Clean your desk! Transformers for unsupervised clustering of document images

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

Abstract

I explore the task of unsupervised document clustering. Given an assortment of documents, the objective is to cluster similar documents together, where similarity accounts for factors like visual appearance, layout, and semantic meaning. Toward this goal, I obtain document embeddings using two relatively new multi-modal Transformer-based encoders: LayoutLM and LayoutLMv2. I test these models on three datasets: a sample of RVL-CDIP documents, SROIE receipts, and a collection of machine learning papers. I find that LayoutLMv2 generally outperforms LayoutLM, although LayoutLM may have advantages for text-heavy documents. Surprisingly, I also find that the `[CLS]` *token output is not always the best document representation, at least in the context of clustering. Code is available at* https://github.com/poojasethi/visual-doc-clustering.

1. Introduction

Suppose a busy scientist had a mixture of documents on her desk, containing scientific papers, bank statements, and receipts from restaurants she last ordered takeout from. Though not her favorite task, it would be no trouble at all for her to organize and separate the documents on her desk into three clusters: one for the papers, one for the statements, and one for the receipts. She may even divide the documents into further subclusters, e.g., distinguishing CVPR from ACL papers.

In this work, I explore the task of *unsupervised document clustering*. Given an assorted set of document images as input, the task is to group similar documents together into one or more clusters. Unsupervised document clustering, unlike document classification, does not require an a priori set of document labels; this makes it a useful task to explore for real-world document understanding applications.

Document understanding is inherently multimodal: when the busy scientist is organizing her documents, she is not just reading the text, but also picking up on many important non-verbal cues such as page structure, images, color, font, and document length; she then uses these features to decide which documents should be grouped together. Thus, a strong document representation should likely incorporate both text and visual features.

2. Related Work

Prior work in document understanding has typically focused on text-only understanding or vision-only understanding, but not the combination of the two.

CNNs, including notable architectures such as AlexNet [17] and ResNet [12] have proven to be successful at a wide range of object detection and facial recognition tasks, but are less commonly applied towards document-related tasks. The notable exception is Optical Character Recognition (OCR) [24], but this focuses on recognizing constituent parts of a document as opposed to holistic document understanding.

On the NLP side, Transformer-based [26] models such as BERT [7] have proven to be extremely effective at tasks such as document classification, entity extraction, and question answering, but still entirely leave out visual features. Transformer-based architectures, such as Vision Transformers (ViT) [8], have been adopted for vision tasks and shown to have competitive performance with CNNs, but leave out textual features.

Only recently has work begun to bridge the divide between the language and visual modalities and applying them towards document understanding. A notable example is LayoutLM [30], which extends BERT by adding 2-D positional embeddings, but otherwise still only uses text input. In follow-up work, LayoutLMv2 [32] and LayoutLMv3 [13] also add *visual* token embeddings as input as well as new text-image alignment and text-image matching pretraining tasks. Other similar and notable work towards Visually-rich Document Understanding include DocFormer [4] and LAMBERT [10].

The authors of the above papers finetune and test Transformer-based models for tasks such as document classification and entity extraction, but to my knowledge no

work has examined the use of such models to obtain general-purpose document embeddings. In NLP, embeddings are typically only obtained at the word, sentence, or paragraph level, and such embeddings do not include visual or layout information [18, 20, 23]. Thus, the key contribution of this work is to evaluate multi-modal Transformer-based architectures, namely LayoutLM and LayoutLMv2, for obtaining document embeddings, using unsupervised document clustering as the evaluation task.

3. Methods

More formally, we can define the task of unsupervised document clustering in two key steps:

1. **Document Embedding** Given a document d_i, preprocess it into its constituent text, bounding boxes, and images. Then, using a model m_{encoder}, return its embedding e_{di}.
2. **Document Clustering** Given the embeddings e_{di} for N documents, use a model m_{cluster} to divide them into k clusters, such that each cluster is at least size 1 but no larger than N.

For the encoder model m_{encoder}, I first tested text-only and vision-only baselines:

- **Bag-of-Words (BoW)** For this naive but surprisingly effective baseline, I encoded the text (only) using sklearn's CountVectorizer [21], which I reduced to a maximum size of 1024 using PCA.
- **ResNet-18** Using a pretrained model from Image2Vec [6], I encoded the image (only), obtaining a 512-dimensional embedding.
- **AlexNet** Using a pretrained model from Image2Vec [6], I encoded the image (only), obtaining a 4096-dimensional embedding which I reduced to a maximum size of 1024 using PCA.

Following these baselines, I tested several Transformer-based document encoding models, each trained with one or more of text, layout (position of the text), and visual features. From the last layer of a given Transformer-based model, I obtained token embeddings, i.e. last layer hidden states, of shape $(\text{maximum sequence length}, \text{embedding dim})$. On top of this, I implemented several methods to select or pool the token embeddings into a single document embedding of shape $(\text{embedding dim},)$. I leveraged pretrained LayoutLM and LayoutLMv2 models released on HuggingFace to get the token embeddings [29, 31].

For the clustering model m_{cluster}, I implemented a Gaussian Mixture Model (GMM) using the sklearn library. Finally, I implemented a t-SNE [25] visualization to understand the cluster assignments.

3.1. LayoutLM

The base LayoutLM model is very similar to the BERT [7] architecture, but with two key differences: namely, it introduces and adds 2-D positional embeddings and it replaces the Masked Lanugage Modeling (MLM) and Next Sentence Prediction (NSP) training objectives with MVLM and MDC, respectively.

3.1.1 Model Inputs

Before a sequence of text is encoded by BERT, the text is broken into tokens using WordPiece [27]. For each token, its token embedding, position embedding, and segment embedding are then summed together before being passed to the encoder. LayoutLM, to this sum, also adds 2-D position embeddings. In total, it has three learnable positional embedding lookup tables: one for the 1-D positions, one for the 2-D x positions, and one for the 2-D y positions.

The 2-D positional embeddings represent the spatial location of a word on the page. When a document is preprocessed, OCR returns the bounding box for each word, represented by a top-left (x_0, y_0) and bottom-right coordinate (x_1, y_1). The positions are normalized and discretized such that the x and y coordinates are integer values, each ranging from $[0, 1000]$. Finally, per-token, four 2-D position embeddings are also looked up and added, one for each of x_0, x_1, y_0, y_1.

3.1.2 Training Objectives

Masked Visual-Language Model (MVLM) In this task, some of the input tokens are randomly masked but their corresponding 2-D positions are kept. The task is then to predict the masked tokens. The intuition is that this helps the model learn the relationship between word and spatial position.

Multi-label Document Classification (MDC) In this task, the `[CLS]` token is used to predict which of (multiple) possible document tags a given document has. Supervision is obtained from the document tags given in the IIT-CDIP Test Collection (containing 11 million document images), which is the dataset LayoutLM is pretrained on.

3.2. LayoutLMv2

The main advantage of LayoutLMv2 over LayoutLM is that also takes in visual features as input, not just text and positional information. In addition, it modifies the Transformer self-attention mechanism to be spatially aware and replaces the MDC task with TIA and TIM.

3.2.1 Model Inputs

LayoutLMv2, to the text embeddings, also concatenates visual token embeddings. To each of them, it then adds 1-D position, 2-D position, and segment embeddings. The inputs to the model are illustrated in Figure 1.

Visual Embedding To obtain visual token embeddings, the document image is resized to 224 x 224 and then encoded using a ResNeXt-FPN model [28]. The output feature map is average-pooled to size 7 x 7 (or the configured W and H) and then flattened to a sequence of length 49. Finally, a linear projection layer is applied to make the dimensionality of the visual token embeddings match the text token embeddings. All the visual tokens are assigned to a special segment `[C]`.

Layout (2-D Position) Embedding LayoutLMv2 also uses 2-D positional embeddings, with two key differences. First, it also represents the *height* and *width* of the bounding box. Second, it concatenates all of the 2-D positional embeddings together instead of adding them.

$$\mathbf{x_{0i}}, \mathbf{x_{1i}} = PosEmb2D_x(x_0), PosEmb2D_x(x_1)$$
$$\mathbf{y_{0i}}, \mathbf{y_{1i}} = PosEmb2D_y(y_0), PosEmb2D_y(y_1)$$
$$\mathbf{h_i} = \mathbf{y_{1i}} - \mathbf{y_{0i}}$$
$$\mathbf{w_i} = \mathbf{x_{1i}} - \mathbf{x_{0i}}$$
$$\mathbf{l_i} = Concat(\mathbf{x_{0i}}, \mathbf{x_{1i}}, \mathbf{y_{0i}}, \mathbf{y_{1i}}, \mathbf{h_i}, \mathbf{w_i})$$

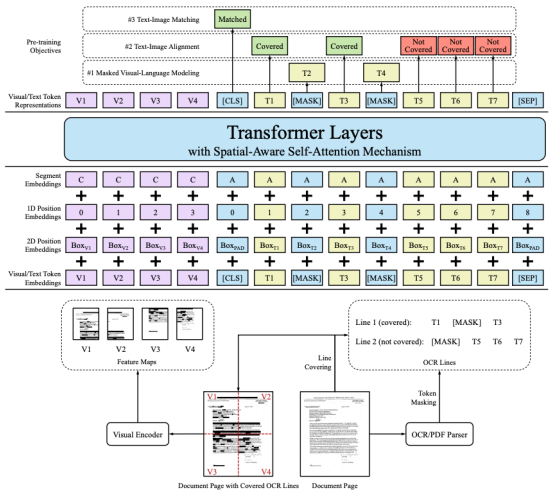

Figure 1. The LayoutLMv2 architecture, taken from [32], illustrating the visual and text inputs. Note that the diagram slightly misrepresents the inputs since the visual tokens are actually appended, not prepended, to the text tokens.

3.2.2 Spatial-Aware Self-Attention

The authors of LayoutLMv2 point out that the original Transformer self-attention mechanism [26] only implicitly captures the relationship between input tokens and their positions. To model local invariance in the document layout more efficiently, LayoutLMv2 also adds learnable biases to the attention scores that capture the relative difference in position between two tokens.

More formally, we have input tokens $\mathbf{x_i}$, where $0 \leq i \leq L + WH$ (L for the number of text tokens and WH for the number of image tokens).

As in the original Transformer paper, each self-attention head still computes an attention score between each query ($\mathbf{x_i W^Q}$) and key ($\mathbf{x_j W^K}$) using the below equation:

$$\alpha_{ij} = \frac{\mathbf{x_i W^Q} (\mathbf{x_j W^K})^T}{\sqrt{d_{\text{head}}}}$$

The change that LayoutLMv2 makes is to also add learnable bias terms for the 1-D and 2-D positions. The difference between the respective biases is taken when computing the new attention score. Letting (x_i, y_i) denote the top-left corner of the bounding box of $\mathbf{x_i}$, and (x_j, y_j) the top-left corner of $\mathbf{x_j}$, we compute the new spatial-aware attention score as follows:

$$\alpha'_{ij} = \alpha_{ij} + \mathbf{b}^{(1D)}_{j-i} + \mathbf{b}^{(2D_x)}_{x_j - x_i} + \mathbf{b}^{(2D_y)}_{y_j - y_i}$$

Finally, the output $\mathbf{h_i}$ of an attention head for a given $\mathbf{x_i}$ is the weighted average of value vectors, where the weights are calculated by taking the softmax over the attention scores:

$$\mathbf{h_i} = \sum_j \frac{\exp(\alpha'_{ij})}{\sum_k \exp(\alpha'_{ik})} \mathbf{x_j W^V}$$

3.2.3 Training Objectives

LayoutLMv2 uses the same MVLM objective as LayoutLM. However, MDC is replaced by TIA and TIM, eliminating the necessity of labels during pretraining.

Text-Image Alignment (TIA) In TIA, some lines of tokens are randomly selected and their corresponding image regions are covered in the document image. The term "covered" is used to avoid confusion with masking of text done by MVLM. When LayoutLMv2 was pretrained, a token-level classification head was added to the model and used to predict whether a given token is covered or not covered. The binary cross-entropy loss is then computed. If the token is `[MASK]`, then it is not included in the loss for TIA.

Text-Image Matching (TIM) In TIM, the `[CLS]` token is used to predict whether an image and text are from the same page of a document. Negative examples are constructed by dropping the image for a page, or swapping it with an image from another document. The binary cross-entropy loss is then computed.

3.3. From Token to Document Embeddings

Finally, I tested several methods for combining the token-level output embeddings returned by LayoutLM and LayoutLMv2 into a single document embedding. Namely, I tested using the `[CLS]` token only, `[SEP]` token only, averaging all the token embeddings (including image tokens if applicable, but always excluding the `[SEP]`, `[CLS]`, and `[PAD]` tokens), and averaging all the image token embeddings (for LayoutLMv2).

4. Dataset

I prepared three datasets in total. Though to my knowledge, there are no benchmarks for unsupervised document clustering, I created my own by sampling 626 document images from SROIE2019 [14], a dataset of scanned receipts, and 1,000 document images from RVL-CDIP [11], a dataset of assorted documents like newsletters, invoices, emails, handwritten notes, etc., both of which are also used by LayoutLM authors [30, 32] for evaluating token and document classification performance. Example documents can be seen in Figure 2. Because document clustering is unsupervised, I discarded the existing labels that came with the document images. For this reason, I also did not need to have separate train/validation/test splits.

In addition to the above two *unlabeled* datasets, I created my own *labeled* dataset. I sampled 20 papers from MLSys 2022 and 10 papers from NeurIPS 2021 conference proceedings. I chose papers from these conferences in particular because they use similar LaTeXtemplates but with slight, visually discernible differences. I assigned each paper one of three labels, as shown in Figure 3. I used the labels solely to evaluate how well the predicted clusters matched my expectations, and did not use them to train m_{cluster}.

Notably, every document encoder, m_{encoder}, was pretrained on separate datasets. The three datasets I prepared were used for inference on the encoder to obtain its hidden states (token embeddings), pooled into document embeddings e_{d_i}, and then used to fit the clustering model m_{cluster}.

For simplicity, if a document had more than one page, I only used the first one. The document text was truncated or padded with the `[PAD]` token such that all sequences had the same length of 512 tokens.

Preprocessing of the text was done using Impira [15], which I used to apply OCR to the documents and get their text and bounding boxes. Preprocessing of the images was done using the LayoutLMProcessor on HuggingFace. Each

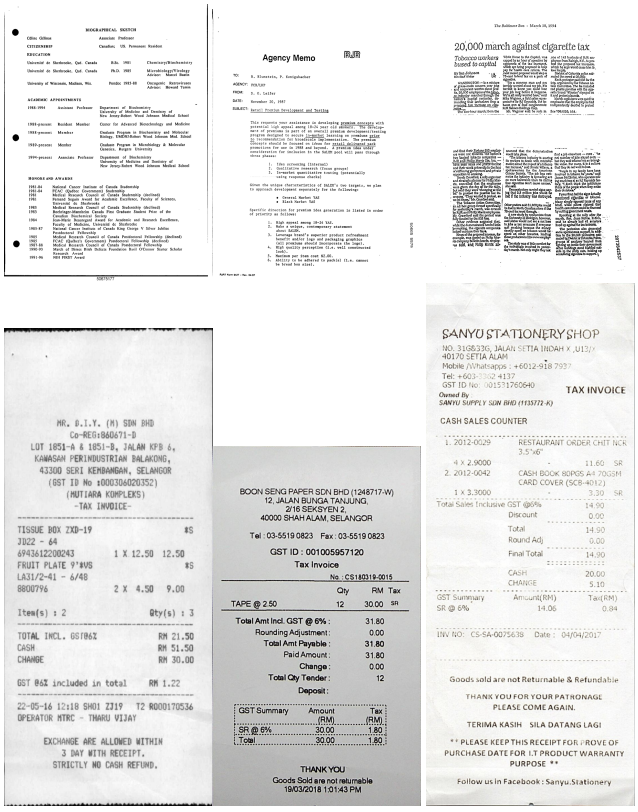

Figure 2. Examples of document images from the unlabeled datasets. The first row shows examples of documents from RVL-CDIP, such as a resume, memo, and news article. The second row shows a sample of receipts from SROIE2019.

image was resized to 224 x 224, average pooled to a 7 x 7 feature map, and flattened into a sequence of 49 tokens.

5. Results and Discussion

I tested two pretrained LayoutLM models: the `BASE` model, which has 12 layers, 12 self-attention heads, hidden state size of 768, and 113M parameters; and the `LARGE` model, which has 24 layers, 16 self-attention heads, hidden state size of 1024, and 343M parameters. These architectures were chosen so that LayoutLM could be initialized with the weights from correspondingly-sized pretrained BERT models. Both the LayoutLM base and large models were further trained on 11 million document images for 2 epochs. The batch size was set to 80. Optimization was done with Adam with a learning rate of 5e^{-5} and a linear decay learning rate schedule.

Similarly, I tested two pretrained LayoutLMv2 models: a `BASE` model and a `LARGE` model, both of which had the same number of layers, self-attention heads, and hidden state sizes as the corresponding LayoutLM models. The LayoutLMv2 models were initialized and trained in similar

Figure 3. The papers on the top row are from the MLSys 2022 conference proceedings [9, 22]. To these papers, I either assigned the label `mlsys_no_stamps` (e.g. top-left) or `mlsys_stamps` (e.g. top-right, which has green and red circular stamps on the page corner). The papers on the bottom row are from the NeurIPS 2021 conference proceedings [5, 19]. To all of these papers, I simply assigned the label `neurips`. The MLSys papers are two-column and have smaller margins; the NeurIPS papers are one column, have wider margins, and slightly bolder headers.

fashion to the LayoutLM models. However, the base model was trained with a batch size of 64 for 5 epochs, while the large model was trained with a batch size of 2048 for 20 epochs. Optimization was done with Adam, with a learning rate of 2e^{-5}. The learning rate was linearly warmed up over the first 10 steps then linearly decayed.

For the clustering model m_{cluster}, I set the number of clusters k equal to 10 for SROIE-2019 and RVL-CDIP and 3 for the ML Papers (since I knew there were 3 distinct types of papers in the dataset).

5.1. SROIE and RVL-CDIP

To evaluate the cluster assignments made by m_{cluster}, I used two unsupervised clustering metrics: the Silhouette and Calinski-Harabasz (CH) scores [2, 3]. Both of these metrics reward clusters that are tight (have low *inter-cluster* distance) and well separated (have high *intra-cluster* distance). The Silhouette score ranges between $[-1, 1]$ and the CH score ranges between $[0, \infty]$. For both metrics, a higher score is better. A negative Silhouette score indicates a document was assigned to the wrong cluster.

The results obtained by testing different m_{encoder} models and token embedding pooling strategies are shown in Table 1. LayoutLMv2 Base most often led to the best results, as indicated in bold. On SROIE, it led to the highest Silhouette and CH scores. On RVL-CDIP, it led to the highest CH score. These results indicate that multi-modal learning of text, layout, and visual features is helpful for document clustering.

Table 1. Unsupervised Clustering Results with No Labels
Metrics: (Silhouette Coefficient / Calinski-Harabasx (CH) score)

Method	SROIE ($n = 626$)	RVL-CDIP ($n = 1000$)
Baselines		
Bag-of-Words (BoW)	0.093 / 27.5	0.134 / 90.3
ResNet-18	0.101 / 69.888	0.047 / 57.977
AlexNet	0.116 / 75.764	0.070 / 69.955
LayoutLM Base		
`[CLS]` token	0.155 / 88.406	0.155 / 107.911
`[SEP]` token	0.194 / 378.588	**0.248** / 118.642
Average all tokens	0.128 / 41.326	0.055 / 42.778
LayoutLM Large		
`[SEP]` token	0.156 / 44.062	0.057 / 36.746
LayoutLMv2 Base		
`[CLS]` token	0.127 / 212.757	0.131 / 223.778
`[SEP]` token	0.150 / 84.804	0.091 / 126.566
Average image tokens	**0.281** / **713.625**	0.187 / **666.915**
Average all tokens	0.177 / 366.174	0.130 / 262.591
LayoutLMv2 Large		
Average image tokens	0.079 / 78.228	0.056 / 96.490

The notable exception to this is that LayoutLM Base outperformed LayoutLMv2 Base on RVL-CDIP when measured by the Silhouette score (0.248 vs. 0.187). Similarly, another interesting pattern emerges when looking at the results from the baselines. We see that the CNNs outperform Bag-of-Words on SROIE, but that Bag-of-Words outperforms the CNN encoders on RVL-CDIP. These results suggest that the best document encoding model might be dataset dependent. For text-heavy documents, such as those in RVL-CDIP, textual, semantic understanding may be more important than visual understanding. For other documents like receipts, which often contain large headers, additional visual understanding is beneficial.

Surprisingly, both the LayoutLM and LayoutLMv2 large models significantly underperformed the smaller base mod-

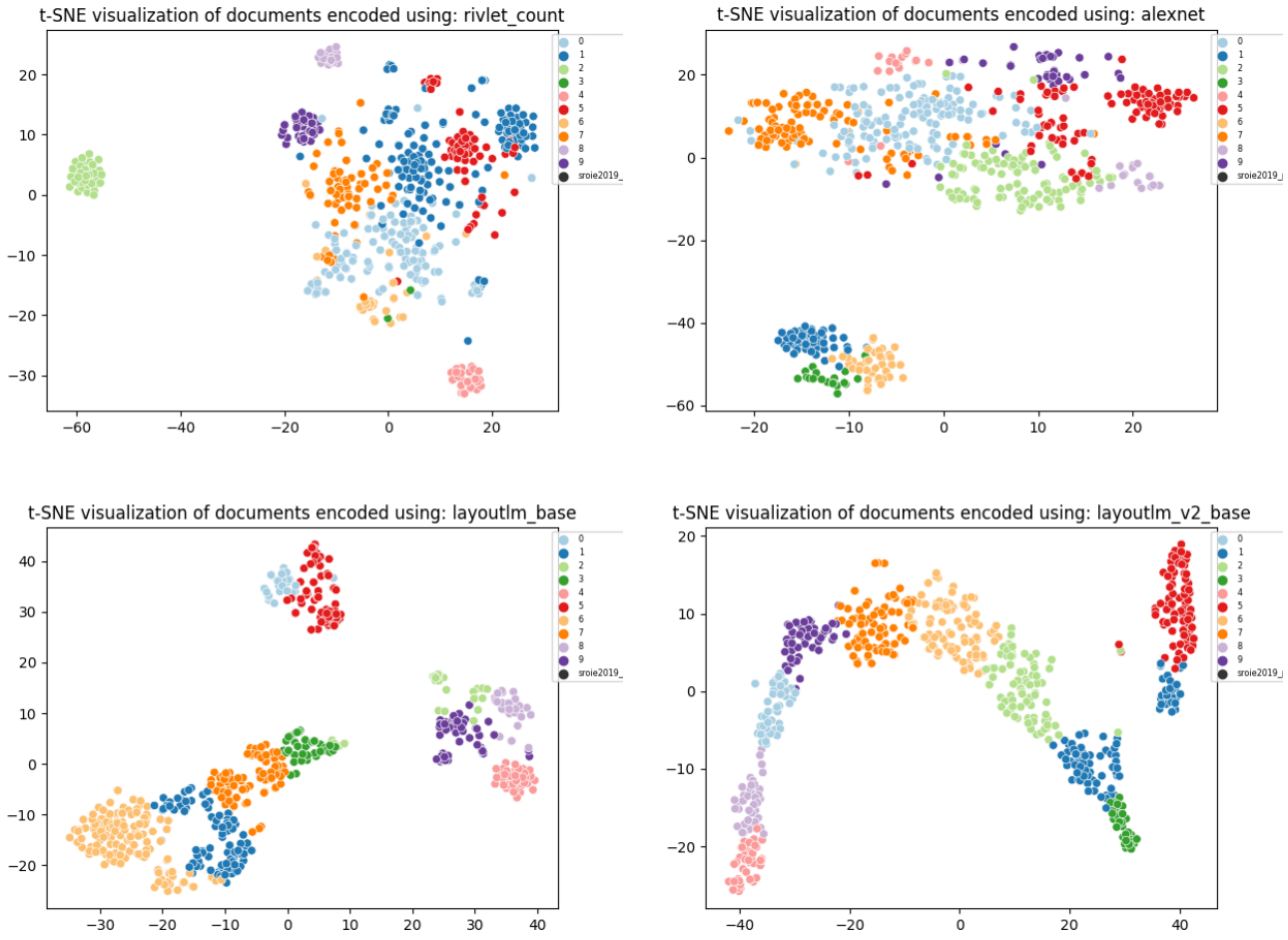

Figure 4. These t-SNE plots show document embeddings from the four m_{encoder} models on the SROIE dataset. In quadrant order: Bag of Words (BoW), AlexNet, and LayoutLM (Base), and LayoutLMv2 (Base). Each point corresponds to the first page document embedding. For LayoutLM, I use the `[SEP]` output hidden state and for LayoutLMv2 I average all of the image token ouput hidden states. Each color represents a predicted cluster. The clusters get progressively tighter and better separated.

els. The large models return token embeddings of size 1024, while the base models returns token embeddings of size 768. A possible explanation is that the larger dimensional embeddings are typically more difficult to cluster than lower dimensional ones. Another is that the large models may not have converged during training as well as the small models. For example, LayoutLMv2 was trained with a relatively large batch size of 2048. Some have found that large-batch training can lead to generalization gaps [16].

Finally, I also tested various strategies for pooling the token embeddings returned by the LayoutLM* models into a single document embedding. While I expected the hidden state corresponding to the `[CLS]` to lead the best results, since this token is used during the pretraining procedure for document-level prediction tasks, surprisingly, the hidden state corresponding to the `[SEP]` token (marking the end of the text) led to even better results for LayoutLM. This might indicate that the `[CLS]` token hidden state is "overfitting" to perform well on the pretraining tasks, and that perhaps a different task is needed to encourage it to learn a general-purpose document embedding. For LayoutLMv2, taking the average of the hidden states corresponding to the image tokens led to the best results.

To qualitatively evaluate the clusters, I compressed the document embeddings into two dimensions using t-SNE and plotted them, as shown in Figure 4. The visualization illustrates that stronger encoding methods lead to noticeably tighter and better separated clusters. Probing a particular cluster, we can see that the documents within it are indeed visually similar, as show in in Figure 5.

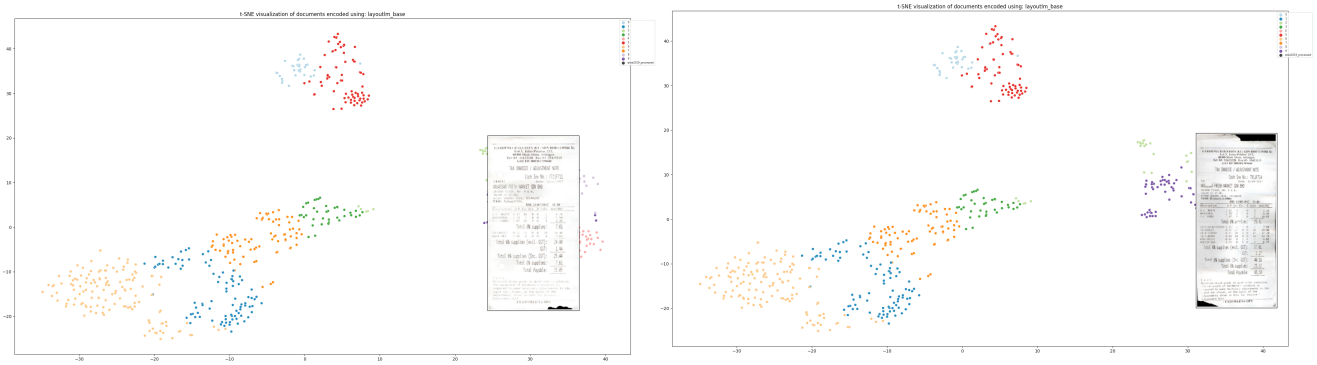

Figure 5. Zooming into the light pink cluster from the LayoutLM base model in Figure 4, we can inspect the documents within it. The left and right plots show that two different receipts from Gardenia Bakery have been clustered together.

5.2. ML Conference Papers

Next, I evaluated the cluster quality on my hand-curated dataset of document images from ML papers. I compared how the predicted clusters fared against my expectations of how the documents should be clustered, hoping that the m_{cluster} predictions 0, 1, and 2, would correspondingly contain the documents from `mlsys_no_stamps`, `mlsys_stamps`, and `neurips` (not necessarily in that order). I matched each cluster to a label in way that would maximize the overall accuracy. The problem of matching clusters to labels in a way that maximizes accuracy can be framed as Linear Sum Assignment, which is used to do minimum weight matching of bipartite graphs [1]. Using this approach, I matched each cluster to a label and calculated the accuracy of the clustering model, as well as the precision and recall per label, as shown in Table 2.

Table 2. Unsupervised Clustering Results with Labels
Metrics: P/R (per-class); Accuracy (overall)

Method	MLSys (No Stamps) ($n = 10$)	MLSys (Stamps) ($n = 10$)	NeurIPS ($n = 10$)	
LayoutLM	.33/.20	.54/.70	.55/.60	**0.50**
LayoutLMv2	.40/.40	.41/.70	.67/.20	0.43

I used the base versions of LayoutLM and LayoutLMv2. To pool the token embeddings into a document embedding, I used the `[SEP]` token hidden state for LayoutLM and the average of all the image token hidden states for LayoutLMv2.

LayoutLM outperformed LayoutLMv2, obtaining an accuracy of 50% vs. 43%. This result does align with the previous experiments where I found that for text-heavy documents (RVL-CDIP), Bag-of-Words and LayoutLM (at least on some measures) outperformed the CNNs and LayoutLMv2.

This being said, it was still surprising that LayoutLMv2 did not fare better. There are obvious visual cues that indicate which group a paper should belong to (e.g., margin size, boldness of text, one vs. two columns, brightly colored stamps). In addition, there are obvious textual clues as well (e.g., the footer of the documents include the name of the conference).

6. Conclusion

In this work, I explored how to cluster together related document images, in an unsupervised fashion. I encoded documents using an encoding model, used the outputs of the encoding model to obtain a document embedding, and finally, clustered the document embeddings. I ran experiments on three different sets of document images: a sample of 1,000 images from RVL-CDIP, 626 images from SROIE, and a hand-curated dataset of 30 machine learning conference papers. I tested two Transformer-based architectures, LayoutLM and LayoutLMv2, as the encoding model, and found that they both showed strong performance over the baselines. LayoutLMv2, likely due to its ability to also learn from visual tokens in addition to text, tended to outperform LayoutLM. However, LayoutLM may still be a better choice for clustering text-heavy documents.

I also tested various models for selecting or combining the token embeddings returned by LayoutLM or LayoutLMv2 into a single document embedding. One surprising takeaway was that the `[CLS]` embedding output, which is often used as the default choice as an overall document representation, wasn't always the best for clustering. For LayoutLM, using the `[SEP]` output was best, and for LayoutLMv2, averaging the outputs corresponding to the image tokens was best. The reason for this is unclear. One could visualize the attention maps corresponding to the `[SEP]` and `[CLS]` outputs to get a sense of their differences.

Finally, an interesting future direction may also to be al-

low a human (or model) to select the most salient parts of the document image and text that should be used for clustering. Masking out the "unnecessary" details of a document before feeding into into an encoder could lead to better clusters. For example, to cluster the machine learning papers, one could select the title, footer, and abstract and black out any remaining text.

7. Acknowledgements

I would like to thank the Impira team for providing the software used for preprocessing the document images and many invaluable discussions related to this work. Thanks to Bryan Chia for his ideas and contributions to an early version of this project for CS224N. Finally, thanks to the Spring 2022 CS231N teaching staff for their work on this course.

References

[1] Accuracy: from classification to clustering evaluation. Available at https://smorbieu.gitlab.io/accuracy-from-classification-to-clustering-evaluation/. 7

[2] Calinski-harabasz index. Available at https://scikit-learn.org/stable/modules/clustering.html#calinski-harabasz-index. 5

[3] Silhouette coefficient. Available at https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient. 5

[4] Srikar Appalaraju, Bhavan Jasani, Bhargava Urala Kota, Yusheng Xie, and R. Manmatha. Docformer: End-to-end transformer for document understanding. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 973–983, 2021. 1

[5] Aljavz Bovzivc, Pablo Rodríguez Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. In *NeurIPS*, 2021. 5

[6] Christiansafka. Christiansafka/img2vec: Use pre-trained models in pytorch to extract vector embeddings for any image. 2

[7] 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. 1, 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2021. 1

[9] Pratik Fegade, Tianqi Chen, Phillip B. Gibbons, and Todd C. Mowry. The cora tensor compiler: Compilation for ragged tensors with minimal padding. *ArXiv*, abs/2110.10221, 2022. 5

[10] Lukasz Garncarek, Rafal Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał P. Turski, and Filip Grali'nski. Lambert: Layout-aware language modeling for information extraction. In *ICDAR*, 2021. 1

[11] 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)*. 4

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1

[13] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking. *ArXiv*, abs/2204.08387, 2022. 1

[14] 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. 4

[15] Impira, 2022. 4

[16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *ArXiv*, abs/1609.04836, 2017. 6

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012. 1

[18] 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. 2

[19] Shasha Li, Abhishek Aich, Shitong Zhu, M. Salman Asif, Chengyu Song, Amit K. Roy-Chowdhury, and Srikanth V. Krishnamurthy. Adversarial attacks on black box video classifiers: Leveraging the power of geometric transformations. In *NeurIPS*, 2021. 5

[20] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013. 2

[21] 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. 2

[22] James K. Reed, Zach DeVito, Horace He, Ansley Ussery, and Jason Ansel. torch.fx: Practical program capture and transformation for deep learning in python. *ArXiv*, abs/2112.08429, 2022. 5

[23] 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. 2

[24] Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference*

on Document Analysis and Recognition, 2003. Proceedings., pages 958–963, 2003. 1

[25] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 2

[26] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017. 1, 3

[27] Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016. 2

[28] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017. 3

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

[30] Yiheng Xu et al. Layoutlm: Pre-training of text and layout for document image understanding. In *Association for Computing Machinery (ACM)*, 2020. 1, 4

[31] Yang Xu et al. Layoutlmv2. 2021. Available at https://huggingface.co/docs/transformers/v4.16.2/en/model_doc/layoutlmv2. 2

[32] Yang Xu et al. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. In *Association for Computational Linguistics (ACL)*, 2021. 1, 3, 4