Word Embedding Fine-Tuning using Graph Neural Networks on Local Word Co-Ocurrence Graphs

Federico Reyes Gomez

Introduction

The goal of this project is to investigate whether structural information of text, represented by word co-ocurrence graphs, can be used by Graph Neural Networks (GNNs) to fine-tune pretrained word embeddings for a specific domain and document. GNNs are good at augmenting existing tasks with structural information, and at worst the fine-tuning model will make no changes to the pre-trained word embeddings, but can potentially add non-trivial amount of knowledge that will significantly improve

Task & Data

This project is based on the CS224n Default Final Project (IID Track), which includes a baseline Bi-directional Attention Flow [1] (BiDAF) model for Question Answering on the Stanford Question Answering Dataset 2.0 (SQuAD 2.0) dataset. Our approaches are meant to be comparative against a given baseline, so we kept the BiDAF model unchanged while adding one or more intermediate layers after the pretrained GloVe [2] word embeddings to attempt to fine-tune them to better suit the task at hand.

Graph Neural Networks

We used Graph Neural Networks for this project, which are Deep Learning models that take in a set of nodes and edges along with node features and generate embeddings for each node in the graph. The following is the message-passing update layer.

$$\underbrace{\mathbf{x}_i^{(k)}}_{\text{New source node embedding}} = \overbrace{\gamma^{(k)}}^{\text{Update()}} \left(\underbrace{\mathbf{x}_i^{(k-1)}}_{\text{Old source node embedding}}, \overbrace{\square_{j \in \mathcal{N}(i)}}^{\text{Aggregate()}} \overbrace{\phi^{(k)}}^{\text{Message()}} \left(\mathbf{x}_i^{(k-1)}, \underbrace{\mathbf{x}_j^{(k-1)}}_{\text{Old neighbor node embedding}}, \underbrace{\mathbf{e}_{j,i}}_{\text{Edge features}} \right) \right)$$

For the SQUAD dataset, each example consists of a context paragraph and a query paragraph. We take each paragraph and calculate a word co-ocurrence graph where two words are connected if they co-occur in the paragraph with an edge weight corresponding to their frequency.

We used a graph convolutional layer from the Graph Attention Network (GAT) [3] model given the small graph sizes and existence of edge features.

Here is the equation for the GAT Convolutional layer:

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j,$$

where the attention coefficients are calculated as follows where $e_i j$ is the word co-ocurrence count betwee word i and j:

$$\alpha_{i,j} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_j \,\|\, \mathbf{\Theta}_e \mathbf{e}_{i,j}]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_k \,\|\, \mathbf{\Theta}_e \mathbf{e}_{i,k}]\right)\right)}.$$

Model Architecture & Experiments

Experiments implemented in PyG [4]:

1. **Baseline**: Starter code model, BiDAF
2. **Intermediate Local Single-Layer GNN**: Single layer of graph convolutions between the embedding layer and downstream model.
3. **Intermediate Local Multi-Layer GNN**: Experiment (2) with multiple layers
4. **Intermediate Local Multi-Layer GNN with Skip Connection**: Experiment (3) with sum skip-connection from GloVe embeddings
5. **Intermediate Local Multi-Layer GNN with Rescaling**: Experiment (4) with rescaling GNN embeddings to be same norm as GloVe embeddings and combining embeddings using a Combination (Linear) Layer.
6. **Intermediate Local Multi-Layer GNN with Rescaling and Offset Output**: Experiment (5) with weakly conditioning y2 on y1.
7. **GNN Encoder**: Removes the encoder submodule from the BiDAF model, replaced by a GNN
8. **Intermediate Context-Query GNN Mixing**: Jointly embedding the context and query using a graph convolutional layer

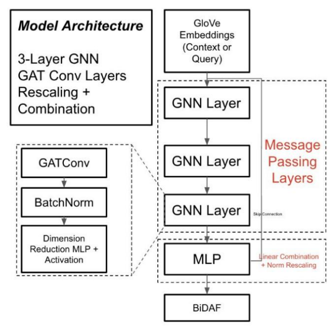

Figure 1: Model Architecture: Intermediate Local Multi-Layer GNN with Rescaling. This process is repeated for both the context and query independently and passed into the BiDAF model as normally

Experiment (Development Set Split)	F1	EM	AvNA
Lower learning rate (< 0.5)	too slow	too slow	too slow
Higher learning rate (> 0.5)	diverged	diverged	diverged
Intermediate Local Single-Layer GNN	51.47	48.97	59.57
GNN Encoder (Single Layer)	52.19	52.19	52.14
Intermediate Context-Query GNN Mixing	58.30	54.80	65.54
Intermediate Local Multi-Layer w Rescaling and Offset Output (batch 1)	60.29	57.25	66.58
Intermediate Local Multi-Layer w Rescaling and Offset Output (batch 256)	60.45	57.42	66.76
Intermediate Local Multi-Layer w Rescaling and Offset Output (batch 64)	60.48	57.47	66.91
Baseline	60.85	57.79	67.89
Intermediate Local Multi-Layer w Skip Connection	60.92	57.82	67.30
Intermediate Local Multi-Layer w Rescaling	**61.66**	**58.38**	**68.11**

Table 1: Results on dev set. Our approach didn't really work, giving minimal improvements on the baseline.

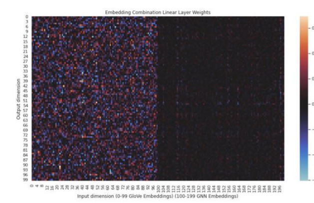

Figure 2: Heatmap visualization of Combination Layer weights, showing model learned to ignore GNN embeddings in favor of pretrained GloVe emebeddings.

Conclusion & Future Work

Generating the graphs took a while, limiting our results, but we experimented with various architectures of different depths, batch sizes, post-processing, and connectivity. A couple of model choices ended up performing better than the baseline by less than 1%, indicating weak performance. Further tuning of the GNN architecture including pre and post-processing can lead to better performance, but this sort of approach does seem fairly limited.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. CoRR, abs/1611.01603, 2016.
[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.
[3] Petar Veli˘ckovi´c, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.
[4] PyTorch Geometric creating message passing networks. https://pytorch-geometric. readthedocs.io/en/latest/notes/create_gnn.html. Accessed: 2022-02-07.