Paragraph Segmentation Using Semi-Supervised Deep Clustering Network

Soujanya Basangari , **Toralben Arvindbhai Davara** , **Sai Manasa Tanniru** , **Varuni Gururaja** and **Vidyasagar Aithal Radhakrishna**
University of Koblenz-Landau
{sbasangari, torald, tmanasa, varunig, aithalrvs10}@uni-koblenz.de

Abstract

Extracting information from PDF documents has always been a challenging task. PDF being a layout-based format: the positions and fonts are predefined of the individual characters, of which text is composed. Deriving the beginning and end of the paragraph is crucial for reflow applications and rearranging the text having varied such features like font size, line length and spacing is a difficult task as PDFs are rendered by various tools such as Microsoft Office, Adobe Acrobat, and LaTex. All of the above-mentioned tools have their own rendering techniques and content is written and formatted by different people. There are several freely available tools to extract the information from the PDF documents. But, the variety is so confusing and there does not seem to be a clear winner. In this paper, we present a method to extract information from the PDF documents and provide the segmented paragraphs as an output. In our application, we used a combination of Deep Neural Networks(DNNs) which is built on a three-layered auto-encoder architecture along with the clustering algorithm. We trained our model with a synthetically generated labeled dataset from randomly extracted metadata records from ArXiv On Kaggle[1] containing around 1.7 million articles and unlabeled data scraped from the web. To this end, we built a semi-supervised model by training with a small number of labeled and large sets of unlabeled datasets. Our method achieved an average accuracy of around 99.9% which validates its capability to accurately extract the information from a variety of PDFs in an orderly manner.

1 Introduction

PDF continues to be one of the most popular and powerful electronic document formats. Google alone indexes over 3 billion PDF documents more than any other format except HTML. Extracting information from the PDF is a challenging task. Unfortunately, PDF is a layout-based format: it specifies the positions and fonts of the individual characters, of which the text is composed. Applications usually require information about the semantic building blocks of text(e.g., the sentences and the divisions into paragraphs and segments) and their semantic roles(e.g., whether a sentence is a part of the body or of a footnote). This semantic information is not provided as a part of the PDF. And above all the PDF is usually rendered by various tools such as Microsoft Office, Adobe Acrobat, and LaTeX. All of these tools have their own rendering techniques. Moreover, content is written and formatted by people. All of these factors make PDF documents very complex with text, images, graphs, and tables. Deriving the beginning and end of the paragraph is again crucial for reflow applications and rearranging the text having varied such features like font size, line length and spacing is a difficult task. To this end, a large number of tools for text extraction from PDF is available. A google query for text extraction from PDF provides countless hits with tools or pages recommending tools for this task. The variety is confusing and there does not seem to be a clear winner. Most tools do not specify which of the aspects above are useful. The majority of the tools use a similar "Visual" technique to identify semantically important areas of a PDF or they do word identification and consider word order. Only the more sophisticated tools provide paragraph boundaries and semantic tools. So far, there has been no rigorous benchmark for this problem and no comprehensive evaluation of the existing systems. This is surprising, given the practical importance of the problem, but it also hints at the complexity.

In recent years, motivated by the success of deep neural networks (DNNs) in fully - supervised learning, unsupervised, and semi-supervised deep learning approaches are now widely used for dimensionality reduction and understanding hidden patterns in many of the domains. Fully supervised approaches [12], [18] perform well but are often limiting due to the unavailability of sufficient labeled data. On the other hand, approaches like [23], [7] are unsupervised and rely on unlabeled data alone for learning the network parameters, assuming only the knowledge of the number of clusters. Fully unsupervised approaches are oblivious of any semantic notions of data, and may lead to learning representations that are difficult to interpret and analyze. Weakly or semi-supervised approaches are often used as a reasonable

[1] https://www.kaggle.com/Cornell-University/arxiv

middle-ground between the two extremes. For example, [17] uses pairwise constraints as a weaker form of supervision, and [10] exploit both, labeled as well as unlabeled data for learning. For example, the stacked autoencoder (SAE) [19], Deep Canonical Correlation Analysis (DCCA) [1], and Sparse Autoencoder [16] take insights from Principal Component Analysis (PCA), Canonical Correlation Analysis (CCA), and sparse coding, respectively, and make use of DNNs to learn nonlinear mappings from the data domain to low-dimensional latent spaces. These approaches treat their DNNs as a preprocessing stage that is separately designed from the subsequent clustering stage. We take inspiration from recent work on deep learning for computer vision [10], [6], [25], [11], where clear gains on benchmark tasks have resulted from learning better features. Examples for semi-supervised approaches are [8] uses pairwise constraints as a weaker form of supervision, and [6] exploits both, labeled as well as unlabeled data for learning.

Hence, in this paper, we propose a semi-supervised deep learning method with clustering to extract paragraph segments from a PDF document. Our application is an autoencoder based architecture that learns latent representations suitable for clustering. The training works in a semi-supervised setup utilizing the majority of unlabeled data augmented with a very few labeled dataset. To this end, we automatically generate a corpus from HTML. Furthermore, in the pre-processing step, we feed this generated corpus as an input and get around 30 hand-crafted features of each line of text. Additionally, we manually annotate each line of text as the beginning of the paragraph, regular lines, and other lines(includes headers and footers). Subsequently, we have fine-tuned an implementation of an autoencoder which gets trained on the unseen and unlabelled data to generate learned features. These features are fed into the clustering model to get the clusters. Later, this clustered dataset is segmented to get the orderly segmented paragraphs.

The rest of the paper is organized as follows. Section II gives an overview of some related work in paragraph segmentation for PDF documents. Section III describes the background. Section IV explains about the data generation and describes our methodology with pre-processing, network architecture, model, loss functions, and training phase. Section V reports on our experimental results. Section VI presents our conclusion.

2 Related Works

Identifying the structure of an electronic document is a well-known research problem. Some methods have been presented in the literature to perform the segmentation task. However, most of these methods rely on image preprocessing such as binarization, connected components (CCs) extraction, off-the-shelf classifiers trained on hand-crafted features, and prior knowledge. Most of the solutions are based on the layout features such as font-size and, text-indentation [2], [13]. Mao et al. provide a detailed survey of the physical layout and logical structure analysis of document images. According to them, document style parameters, such as the size of and gap between characters, words, and lines are used to represent document physical layout. Algorithms used in physical layout analysis can be categorized into three types: top-down, bottom-up, and hybrid approaches.

O'Gorman's Docstrum algorithm [17], the Voronoi diagram- based algorithm of Kise [9], and Fletcher's text string separation algorithm [5] is bottom-up algorithms. Gorman et al. describe the Docstrum algorithm using the K-NN for each connected component of a page and use distance thresholds to form text lines and blocks. Kise et al. propose the Voronoi-diagram-based method for document images with a non-Manhattan layout and skew. The X-Y cut algorithm presented by Nagy et al. [15] is an example of the top-down approach based on recursively cutting the document page into smaller rectangular areas. A hybrid approach presented by Pavlidis et al. [20] identifies column gaps and groups them into column separators after horizontal smearing of black pixels.

Hui Chao et al. [3] describe an approach that automatically segments a PDF document page into different logical structure regions, such as text blocks, images blocks, vector graphics blocks, and compound blocks, but does not consider continuous pages. Djean et al. [4] present a system that relies solely on PDF-extracted content using a table of contents (TOC). But many documents may not have a TOC. Muhammad Mahbubur Rahman et al. [21] [22] illustrate a method automatically identifies and classifies different sections of documents and understands their purpose within the document with multiple machine learning models including deep learning architectures for classification and semantic annotation.

Maria F. De La Torre et al. [14] elucidate a method called MATESC(Metadata-Analytic Text Extractor and Section Classifier) for PDF scientific publications. MATESC extracts text spans and uses metadata features such as spatial layout location, font type, and font size to create grouped blocks of text and classify them into groups and subgroups based on rules that characterize specific paper sections. The main purpose of this tool is to facilitate information and semantic knowledge extraction across different domain topics and journal formats. Lloyd Alan Fletcher et al. [5] describes the development and implementation of a new algorithm for automated text string separation which is relatively independent of changes in text font style and size, and of string orientation. The algorithm outputs two images, one containing text strings, and the other graphics. These images may then be processed by suitable character recognition and graphics recognition systems. Xiao Yang et al. [24] proposed a model that extracts the document semantic structure as a pixel-wise segmentation task by tuning a unified model that classifies pixels based not only on their visual appearance, as in the tradition page segmentation task, but also on the content of the underlying text.

Most of the earlier works addressed the problem of extract-

ing the content from the PDF documents using supervised or semi-supervised models. But there are some of the works which have taken an unsupervised feature learning approach for page segmentation of electronic documents. Kai Chen et al. [24] present an unsupervised feature learning method for page segmentation of historical handwritten documents available as color images. They consider page segmentation as a pixel labeling problem, i.e., each pixel is classified as either periphery, background, text block, or decoration. Page segmentation problems could be solved using an unsupervised approach along with the clustering technique. Junyuan Xie et al. [23] proposed a Deep Embedded Clustering (DEC) technique, a method that simultaneously learns feature representations and cluster assignments using deep neural networks. DEC learns a mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective.

3 Background

3.1 Deep Neural Networks(DNNs)

Deep neural networks are a powerful category of machine learning algorithms implemented by stacking layers of neural networks along with the depth and width of smaller architectures. Deep networks have recently demonstrated discriminative and representation learning capabilities over a wide range of applications in contemporary years. A key idea in deep learning is to learn not only the nonlinear mapping between the inputs and outputs but also the underlying structure of the data (input) vectors. Nowadays, researchers use DNNs even in the unsupervised and semi-supervised techniques.

3.2 Constrained KMeans

Our work is inspired by the constrained k-means algorithm that utilizes some labeled data to guide the unsupervised k-means clustering. As opposed to random initializations of cluster centers in traditional k-means, labeled samples are used to initialize the cluster centers in constrained k-means. Secondly, at each iteration of k-means the cluster re-assignment is restricted to the unlabeled samples, while the membership of labeled samples is fixed. This procedure of constrained k-means has shown performance improvement over the k-means algorithm.

4 Data Methodology

4.1 Dataset

For our application, we generate a synthetic pdf document as a dataset. The dataset includes various scientific layouts, footnotes, and paragraphs with varied font sizes which allowed us to train our model on a diverse data set and prevent the network from overfitting and memorizing layouts' exact details. To this end, we randomly extracted metadata records from ArXiv On Kaggle[2]. This dataset is a mirror of the original ArXiv data containing 1.7 million articles and provides only a metadata file in JSON format. This JSON file contains an entry for each paper containing ArXiv ID, authors of the paper, title, abstract, and so forth.

Our aim is to generate 1000 pdf documents with more than 200 paragraphs, headers, and footnotes from abstracts, titles, and authors respectively in each of the files. While generating the pdf, we parallelly create a CSV file to store each line of text and annotate them with the labels. We label each line of text as 3 if it is a header/footer, 1 for the starting of the paragraph, and 2 for the rest of the lines. We see that each of the generated pdf contains a unique set of data by randomly rendering the paragraphs, titles, and authors from the article.

We also have generated a pdf document by scraping the web page having numerous pdf documents. We extracted the data from those PDFs and formed a large pdf document. To this end, we successfully generated a labeled dataset using a JSON file from ArXiv containing randomly extracted metadata from various papers and also unlabeled data through web scraping.

4.2 Methodology

In this section, we present our approach: pre-processing unit, network architecture, and segmentation unit. The generated dataset both labeled and unlabeled during the dataset creation phase is sent through the pre-processing phase to generate a set of features for each line of text which is explained elaborately in the pre-processing unit. The output from the pre-processing unit then passed to our model to get the clustered data, which are then organized to form paragraphs based on the clusters.

Pre-processing Unit

There are several python libraries in the market to extract features from the pdf document such as PyPDF2, PDFMiner, etc. But, these libraries do not completely serve our purpose to extract the features of each line of text. We make use of PyMuPDF (a Python binding for MuPDF - "a lightweight PDF and XPS viewer") in our application to extract the layout information and each line of text from the pdf document.

Basically, the PyMuPDF library goes through each page of the document and stores each of its content in a separate block, and also identifies the font-size, text, colors, coordinates, font-weight, and other information. We derive some other features from the above-mentioned features which are best used to get the physical (text) structure of the document. The identified and derived features are then consolidated to form a set of 26 features describing each line of text. The list of features is tabulated below in table 1.

Model

In this section, we present our approach: network architecture, training setup, and segmentation unit. We consider the following setup for our application. Let $X^i = S_{k=1}^{K}\{X_k^l\}$ denote the set of labeled data denote the set of labeled data corresponding to K clusters. Here $X_k^l = \{x_1^l, x_2^l, x_3^l\} \in \mathrm{R}^n$ is the set of labeled data points in k^{th} class. Let X^u denote

[2]https://www.kaggle.com/Cornell-University/arxiv

Serial No.	Feature	Description
1	LefttoRight	Distance from Left to Right of a line of text relative to the page
2	RighttoLeft	Distance from Right to Left of a line of text relative to the page
3	ToptoBottom	Distance from Top to Bottom of a line of text relative to the page
4	BottomtoTop	Distance from Bottom to Top of a line of text relative to the page
5	LtoRnext	Distance from Left to Right of a line of text relative to the next line
6	RtoLnext	Distance from Right to Left of a line of text relative to the next line
7	TtoDnext	Distance from Top to Bottom relative to current and the next line
8	DtoTnext	Distance from Bottom to Top relative to current and the next line
9	LtoRprev	Distance from Left to Right relative to current and the previous line
10	RtoLprev	Distance from Left to Right relative to current and the previous line
11	TtoDprev	Distance from Top to Bottom relative to current and the previous line
12	DtoTprev	Distance from Bottom to Top relative to current and the previous line
13	istart-parenthesis	Is line starting with parenthesis
14	istartnumber	Is line starting with number
15	istartbullets	Is line starting with bullets
16	istartwith-asterisk	Is line starting with an asterisk
17	normsize	Font size
18	linelength	Length of the current line
19	fontweight	Bold, Italic, or Normal
20	istartcap	Is line starting with the capital letter
21	istartspace	Is line starting with space
22	isenddot	Is line starting with a dot
23	ishorizontaltab	Is line starting with the tab space
24	caps	Number of capitals

Table 1: List of features

a set of unlabeled data points. The goal is to learn a feature representation that is amenable in forming K clusters in a manner that similar pairs of points tend to belong to the same cluster while the dissimilar pairs do not. Clustering algorithms have the advantage of being able to produce clusters directly from the data without needing any labels. But they are, by definition, uninformed of class labeling tasks, and are adversely affected by the curse of dimensionality.

Network Architecture

We build our model using an autoencoder architecture followed by a clustering technique. The parameter for the encoder and decoder is represented by θ_e and θ_d. The cluster centers are given by $\mu \in \mathrm{R}_d$, where d is the dimensionality of the latent space. We use z = f($\theta_e; x$) to denote latent space representations, where f($\theta_e;$) : X $\rightarrow$ Z is a nonlinear mapping from the input space to the latent space.Once the training is done the decoder part is no longer used, and the encoder is left for mapping its input to latent space Z. Lastly, the output of the encoder that maps the latent space representations is fed to the clustering algorithm to form K clusters in a manner that similar pairs of points tend to belong to the same cluster while the dissimilar pairs do no. Throughout our experiment, we have fixed all our hyperparameters such as the dimension size of each of the layers, Stochastic gradient descent (sgd) as a compiler to optimize the losses, and 50 epochs.

Figure 1 is the schematic representation of our model. Here, in our model we are considering both the K-Means clustering loss and Reconstruction loss during the training phase.

Loss Function

Our model being semi-supervised is jointly trained using a combination of the autoencoder reconstruction loss and the k-means clustering loss function. Reconstruction Loss: Generally the reconstruction loss is a distance measure $d_{AE}(x_i, f(x_i))$ between the input x_i to the autoencoder and the corresponding reconstruction f(x_i). Using the mean squared error of the two variables:

$$L = d_{AE}(x_i, f(x_i)) = \sum_i ||x_i - f(x_i)||^2 \quad (1)$$

where x_i is the input and f(x_i) is the autoencoder reconstruction. This loss function guarantees that the learned representation preserves important information from the initial input.

K-Means loss: In order to distribute data points evenly a neural network is trained with the following loss function:

$$L(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{N} s_{ik} \left|\left| z_i - \mu_k \right|\right|^2 \quad (2)$$

where z_i is an embedded data point, μ_k is a cluster center, and minimizing this loss with respect to the network parameters assures that the distance between each data point and

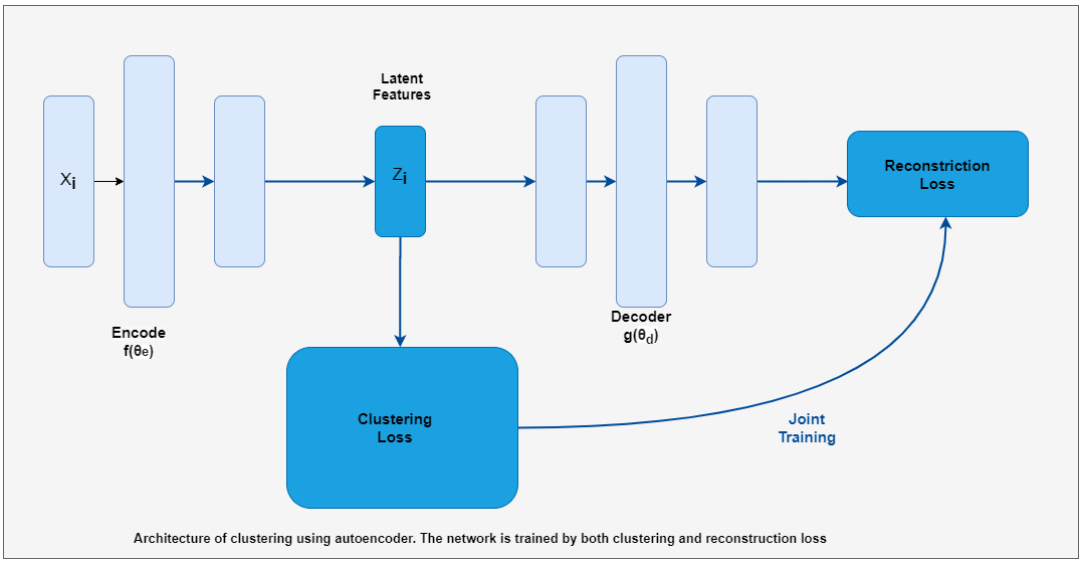

Figure 1: Schematic representation of model

its assigned cluster center is small. Having that, applying k-means would result in better clustering quality.

Training phase

Training our model is done in two phases. In the first phase, we pre-train our model using only available labeled data. Despite using relatively a small set of labeled data, this phase has a significant influence on the results. As the cluster assignments are based on the output of pre-training, which therefore needs to be as accurate as possible. During this pre-training phase, the loss of the DNN is measured using Mean Squared Error(MSE) between q(probability of a data point i to belong to a cluster K) and an empirical distribution obtained from labeled data. In the second phase of training, we train our model with unlabeled data without using true labels until the desired training accuracy is obtained.

In the second phase, we train our model only with labeled data. The rationale behind using this is because during a two-step process the model makes mistakes and corrects them. Indeed, during this phase of training, the model is trained with pseudo-labels, even though the true labels are available, therefore the model is prone to make mistakes. If the misclassification rate is too high, it is possible that some of the labeled data will be misclassified. Retraining the model with only labeled data can correct these classification mistakes. Hence, the combination of above mentioned phases yields more accurate results.

Table 2 shows our model's performance during the training procedure with different clustering techniques. Our model's average accuracy is 0.999 on K-Means and 0.993 on two clustering algorithms: Gaussian Mixture Model(GMM) and MiniBatchKMeans.

Method	Architecture	Clustering Algorithm	Train Accuracy
DCN	MLP	K-Means	0.999
DCN	MLP	GMM	0.993
DCN	MLP	MiniBatch-Kmeans	0.993

Table 2: Training result

Segmentation Unit

In our application, the end result is the segmented paragraphs in an orderly arranged manner. Here, we use the clustered data which is the output of the clustering algorithm to arrange the segmented paragraphs. Ideally, all the data-points have to be clustered amongst one of the three clusters such as first_line, regular_line, and other_line. So, based on the clusters mentioned above we arrange the paragraphs. The final output of this unit will be displayed in the User Interface(UI) where the end-user can download or copy the segmented paragraphs.

5 Evaluation

To evaluate our model, we randomly selected some PDFs which are labeled manually. Table 3 shows the model's output for different datasets. The average precision(AP) for test set 2 is 0.84 on our trained model with KMeans as the clustering algorithm. And, the AP obtained for test set 2 on our trained model with Gaussian Mixture Model(GMM) as our clustering algorithm. These AP values validate our model's capability to accurately extract the information from the diverse range of layouts and styles PDFs. The results reveal that our model performs exceptionally well for extracting the information from the PDFs and clustering each line of text

	Accuracy	Precision	Recall	F-Score
Using KMeans				
Set 1	0. 84	0.84	0.839	0.84
Set 2	0. 85	0.85	0.84	0.84
Set 3	0. 81	0.81	0.81	0.81
Using GMM				
Set 1	0.83	0.835	0.839	0.835
Set 2	0.84	0.84	0.84	0.83
Set 3	0.78	0.78	0.78	0.79

Table 3: Evaluation result

into one of the three categories first_line, regular_line, and other_line.

6 CONCLUSION

In this paper, we introduced an autoencoder based semi-supervised clustering approach that works in an end-to-end fashion. The clustering and representation learning driven by the few labeled samples as well as the unlabeled data and their predicted cluster-membership labels, as they evolve during the training process. The loss comprises a k-means style cluster loss and a MSE regularized by the autoencoder's reconstruction loss.Further, experimental results show that our model achieves better performance over state of the art semi-supervised approaches using very less labeled data and large unlabeled data.

Exceptionally, our model performs well with a limited amount of labeled data and a lot of unlabeled data , it is possible to achieve good clustering and classification results. Even though our model performs well on the most varied PDFs, due to the availability of wide market choices for rendering and producing PDFs with a variety of variations. There is also a chance of creating more reliable models in the future by using either of the following techniques: supervised , unsupervised or semi-supervised models.

References

[1] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255. PMLR, 2013.

[2] Dan S Bloomberg and Francine R Chen. Document image summarization without ocr. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 2, pages 229–232. IEEE, 1996.

[3] Hui Chao and Jian Fan. Layout and content extraction for pdf documents. In *International Workshop on Document Analysis Systems*, pages 213–224. Springer, 2004.

[4] Hervé Déjean and Jean-Luc Meunier. A system for converting pdf documents into structured xml format. In *International Workshop on Document Analysis Systems*, pages 129–140. Springer, 2006.

[5] Lloyd A. Fletcher and Rangachar Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE transactions on pattern analysis and machine intelligence*, 10(6):910–918, 1988.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759, 2017.

[8] Yen-Chang Hsu and Zsolt Kira. Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:1511.06321*, 2015.

[9] Koichi Kise, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382, 1998.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[12] R. Urtasun M. T. Law and R. S. Zemell. Deep spectral clustering learning.

[13] Song Mao, Azriel Rosenfeld, and Tapas Kanungo. Document structure analysis algorithms: a literature survey. In *Document Recognition and Retrieval X*, volume 5010, pages 197–207. International Society for Optics and Photonics, 2003.

[14] BreAnn Anshutz Maria F. De La Torre, Carlos A. Aguirre and William H.Hsu. *"MATESC: Metadata-Analytic Text Extractor and section Classifier for PDF Scientific Publications*. PhD thesis, 2020.

[15] George Nagy, Sharad Seth, and Mahesh Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.

[16] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[17] Lawrence O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 15(11):1162–1173, 1993.

[18] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017.

[19] Isabelle Lajoie Yoshua Bengio Pascal Vincent, Hugo Larochelle and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, page 11(Dec):3371–3408, 2010.

[20] Theo Pavlidis and Jiangying Zhou. Page segmentation and classification. *CVGIP: Graphical models and image processing*, 54(6):484–496, 1992.

[21] Muhammad Mahbubur Rahman and Tim Finin. Understanding the logical and semantic structure of large documents. *arXiv preprint arXiv:1709.00770*, 2017.

[22] Muhammad Mahbubur Rahman and Tim Finin. Unfolding the structure of a document using deep learning. *arXiv preprint arXiv:1910.03678*, 2019.

[23] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[24] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5315–5324, 2017.

[25] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.