TweetGCN: A Graph Convolutional Network Approach to Tweet Classification

Benjamin Bradley[1] Cecily Chung[1] Charles Duong[1]

[1]Brown University

Introduction

In this project, we propose a Graph Convolutional Network (GCN) [4] approach to topic classification on Twitter. We base our approach off of the **TextGCN** paper [6], which implements a GCN in TensorFlow and casts it to the field of NLP. It addresses text classification and experiments on several datasets to classify long documents by topic. For our project, we re-implement TextGCN in PyTorch and cast this to Twitter topic classification.

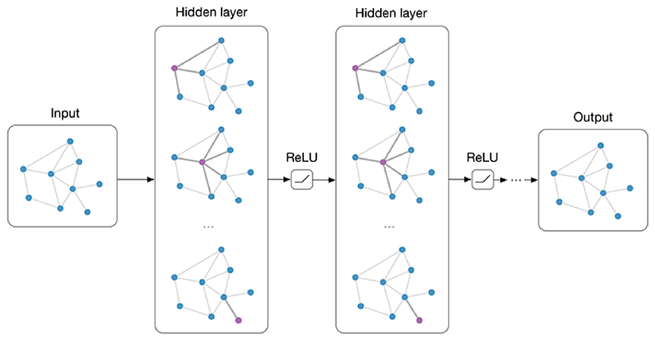

Figure 1. Visualization of the structure of a Graph Convolutional Network (GCN)

This task is highly relevant as Twitter and other text-based social media platforms make up a large part of communication worldwide, making it essential to design models that can accurately interpret Tweets and other short and informal forms of information-sharing. The original TextGCN paper classifies long documents (medical research papers), but a model that can quickly understand brief, casual text will be adaptable and applicable to modern forms of information-sharing.

Problem Statement

We formally define the Tweet classification as follows:

Let $X \in \mathbb{R}^{N \times F}$ represent our dataset, that is, X is the set of textual posts and let N be the number of posts in our input and let F be the number of features in our post.

We now construct a fully connected graph of online posts, $G = (V, E)$, where each post x_i is a node. Let each edge e be the cosine similarity between embeddings (the distance between the vectors).

To formulate our loss function, let $Y \in \{l_k | k = 1, 2, ..., K\}^N$, where l_k is the k^{th} tweet label (music, healthcare, tech, etc) in our labeled dataset and K is the number of classes.

We now formally define our problem as follows: given X and Y, we want to learn a function F paramaterized with W that maps X to Y: $F(X) \rightarrow Y$. We thus minimize the loss function as follows:

$$\arg\min_{E,W} \mathcal{L}(Y, F(X, W, E))$$

Methodology

We used the following process to structure the training and testing of our GCN:

1. Preprocess tweet data
2. Embed tweets using Sentence BERT and set labels to one-hot encodings
3. Graph construction using cosine similarity values
4. Graph Convolution Layers
5. Training & Testing Tweet classification

Our architecture followed the TextGCN paper utilizing two graph convolutional layers, ReLU activation, dropout, and a hidden size of 512.

Experimental Setup

Data and preprocessing:

We utilized the CardiffNLP TweetTopic dataset [1] assembled by the Cardiff University NLP team for their 2022 NLP Hackathon to train and test our model. The dataset consists of 4.37k Tweets assigned to one of six labels (Sports, Business, Pop Culture, etc.).

We implemented basic preprocessing which included omitting links, mentions (ex: @username), the retweet flag "RT", and punctuation (not including apostrophes to avoid splitting contractions). We opted not to remove hashtags or stop words (ex: "not"), as they were often useful in discerning the Tweet's meaning.

Baselines and Metrics

To test our model, we compared against five other traditional machine learning classifiers, implemented with sklearn's built-in models: Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), XGBoost (XGB), and Decision Tree (DT).

We obtained test accuracy, false negative rates (FNR), false positive rates (FPR), and F1 scores. To compare the significance between two models, we also ran McNemar tests to calculate the divergence of distributions. For these tests, our null hypothesis is that the GCN model will have no statistically significant difference between the evaluation metrics. Our alternative hypothesis is that the model will have a statistically significant difference between the evaluation metrics.

Discussion

While we did not run into any specific major roadblocks in this project, implementing a GCN for text-classification was difficult given the minimal coverage of GCNs in class. Going forward, we expect to expand our model in the hopes of improving performance beyond 78% accuracy after 10,000 epochs. We would spend more time fine tuning, playing around with architecture, and crucially finding larger, more balanced datasets. We hope our model can be built upon to classify tones such as hate speech, advertisement, etc.

We found the following technical challenges in our results:

- Performance was likely reduced by a dataset class imbalance, as the three most common labels made up roughly 90% of the dataset. Future works may explore data augmentation techniques to balance training data.
- Tweets often contain informal language leading to noisy network data where few words are relevant to classification. Future works may investigate dealing with such language.
- Constructing a fully connected graph for convolution is a computationally expensive task and may lead to poor performance when applied towards sparse tasks (ex: classifying Tweets as spam). Future works may investigate making graph construction more robust.

Results

Our findings were mixed as we found competitive, though not superior, accuracy compared to the five sklearn models against which we compared. While we observed real, stable learning curves from our model, we also observed difficulty in separating some classes from another, as reflected in Figure 2.

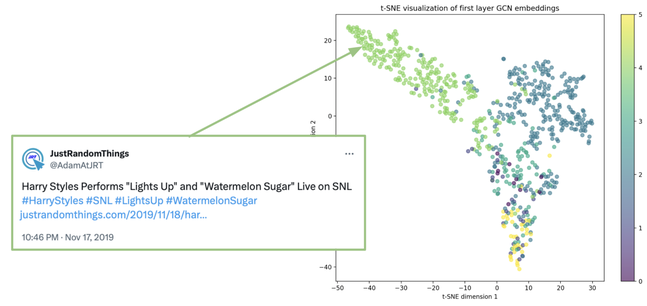

Figure 2. Graph of the 2D decomposed embeddings of the test set after the first GCN layer

As shown in Table 1, the GCN model performed on par with the sklearn classifiers, even outperforming the XGBoost model with respect to accuracy and F1-Scores. The FPR and FNR are also comparable to the traditional classifiers. A key component in benchmarking our model against others was using McNear tests to measure whether the label distribution our GCN learned reflected the distribution learned by others. As shown in Table 2, our GCN learned a statistically significantly different distribution from the other baseline methods, with the distribution being most similar to XGBoost and most dissimilar to Decision Tree.

Model	Accuracy	FPR	FNR	F1-Score
GCN	**0.79**	**0.05**	**0.38**	**0.61**
XGBoost	0.78	0.04	0.39	0.65
Decision Tree	0.58	0.1	0.66	0.37
KNN	0.83	0.04	0.35	0.67
SVM	0.84	0.04	0.35	0.66
Logistic Regression	0.83	0.04	0.36	0.67

Table 1. Model Performance Comparison

Model A	Model B	χ^2	p-value	p <0.05
GCN	LR	19.75	8.81E-06	Yes
GCN	KNN	13.67	2.17E-04	Yes
GCN	SVM	25.88	3.61E-07	Yes
GCN	XGB	8.76	3.07E-03	Yes
GCN	DT	80.40	3.05E-19	Yes

Table 2. McNemar's Test Calculations Against GCN

References

[1] D. Antypas, A. Ushio, J. Camacho-Collados, L. Neves, V. Silva, and F. Barbieri. Twitter Topic Classification. In *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics.

[2] K. Gu. *Text-GCN: PyTorch implementation of "Graph Convolutional Networks for Text Classification."*, 2020.

[3] T. Kipf. *Graph Convolutional Networks*, 2016.

[4] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[5] Y. L. Liang Yao, Chengsheng Mao. *text_gcn: Official Implementation of TextGCN*, 2019.

[6] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377, 2019.

benjamin_bradley@brown.edu cecily_chung@brown.edu charles_duong@brown.edu