Information Extraction from Visually Rich Documents

Marcus Skinner
Indiana University–Bloomington
marcskin@iu.edu

Abstract

This paper presents a solution to extracting information from receipts using graph convolutional neural networks. It outlines a method for effectively capturing the structure of a receipt, including presenting methods for feature engineering. The experiments include results for Receiptnet, a deep learning architecture that employs several graph convolutions, including graph, chebychev, Sage, Topology Adaptive, and Graph Attention (GAT) using the Deep Graph Library (DGL). The model achieved a precision of 41% on predicting the price total of a transaction.

1 Introduction

Information extraction (IE) is the process of extracting information from documents. Traditionally, IE has focused on extracting information from plain text documents. Recently, there's been a need to extract information from unstructured documents, such as invoices, receipts, etc. Unlike plain text documents, unstructured documents do not follow a certain template, so it's not known exactly where to find the relevant information. In addition, unstructured documents lack language cues for certain information, so methods can't employ traditional language processing, like regular expressions or leveraging key words. It might be that a receipt contains many different dates or price totals.

A receipt might have a 'subtotal', a 'total including GST', a 'total after discounts', and a 'grand total'. Different receipts might also use different words to designate their final totals, including 'grand total', 'final total', 'total', 'subtotal', etc. To explicitly extract information from a receipt, through language processing or some other method, it would take an almost infinite set of rules. Even if an application had such an ubiquitous set of rules, it would need to be updated over time, as the language used in the document might change. Not to mention taking into account different languages and dialects. Explicitly denoting rules for information extraction on receipts is neither efficient, accurate, nor scalable.

Receipts are a type of visually rich document (VRD), and most of the information is not found in the actual text but in the visual representation of the information. Oftentimes, the relation of a section to another sections gives a hint to the underlying information. This observation has led to work in template matching (TM). TM captures the structure of a specific receipt and generates a template. The template is used to extract information from other, similar documents. Unfortunately, this also does not scale well since there isn't a conventional structure to receipts. This would work well for a single type of receipt, like Walmart receipts, but it doesn't work with receipts from different vendors. TM requires a unique template for every receipt variation. This type of solution is not viable.

An application needs to generalize the structure of a receipt to effectively extract information. While there is not a set structure to receipts, companies still design them in such a way that is easy for a customer to understand. So, must be some relation between the text and its location. For example, the company name on a receipt is usually located at the top, above everything else. The total price of a transaction is usually located toward the bottom. So, a model needs to learn a receipt's underlying structure. A solution to the receipt problem is using graph convolutional neural networks.

The network will take into account a node's features and its connections to other nodes. Once the structure of a graph is effectively captured, GNNs can generalize it.

2 Graph Convolution Neural Networks

A GNN works similarly to a convolutional neural network (CNN), in regards to the convolutions working as a way to extract features. CNNs are so effective because of their Shift In-variance. That is, features can be extracted from data regardless of the position of the features in the data. For example, a CNN might learn to extract the ears in images of cats, regardless of the position of the ears in the image. This makes the network more robust because an ear could be located at any position within the image, and the network would still recognize it as an ear. Even so, CNNs still require information about neighboring pixels, so the feature extraction is rooted in euclidean space.

However, the structure of a graph isn't based on euclidean space like an image is. Indeed, the connections are based on the nearest neighbors, but the space between text boxes is irrelevant. Only a text box's relation to other text boxes is important to capturing the structure of a receipt.

If a CNN was applied to the pixel data of a receipt, no meaningful features will be found. This is because the pixels themselves don't contain any useful information. The relation of individual items on a receipt are what contains the information. But not in what the receipt looks like, but in its underlying structure. Because the underlying structure of a receipt is important, it has the be represented in a meaningful way. A graph captures the structure of the receipt.

3 Graph Convolutions

A graph convolution summarizes the features in a neighborhood of nodes. For each node in a graph, the convolution applies a function to the surrounding neighbors to extract features. For example, let's say a house can be classified as residing in a poor or rich neighborhood. The class of the house cannot be found by simply looking at the house itself. The neighbors of the house need to be considered. So, a good function might be the mean of the net incomes of the surrounding household within K positions. If the median is above a certain threshold, the house might belong to a rich neighborhood.

Essentially this is how a graph convolution works. For each node, it summarizes the neighborhood of nodes within K edges. ReceiptNet uses three separate convolutions–Graph, Chebychev, and GAT.

For all convolutions b represents the bias vector and W represents the network weight vector for layer l. Each described as implemented in the DGL library. ReceiptNet also implements the Sage and Topological Adaptive convolutions, but for brevity, they aren't given here. Further information on those convolutions can be found in the DGL documentation.

3.1 Graph Convolution

The simplest convolution is the Graph convolution given by the function

$$h_i^{l+1} = \sigma(b^{(l)} + \sum_{j \in N(i)} \frac{e_{ji}}{c_{ji}} h_j^{(l)} W^{(l)})$$

where $N(i)$ is the set of neighbors of node i, $c_{ji} = \sqrt{|N(j)|}\sqrt{|N(i)|}$, σ is the activation function, and e_{ji} are the weights of the edge from node j to node i. Note that e is 1 if no edge weights are given.

3.2 Chebychev Convolution

$$h_i^{l+1} = \sum_{k=0}^{K-1} W^{k,l} z_i^{k,l}$$

where $Z^{0,l} = H^l$, $Z^{1,l} = \hat{L} \cdot H^l$, $Z^{k,l} = 2 \cdot \hat{L} \cdot Z^{k-1,l} - Z^{k-2,l}$, $\hat{L} = 2(I - D^{-1/2} A D^{-1/2}/\lambda_{max} - I$

3.3 GAT Convolution

$$h_i^{l+1} = \sum_{j \in N(i)} \alpha_{i,j} W^{(l)} h_j^{(l)}$$

where α_{ij} is the attention score between node i and j.

The attention score is given by,

$$\alpha_{ij}^l = softmax_i(e_{ij}^l)$$

and,

$$e_{ij}^l = LeakyReLU(a^T [W h_i || W h_j])$$

4 CORD Dataset

The Consolidated Receipt Dataset (CORD) (Park et al.) that contains receipts with Optical Character Recognition (OCR) already applied. Since OCR is already a well established line of research, this paper doesn't go into detail on it.

Figure 1: Example Receipt from the CORD dataset

The dataset contains 1,000 examples from Indonesian receipts. Each extracted text box falls into 1 of 5 classes–the menu, the void menu, subtotal, void total, and total. ReceiptNet will try to classify the node that belongs to the total class, or the price of a transaction after all deductions. The dataset is split with 800 training examples and 200 testing examples.

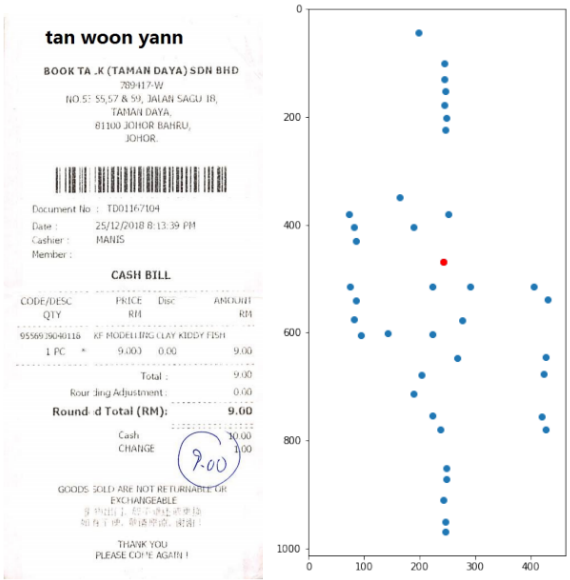

Figure 2: Nodes are created using the center of each text box found by OCR

5 Graph Modeling

The data processing is done with the graph modeler. The goal of the graph modeler is to capture the structure and information of a receipt into a graph. Specifically, the graph modeler converts a receipt to a DGL (Deep Graph Library) graph. DGL is relatively new with the first version of DGL came out in 2018.

DGL can use Pytorch, Tensorflow, and other libraries as backends. PyTorch seems to be the better supported of the libraries. Specifically, DGL extends the PyTorch module for neural networks. A little work has to be done to covert the CORD dataset into a DGL graph and that is handled in the training file. A DGL graph object is created by specifying the edges in the graph.

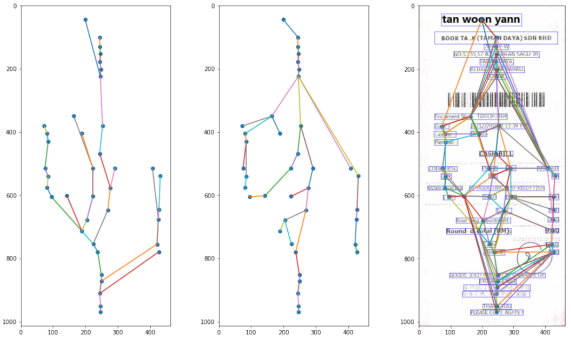

Figure 3: Receipt from the ICDAR 2019 Robust Reading Challenge. Each node is connected it its nearest neighbor in four directions

The graph modeler finds the edges by connecting each node to its nearest neighbor in each of the four main direction (top, bottom, left and right). First, it computes the midpoints for each textbox, then looks for the nearest neighbor in each direction. Finally, the modeler produces a DGL graph with each node connected with 4 edges.

Finally, the labels for each node are added to the graph. In this case, 0 is used for "None" and 1 is used for "price total."

6 Feature Engineering

A couple node features are added to the graph to help the model. Features that improve precision include identifying if a node is a price, number or date. A simple regular expression is used to classify a text box as one or more of these types and added to the graph. This distance between each node to its four nearest neghbors is also added.

7 Model Architecture and Training

The model ReceiptNet is built using Pytorch (Paszke et al., 2019) and DGL (Wang et al., 2019). ReceiptNet is a graph convolutional neural network that takes in a graph and tries to classify the nodes. That is, ReceiptNet does node classification. Specifically, ReceiptNet will take in a graph and classify each node as the 'total price' or 'something else.' Right now, ReceiptNet finds the node that is most likely to be the price total and returns that node. It does this by producing a softmax layer, with each output corresponding to a different node. It then takes the argmax of the softmax layer to find the node with the highest probability of being the price total. The model assumes that only one node with represent the price total.

ReceiptNet was tested on some built-in datasets in DGL. In particular, the model achieved an 80% accuracy on node classification using the CORA dataset.

Graph Convolution	Precision
`Graph`	41%
`Chebychev`	24%
`Sage`	37%
`TAG`	31%
`GAT`	26%

Table 1: Precision of ReceiptNet on CORD dataset per Graph Convolution

Since the data is unbalanced (that is, most of the nodes are not the price total), accuracy is not a good metric. The accuracy is always close to 98%, which sounds really good but since most of the nodes will be classified as not the price total, it's not as good as it sounds. If the model labeled all the nodes 0, it would still get an accuracy in the upper 90s. Instead, a confusion matrix is computed when testing each model and looked at the number of true positives divided by the number of graphs. That is, measure model performance is measured using precision. Using this metric, with about 800 training samples, ReceiptNet achieved a precision of 41% using the Graph convolution. Please note that this number to go up if given more training samples. But for so little time, having more than a couple hundred receipts is not viable.

8 Future Improvments

Adding additional classes to ReceiptNet is the next major step. As seen, CORD offers 5 classes with several subclasses within that. However, with a little testing not discussed in this paper, different convolutions perform better or worse for different labels. For example, Graph is not the best convolution for predicting the node that represents the date. In that case, a sort of ensemble learner would be ideal. Essentially, a separate prediction would occur, with different sub networks, for each class. Currenly, ReceiptNet's architecture supports adding sub networks with different convolutions.

Furthermore, the line items on a receipt might be of interest. In this case, image segmentation might be a good route to pursue as a prepossessing step in an attempt to group the line items together.

Other features could also increase performance. Common key words could be taken into account, such as the word 'total' or 'price.' A little of this is already implemented but it could be expanded. In addition, TF-IDF or another way to represent text as a node feature might offer useful information.

9 Conclusion

ReceiptNet uses GNNs to classify nodes from a graph created from a receipt with a graph modeler. Several different convolutions were explored and will give varying results depending on the type of classification. In the case of predicting price total, the Graph convolution performed the best, resulting in a precision of 41%.

10 Further Readings

The following text cites various types of articles so that the references section of the present document will include them.

- Book on GNNs (Hamilton).
- Deep Learning Book (Goodfellow et al., 2016)

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. Cord: A consolidated receipt dataset for post-ocr parsing.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks.