Logical Layout Analysis using Deep Learning

Annus Zulfiqar‡, Adnan Ul-Hasan⋆, and Faisal Shafait†

‡†*National University of Sciences and Technology (NUST), Islamabad, Pakistan*
{‡*azulfiqar.bee15seecs,* †*faisal.shafait*}*@seecs.edu.pk*
[1]⋆†*Deep Learning Lab, National Center of Artificial Intelligence, Islamabad, Pakistan.*

***Abstract*—Logical layout analysis plays an important part in document understanding. It can become a challenging task due to varying formats and layouts. Researchers have proposed different ways to solve this problem, mostly using visual information in some way and a complex pipeline. In this paper, we present a simple technique for labelling the logical structures in document images. We use visual and textual features from the document images to label zones. We utilize Recurrent Neural Networks, specifically 2 layers of LSTM, which input the text from the zone that we want to classify as sequences of words and the normalized position of each word with respect to the page width and height. Comparisons are made by comparing the image under test with the known layouts and labels are assigned to zones accordingly. The labels are *abstract, title, author names*, and *affiliation*; however, the text also contains very important information for the task at hand. The presented approach achieved an overall accuracy of 96.21% on publicly available MARG dataset.**

I. Introduction

Deep learning has proven to outperform human accuracy in many computer vision and natural language processing tasks. Therefore, deep learning has become the default choice when use textual and visual features for labelling of logical units in document images. Logical labelling enables us to extract information from documents. Parts of documents are labelled such as Author, date, abstract, etc., so that retrieval efficiency can be increased. Logical layout analysis is the first step in any advanced searching systems where the data is contained inside a document such as a scientific papers and articles.

Layout specific approaches have been proposed in literature where the knowledge utilized to label zones in a document image comes from the geometrical features and the physical appearance of the layouts that have already been seen by the model during training. But this approach works well only if the test image has a similar layout as compared to the layouts in the training set.

Two important features can be used for layout analysis and labelling zones in document images, the position of the text block on the page and the text inside those blocks. Text can provide a lot of information about the nature of the zone, and the information about the location of this text on the page further helps in assigning a label to it.

It is possible to read a part of text and tell whether it was the title or name of the author. The visual information conveys some information as well, for example in this paper the abstract has been italicized. It is possible to just look at the paper and tell which blocks of text belong to the abstract and which do not. Also the title is normally a huge block of text on the top of the page.

Therefore, both visual and textual features can help in logical understanding of a document image. One can tell that the abstract will always come before the introduction in a paper, and that the title will be a few words only and smaller than the abstract. Similarly the affiliation will usually contain some addresses that we can be used to recognize these blocks in an image.

Our hypothesis in this work is that if a human can tell the difference between these zones based on their location and the text that they contain, then a machine learning model trained with sufficient data should also be able to do that. But visual features alone cannot provide enough information to label new layouts with good accuracy. Varying layouts makes this problem difficult to solve and efforts are being put in to come up with generalized systems for logical layout analysis on a broad spectrum of document classes. For the purpose of this paper, only the positions of the text blocks, positions of individual words in those blocks and their text has been used.

Natural language processing (NLP) is one of the paradigms in which machine learning has clearly outperformed heuristics and statistical approaches used in the past. We are now witnessing learned models to do things that were not possible with the traditional approaches. Most of the research in this field these days focuses on search, that does not require the user to type in long queries but simply ask the program for something and the natural language spoken by the person is automatically recognized as a query and the results are returned. The most popular commercial applications of this technology are found in Google's home, Amazon's Alexa, Window's Cortana and Apple's Siri assistant systems. Other common applications of NLP include language modelling, language translation, speech recognition and image captioning. These powerful applications clearly indicate the state of the art and what future this technology holds for us. For the purpose of our work the natural language that we want to process is not in the form of audio but plain text. Therefore we need models that are able to comprehend written text, make sense of it and recognize them as being titles of papers or names of people. We need vectors to represent text as arrays of numbers so that some learning network can process the text and output some predictions against it.

Recurrent neural networks (RNN), and especially its variant Long Short-Term Memory (LSTMs) [2] have been proven to be ideal for learning from sequential data. Text is one example of such data. Every single word in a sentence is linked to all other words in the sentence and every single sentence in

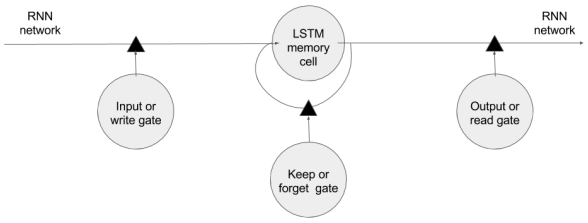

Figure 1: A simplified diagram of a single LSTM unit. The next state is a function of input at the current timestep as well as the previous state.

a paragraph or zone is connected to all other sentences in the zone. We need to keep track of these links in order to make some sense of what the text means. RNNs have an internal memory that keeps track of what it has already seen in a part of a sequence and that keeps changing the state of the RNN. It is possible to imagine RNNs as being state machines (see Figure 1) because the next state of an RNN is a function of both: the input at the current time step as well as the previous state of the RNN. But long term dependencies (or links) are better learned by LSTMs [2]. LSTM units are embedded into RNNs to learn these dependencies. LSTMs are placed between RNN layers such that the output of one RNN layer is the input to the LSTM and the value stored in the LSTM memory cell can be the input to the next RNN layer. For this reason we have chosen to work with LSTMs.

This paper is further arranged as follows. Section II overviews the related research work that has been carried out in past. Section III describes our methodology in detail. Section IV outlines the experimental details of our method and Section V concludes the paper with discussion and future work.

II. Related Work

Marc [4] did a very similar work where Named Entity Recognition was applied on the Grotoap2 dataset. Their model reads the document from left to right feeding in the words with their locations into two layers of bidirectional LSTMs. They achieved accuracy up to 94.47% on the Grotoap2 dataset.

Rangoni et al. [6] devised a dynamic perceptive neural network and used the geometric, morphological, and semantic features in the images to label zones. They analyzed the output of their model after each test to decide whether the output was correct or the input needed to be modified and fed back into the network. The network overall used feedback with three feedback iterations and a complex pipeline that also utilized k-means on the way for finding which of the known layouts the image under test best resembled to make correct predictions about the text zones. They have demonstrated an overall accuracy of 97.5% on MARG dataset.

Aiello et al. in [1] presented a technique on labelling the logical zones in document images and also predicting the reading order. They utilized global and spatial locations of document objects (title, caption, etc.) in the form of thirteen relations: *precedes, meets, overlaps, starts, during, finishes, equals*, and *their inverses* to describe spatial relationships among the entities in the document image in the form of thick boundary rectangle relations (TBRR) and predicted the reading order in the document by assuming that all text blocks are connected with each other as vertices on a graph and then used a spatial reasoning module along with NLP (natural language processing) to find the weights on the edges. They used Desicion Tree as a classifier with features including aspect-ratio, font style and number of lines. Performance measure on the UW-II database yielded up to 98% precision.

Todoran et al. [9] presented a model that predicts the reading order in document images from vast number of classes on the basis of the spatial features in the image and a spatial reasoning module that makes the decision on the basis of the location of each individual text block.

van Beusekom et al. [10] demonstrated an example based approach in which a set of labeled document layouts and a single unlabeled document layout is taken as input and their solution finds the best matching layout in the set. The labels of this layout are used to label the new layout. The similarity measure for layouts combines structural layout similarity and textural similarity on the block-level.

Shafait et al. [7] presented an approach that models known page layouts as a structural mixture model. Then a probabilistic matching algorithm is presented that gives multiple interpretations of input layout with associated probabilities. [7] has reported 99.6% accuracy on the problem of geometrical layout analysis on a portion of MARG and some other dataset, a total of 1300 images from 6 journals.

III. Methodology

This section describes our approach using LSTM networks. The basic architecture of a single LSTM unit is as follows: One LSTM unit has a memory cell, to remember some analog value, which is controlled via three gates, or controlling variables that are set and reset by the rest of the RNN surrounding this LSTM. These three gates are called the *read* gate, the *write* gate and the *keep* or *forget* gate. The names given to these gates are self-descriptive. Firstly, we set the keep gate to 0 to tell it that it should wipe the information stored in the LSTM memory cell. It does this by a multiplicative action. The value of the keep gate is always between 0 and 1, a zero meaning forget everything by multiplying the stored value by a 0. A 1 means retain whatever analog value is in the cell by multiplying it by a 1, and some other value of the keep gate, say 0.3 means remember only 30% of whatever is in the memory cell. The write gate is then activated by the rest of the network to let information flow into the LSTM. So if the write gate is 1 it means that we are now writing a value into the LSTM memory cell. Once written, we deactivate the write gate and set the value of keep or forget gate to 1. So now the LSTM unit will remember this value as long as the keep gate is set to 1. Whenever this value is needed at some point in the future, we simply set the read gate to 1 and the RNN that follows this LSTM block can read the value from the LSTM's memory cell. A simplified diagram of a single LSTM unit is shown in

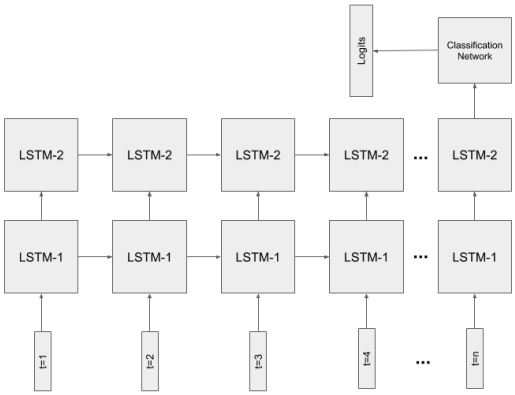

Figure 2: Our Model: LSTM 1 and 2 represent the first and second LSTM layers. Each layer has 256 units. ReLU has been used as activation function in current work.

Figure 1. Further details on the implementation of these gate mechanisms and how these LSTMs are embedded into RNNs can be found in [2]. By using these RNNs and LSTMs, a light weight model is presented in next section that will need the words in a zone along with their coordinates converted into long vectors and the network will directly predict its label that whether it is the name of the author or his affiliation.

A. Our Model

For the purpose of this work, we assume that zone corners are available to us, so we will use the zone coordinates from the ground truth, but the text and positions of each word will be taken from OCR output, that will be performed on the document images using these zone corners. We have used tesseract OCR engine to extract textual blocks from the document image. The vectors obtained using GloVE [5] (for each individual word from the tesseract output for each zone) were appended with their normalized positions on the document page and fed into 2 layers of LSTM network, each layer having 256 LSTM units, followed by 2 matrix multiplications with ReLU used as the activation function. So the overall network architecture is extremely simple: an input layer, followed by 2 layers of LSTMs having 256 LSTM units each, and finally two matrix multiplications at the end to predict the zones. A simplifed block diagram of the network is shown in Figure 2. The number of time steps, that is the number of words in the text zone in our case is 128, so the sequence length cannot be longer than 128. If the sequence length is longer than 128, we truncate the access words and if the sequence already contains words less than 128, we append 0s.

B. Feature Vectors

Since the model needs to learn numbers as weight and bias matrices in the network and predict the class of the text as numbers as well, therefore we need to convert the input text into numbers too. Words in the text are converted into large vectors of numbers for providing as input to network. For that reason we have used GloVE [5] (an unsupervised learning algorithm for obtaining vector representations for words) vectors open sourced by Stanford University for getting word embedding for every word in the sequence of words in each of the text zones. GloVE provides 300 dimensional vectors, one vector for each word. We have used the one trained on common crawl having 840 billion tokens and vectors for a total of 2.2 million words. Since we also want to append word locations to these vectors to get the spatial position of the zone embedded into each word vector, so we first normalize their coordinates according to page width and height and then append them at the beginning of the vector. This makes it a 304 dimensional vector.

C. Training

Default settings were used for training this network, softmax cross entropy with adam optimizer [3] for stochastic gradient descent. A batch size of 64 sufficed for training on MARG data. Tests were performed on the OCR output using only the zone corners from the ground truth data. The learning rate was set to 0.0001 and the sequence lengths in each of the zones text was restricted to 128. Training takes only 100 iterations on MARG to give above 90% cross validation accuracy in less than 3 minutes of training with a batch size of 64. This should not be surprising since MARGs zones are easy to distinguish from each other if textual features are used. We believe that the way we initialized the model is mostly responsible for this speed of convergence. A random normal initializer with range between positive and negative 0.1 was chosen for the two LSTM layers, and for the two matrix multipliers we used random normal initializer with mean and deviation 0.5 and 1 respectively. We also used dropout with a keep probability of 0.7 between the last two matrix multiplications for one of the 2 experiments.

IV. Experiments and Results

A. Data set

MARG (Medical Article Records Groundtruth) [8] is a freely available repository containing front pages of medical articles of renowned journals and their associated ground truth. It contains a large variety of journal layouts with several examples of title pages from each journal. It was developed as a part of the efforts made in digitizing the US National Library of Medicine. It contains 4 major zones to label namely *title, authors, affiliation*, and *abstract*. The journal layouts are categorized into nine classes based on the geometric arrangement of logical page blocks (title, author, affiliation, abstract). Eight of the layouts are unique in their geometrical features while the last one contains all of the layouts that are smaller in number.

We will only compare our results with those papers that have demonstrated their results on MARG dataset.

B. Experimental Evaluation

First we used 50% of total document images in MARG for training, 10% for evaluation and 40% for testing. With this split we get an overall accuracy of 96.21% on MARG. [6] reported an error rate of less than 2.5% on MARG. It seems that their model performs better in this case but it has

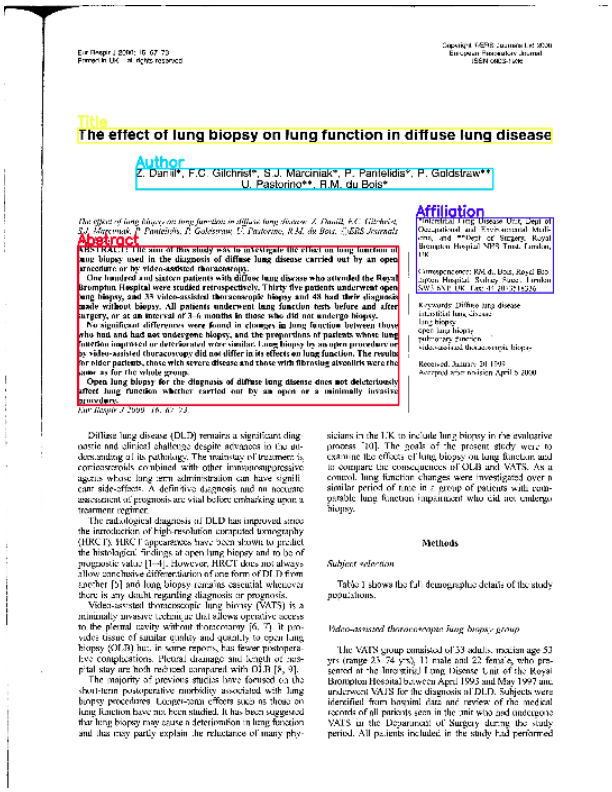

Figure 3: A labelled image from MARG dataset

to be noted that their model uses a feedback mechanism with up to 3 perceptive cycles and alter the input to better match the representation of a known layout, against which we have a lightweight feed-forward network that gives less than 4% error rate in a single pass with the same proportion of the data used for training the model. Another thing to be noted is that our approach is layout agnostic because we are not trying to match the test image with any of the known layouts although the positions of the text zones are used because the positions of title and names of the author are usually placed in the same region on the title page of scientific papers. Table I reports our accuracies on this test for each of the four zones.

Zone	Accuracy%
Title	94.95%
Author	97.14%
Abstract	96.64%
Affiliation	96.13%

Table I: Accuracy on MARG Zones

Next we investigated how this model labels zones from an unseen layout, that is to say how does this model perform when it has to predict the labels for a zone that is coming form a layout that it has never seen before during training. For this purpose, we used leave-one-layout approach like [10] and trained the model on all layouts except one, and that one was only used for testing. No dropout was used in this case. Table II summarizes our results.

Test Layout	Accuracy%
type a	95.9
type b	95.2
type c	95.8
type d	96.3
type e	94.9
type f	97.4
type g	95.6
type h	95.0
type other	92.3
Overall Acc.	95.4

Table II: Leave one layout test.

van Beusekom [10] reported 94.8% accuracy on leave-one-layout test. We are assuming that their results are averaged over all types, so it can be seen that with our approach we get 95.38% accuracy upon testing our trained model on an unseen layout. [10] was using a model that was transferring labels from one of the known layouts to an unseen layout on test time. The text in the zones was not used for doing so, only the graphical representation of the image was used. Comparison with the known layouts determined that from which of the known layouts the labels had to be transferred. Obviously the test layout was not in their model's knowledge at test time but still they compared and assigned labels and still got a very high accuracy number. On the other hand our model is not transferring the labels from some known layout directly to the test image but instead utilizing the knowledge gained from the text of the training images and the relative positions of the zones. Hence we conclude that our model generalizes better to unseen layouts as well. It is to be noted that here that we have not taken into account any errors that occur due to the OCR and therefore its consequences have not been investigated for this work.

Table III and Table IV summarize comparisons on the two tests that we performed. Since we have only performed test on recognizing zones in MARG, so we compare with those works that have used MARG for the same purpose.

Method	Accuracy%
Rangoni et al. [6]	97.5%
Our model	96.2%

Table III: Comparison of Accuracy with random split on MARG

It can be seen that our accuracy falls short of [6] when we train and test on MARG with random splits. Also [6] has averaged results on 10 different random splits to minimize variability. Although our method performs better than [10] on unseen layouts.

Method	Accuracy%
Beusekom et al. [10]	94.8%
Our model	95.4%

Table IV: Overall Accuracy on "leave one layout" on MARG

V. Conclusions and Future Directions

We have presented an approach that can perform logical layout analysis on front pages of scientific papers but it can easily be scaled to do the job on document images from a wide range of classes. This method performs well even on unseen layouts because we take text and normalized positions of all words in that text as input to the network for predicting the labels of their zones. Although it performs slightly poor on random splits as compared to some other known techniques but it is much lighter in weight as compared to most of those models. Accuracy rates of up to 96.21% were achieved with random split and up to 95.38% on unseen layouts. For this work, we assumed that the zone corners were already available to us, so we used the ground truth zone corners. For our future work, we intend to completely automate this process into a single system that predicts the corners of the text blocks, runs this model on those zones and predicts the labels for each of them. Moreover we would like to demonstrate our results on bigger datasets with more types of zones to label.

References

[1] M. Aiello et al. "Document understanding for a broad class of documents". In: *International Journal on Document Analysis and Recognition* 5.1 (2002), pp. 1–16.

[2] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[3] D. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[4] R. Marc. *Understanding Structured Documents with a Strong Layout*. 2017.

[5] J. Pennington, R. Socher, and C. Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[6] Y. Rangoni, A. Belaïd, and S. Vajda. "Labelling logical structures of document images using a dynamic perceptive neural network". In: *International Journal on Document Analysis and Recognition* 15.1 (2012), pp. 45–55.

[7] F. Shafait et al. "Structural mixtures for statistical layout analysis". In: *Document Analysis Systems, 2008. DAS'08. The Eighth IAPR International Workshop on*. IEEE. 2008, pp. 415–422.

[8] G. Thoma. "Ground truth data for document image analysis". In: *Symposium on document image understanding and technology (SDIUT)*. 2003, pp. 199–205.

[9] L. Todoran et al. "Logical structure detection for heterogeneous document classes". In: *Document Recognition and Retrieval VIII*. Vol. 4307. International Society for Optics and Photonics. 2000, pp. 99–111.

[10] J. Van Beusekom et al. "Example-based logical labeling of document title page images". In: *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*. Vol. 2. IEEE. 2007, pp. 919–923.