective of the task is to group related documents together by layout, which refers to the physical structure of a document and as its semantic meaning. Towards this goal, we tested various methods of producing document embeddings. We found that the base pretrained LayoutLM [7] model and variants of it (finetuned versions, as well as LayoutLMv2 [8]) significantly outperformed simple but effective baselines, as measured by Silhouette score and the Calinski-Harbasx index. We also found that an effective way to produce document embeddings is to mask out the hidden states of all padding tokens, average the remaining hidden states together, and finally append the sequence length as an additional feature. This enabled a GMM model to learn tight yet separable clusters.

We did also discover several limitations. First, we would like to dig more deeply into why LayoutLMv2 did not increase performance. A possible next step could be train LayoutLMv2 with higher-resolution and more accurately re-scaled images, which may better preserve layout. Second, we were also surprised finetuning did not help, especially when using in-domain data. We think this happened because we also changed the training objective when finetuning. Keeping the original masked language modeling training objective while just adding the additional data may have been fruitful.

More broadly, we are excited about future possibilities in layout identification. Incorporating features such as font and color, jointly learning the document purpose (e.g., scientific report vs. bank statement), and being able to handle multi-page documents could ultimately lead towards richer document understanding.

References

[1] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[3] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[4] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[5] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*, 2021.

[6] Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. Evaluation of deep convolutional nets for document image classification and retrieval. In *International Conference on Document Analysis and Recognition (ICDAR)*.

[7] Yiheng Xu et al. Layoutlm: Pre-training of text and layout for document image understanding. In *Association for Computing Machinery (ACM)*, 2020.

[8] Yang Xu et al. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Association for Computational Linguistics (ACL)*, 2021.

[9] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31 st International Conference on Machine Learning*. International Conference on Machine Learning, 2014.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Yiheng Xu et al. Layoutlm. 2020. Available at `https://huggingface.co/docs/transformers/v4.16.2/en/model_doc/layoutlm`.

[14] Siva Hemanath. Document classification - layoutlm. 2021. Available at `https://www.kaggle.com/sivahemanth/document-classification-layoutlm`.

[15] Niels Rogge. Fine-tuning layoutlm for sequence classification on rvl-cdip. 2021. Available at `https://colab.research.google.com/github/NielsRogge/Transformers-Tutorials/blob/master/LayoutLM/Fine_tuning_LayoutLMForSequenceClassification_on_RVL_CDIP.ipynb`.

[16] Yang Xu et al. Layoutlmv2. 2021. Available at `https://huggingface.co/docs/transformers/v4.16.2/en/model_doc/layoutlmv2`.

[17] Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and C. V. Jawahar. Icdar2019 competition on scanned receipt ocr and information extraction. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520, 2019.

[18] Impira, 2022.

[19] Jean-Philippe Thiran Guillaume Jaume, Hazim Kemal Ekenel. Funsd: A dataset for form understanding in noisy scanned documents. In *Accepted to ICDAR-OST*, 2019.

[20] Understanding of internal clustering validation measures. In *2010 IEEE International Conference on Data Mining*.

A Appendix

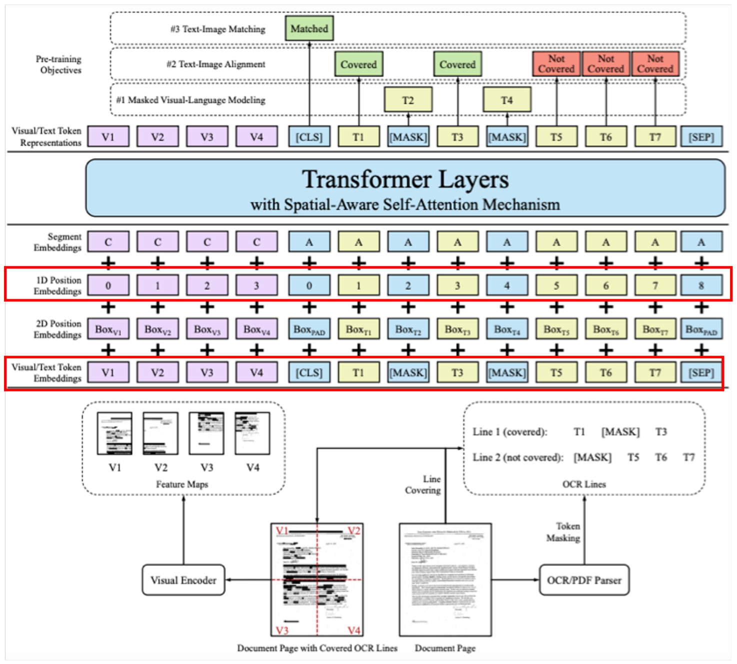

Figure 5: The LayoutLMv2 architecture. The key changes over the LayoutLMv1 architecture are shown in the red boxes.

Table 2: Train and Test Accuracy for Finetuned Models

	LayoutLM Related Task: Document Classification (16 Labels)	**LayoutLMv2 Related Task: Document Classification** (16 Labels)	**LayoutLM Unrelated-Task: Token Classification** (4 Labels)
Train / Test Size	35,988 / 4008	22,500 / 2500	160 / 40
Train Accuracy	86.7%	94.5%	87.2%
Test Accuracy	79.9%	79.9%	47.2%

```
[
    {

        "confidence": 0.8999999761581421,
        "location": {
            "height": 0.018456375838926183,
            "left": 0.0633116883116883,
            "page": 0,
            "top": 0.05453020134228188,
            "width": 0.2094155844155844
        },
        "processed_word": "gardenia",
        "rotated": false,
        "source": "ocr",
        "uid": "ocr-836e40e7c89dcedfb91ec0159cb46c54ae132819-0",
        "word": "GARDENIA"
    },
    {
        "confidence": 0.8199999928474426,
        "location": {
            "height": 0.018456375838926183,
            "left": 0.2905844155844156,
            "page": 0,
            "top": 0.05453020134228188,
            "width": 0.20129870129870125
        },
        "processed_word": "bakeries",
        "rotated": false,
        "source": "ocr",
        "uid": "ocr-334ffc1b8297a92946386ee0768d32111a59d7f6-0",
        "word": "BAKERIES"
    }
]
```

Figure 6: An abbreviated example of a `rivlets.json` file taken from our layout identification dataset, based on SROIE2019 receipts.

$$
\begin{aligned}
\frac{1}{NC}\sum_{i}\Big(\frac{1}{n_i}\sum_{x\in C_i}\frac{b(x)-a(x)}{max\{b(x),a(x)\}}\Big) &\in [-1,+1] \\
a(x) &= \frac{1}{n_i-1}\sum_{y\in C_i, y\neq x} d(x,y) \\
b(x) &= min_{j,j\neq i}\frac{1}{n_j}\sum_{y\in C_j} d(x,y)
\end{aligned}
\tag{1}
$$

Equation 1: Silhouette score, where NC: number of clusters, C_i: cluster i, n_i: number of observations in C_i, and $d(x,y)$ is a distance measure between two points.

The Silhouette score is especially useful when looking at the effect of breaking up neighboring subclusters (e.g., breaking up a cluster of bank statements into subclusters of statements from different banks), as it is maximized by keeping close subclusters as a large cluster.

$$\frac{\sum_i n_i d^2(c_i, c)/(NC-1)}{\sum_i \sum_{x \in C_i} d^2(x, c_i)/(n-NC)} \in [0, \infty] \tag{2}$$

Equation 2: Calinski-Harabasx index, where NC: number of clusters, C_i: cluster i, c_i: center of i, c: center of all data points, n_i: number of observations in C_i, n: number of data points, and $d(x, y)$ is a distance measure between two points.

In measuring within-cluster tightness, the Calinski-Harabasx index uses the mean distance measure between all points, while the Silhouette score uses a variance-like measure. The Calinski-Harabasx index is thus more sensitive to outliers within the cluster.