# Text Classification using Graph Convolutional Network

*for*

Information Retrieval

*by*

Hrishikesh Harsh, 2020A7PS0313P
Harsh Priyadarshi, 2020A7PS0110P

## Abstract

The field of Information Retrieval has several open Problems which have received attention from Computer Scientists all over the world. One of the problems is the problem of Text classification. The factors like representation of the vocabulary; how to handle huge amounts of words in the vocabulary; how to best represent them in Mathematical format; how to map the correlation between them; and several other considerations have led to years of research. One such research outcome was the Text GCN paper based on the concept and model of Graph Convolutional Networks. In our work, we seek to introduce and experiment with some methods and tweaks in order to observe and improve the results over the conventional way of Classification on a dataset with multiple categories of text data.

## Background

Text classification is a fundamental problem in natural language processing. To process the text in this field, it requires to be represented in an acceptable format. Traditional methods involve posting lists, bags of words, and n-gram models. However, these text representations capture information in local consecutive word sequences but ignore global relations between words and documents. In this paper, we will explore an approach by Liang Yao, Chengsheng Mao, and Yuan Luo, in the paper “Graph Convolutional Networks for Text Classification”. Based on a list of documents, a single large graph is constructed. All the words from the documents are collected together to form a large vocabulary, in which common or infrequent words are removed. The remaining words, along with all the documents, are turned into nodes to make a heterogeneous graph. The edge between two nodes is made in which the weight between a node i and a node j is defined as: -

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & i,j \text{ are words, PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

| | |
|---|---|
| $$\text{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)}$$ $$p(i,j) = \frac{\#W(i,j)}{\#W}$$ $$p(i) = \frac{\#W(i)}{\#W}$$ | Here, *#W(i)* means the number of sliding windows in which the word *i* appears. *#W(i,j)* means the number of windows in which the word *i* and word *j* both appear.<br>A positive PMI value implies a high correlation between the words, whereas a negative correlation means no correlation, because of which such cases are ignored and no edge is made. |

The relationship between a document node and a word node is categorized by the TF-IDF score, where the TF (term frequency) refers to the number of times a word appears in a particular document, and IDF (inverse document frequency) represents the logarithmic inverse fraction of the number of documents the word appears in. However, all the words in the documents are treated equally, though that might not be the case for all documents, which is our point of research.

# Methodology

As mentioned above, the limitation of this paper is that it treats all words in the documents with equal value which may not be the case in documents that are segregated in different formats.

**DATASET**

The dataset we have is a PubMed dataset which involves research papers in the field of Covid, But each research paper is separated into three categories, which are abstract, keywords, and title. All the papers that are strictly related to research on SARS-CoV-2 and Covid-19, are labeled 1 whereas papers that have non-contextually mentioned these terms are labeled 0.

The goal of this project is to turn all these documents into graphs and then train a model using the labeled datasets. But since we have three different types of documents, our approach involves changing the weightage of each of the words in their TF score, based on the number of times they appear in keywords, titles, and abstracts.
This way of classification and weightage can be very important for Books, where authors or titles have higher weightage and hotels where location has a higher weightage.

# Implementation

The original formula for the calculation of the TF score is:-

$$TF = \begin{cases} 1 + log_{10}(termfreq) & , termfreq > 0 \\ 0 & , termfreq = 0 \end{cases}$$

In the original paper, termfreq = number of times a term appears in the entire document.
This basically means that, for the PubMed dataset, the approach would be to simply concatenate all categories of words and assign *W = 1* to all if seen from the lens of our approach.

For our approach, we have taken,

Treat Abstract, Key and Title differently

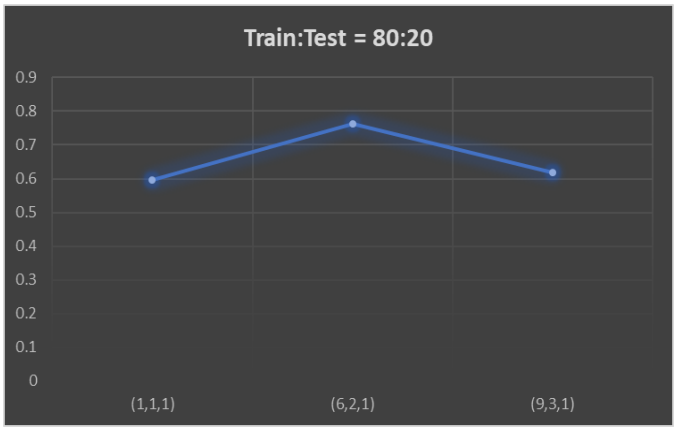
Order: $Key > Abstract \approx Title$

Hence, $W_k > W_a \approx W_t$

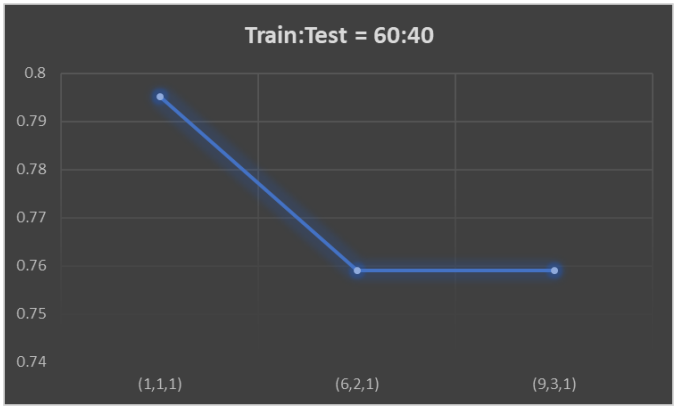
$$TF = 1 + log(n_t * W_t + n_k * W_k + n_a * W_a)$$

Where $W_k$, $W_a$, and $W_t$ represent the weightage of keyword, abstract, and title in the tf score, and $n_k$, $n_a$, and $n_t$ represent the frequency of a word in each of these categories. This was then run on different splitting ratios between train and test, and the resulting accuracy, F1-Score, and weighted score were calculated.

# Results and Analysis









| Split | Accuracy with weights {9,3,1} | Accuracy with weights {6,2,1} | Accuracy with weights {1,1,1} |
|---|---|---|---|
| 80:20 | 0.61 | 0.76 | 0.59 |
| 70:30 | 0.82 | 0.79 | 0.70 |
| 60:40 | 0.75 | 0.75 | 0.79 |
| 50:50 | 0.809 | 0.75 | 0.805 |
| 45:55 | 0.81 | 0.78 | 0.80 |

| | Improved Accuracy using our approach | | Accuracy not improved |
|---|---|---|---|

Based on our results above, we can infer that changing the weightage of keywords, and title words, improves the accuracy of the model. The general trend has been that decreasing the Train:Test split causes a general increase in the accuracy of the model trained on (9,3,1) weights. The explanation can be found in the fact that relevant and important keywords are scarce now and therefore, only a higher weight assigned to them can give them a proper representation in the Training.
However, if the number of labeled documents decreases, then the number of keywords that may not be relevant to Covid-19, but are present in documents labeled true, will have a higher impact on the tf-idf score as well as the prediction of the new data. So it becomes a competition between the relevance of keywords and the number of documents associated with true or false.

# Conclusion

Changing the weights of words in different categories has mostly improved the accuracy of the model. This model can be useful in various environments where classification requires treating different information differently. These cases include tasks like categorizing product reviews, ratings, and stars, categorizing restaurants in an area based on cuisines and descriptions, etc.
There is a scope for further improvement through extensive experimentation with different datasets and tweaking weights assigned to different categories of text data to further solidify our hypothesis.

# References


1. Yao, L., Mao, C., & Luo, Y. (2018). Graph Convolutional Networks for Text Classification. *ArXiv*. /abs/1809.05679
2. Han, S. C., Yuan, Z., Wang, K., Long, S., & Poon, J. (2022). Understanding Graph Convolutional Networks for Text Classification. *ArXiv*. /abs/2203.16060
3. Kipf, T. N., & Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv*. /abs/1609.02907