Personalized Mentoring System using Graph Neural Networks
ENGR-E 516, Spring '23
Dr. Dingwen Tao

Sreesha Srinivasan Kuruvadi, Srinivas Kini, Karan Sharma

May 4, 2023

Contents

1 Project Overview & Introduction

Social media allows us to easily access a plethora of content. For users that want to develop new skills or enhance existing skills, platforms like LinkedIn serve as a great knowledge base. One has access to articles, blog posts, research papers, videos and podcasts related to their interests shared by people of expertise in that domain.

Since this pool of content is so vast and diverse, users may feel overwhelmed and struggle to find a set of personalized resources that can help them leverage it effectively and help them grow. Our system aims to solve this problem using a cloud-based graphical machine learning (ML) model that is trained on the user's historical or current interests and provides them with an ordered set of resources that they can follow. This can be used to construct a custom knowledge graph and provide the user with learning/mastery paths to help them hone their skills, acting like an automated mentor.

2 Related Work

The study of complex graphs is a highly interdisciplinary field that aims to study complex systems by using mathematical models, physical laws, inference and learning algorithms, etc. Complex systems are often characterized by several components that interact in multiple ways with each other. Such systems are better modeled by complex graph structures such as edge and vertex-labeled graphs (e.g., knowledge graphs), attributed graphs, multilayer graphs, hypergraphs, etc.

We are focused on utilizing existing ML algorithms as well as network science measures and apply it to social media data. The essence is to track, summarize and provide recommendations to learning paths based on user intent and behavior.

A personalized knowledge graph is a knowledge graph that has been tailored to an individual user based on their preferences, interests, and behavior. Here are some examples of personalized knowledge graphs:

- Facebook News Feed: Facebook uses a personalized knowledge graph to display posts and content to each user's News Feed. The knowledge graph is based on the user's likes, interests, and behavior on the platform, as well as the behavior of their social network.
- Google Knowledge Graph: The Google Knowledge Graph is a large-scale knowledge graph that is used to enhance Google's search results and other services across multiple platforms, including web search, mobile search, and Google Assistant.
- IBM Watson: IBM Watson is a cloud-based cognitive computing platform that uses a knowledge graph to provide a range of AI-powered services, including natural language processing, machine learning, and data analytics.
- Microsoft Satori: Microsoft Satori is a knowledge graph that powers Bing search results and other Microsoft services, such as Cortana and Microsoft Office.
- Wolfram Knowledgebase: The Wolfram Knowledgebase is a vast and constantly growing knowledge graph that is used to power Wolfram Alpha, a computational knowledge engine that provides answers to questions in a wide range of domains.
- Wikidata: Wikidata is a free and open knowledge graph that provides structured data for Wikipedia and other Wikimedia projects. It is accessible via a web API and can be used by third-party applications and services.

3 System Design

3.1 Architecture

The system that utilizes a Chrome extension to extract data from the LinkedIn Feed. Once deployed, the extension begins scraping data as the user scrolls through the feed. Upon closing the tab, the collected data is hierarchically stored in an S3 bucket and used for training and inference in a Graph Neural Network (GNN) model.

The GNN model is exposed as a web service on Amazon EC2 and queried by a user dashboard deployed on Amazon ECS. The model response includes cross-recommendations, posts to revisit, and topic modelling data, which is sent to the OpenAI API to retrieve relevant knowledge sources. The collected data is then consolidated and displayed on the dashboard for user access.

This system provides a comprehensive solution for automated data extraction, modeling, and knowledge retrieval from the LinkedIn feed, facilitating efficient and effective information retrieval and decision-making processes.

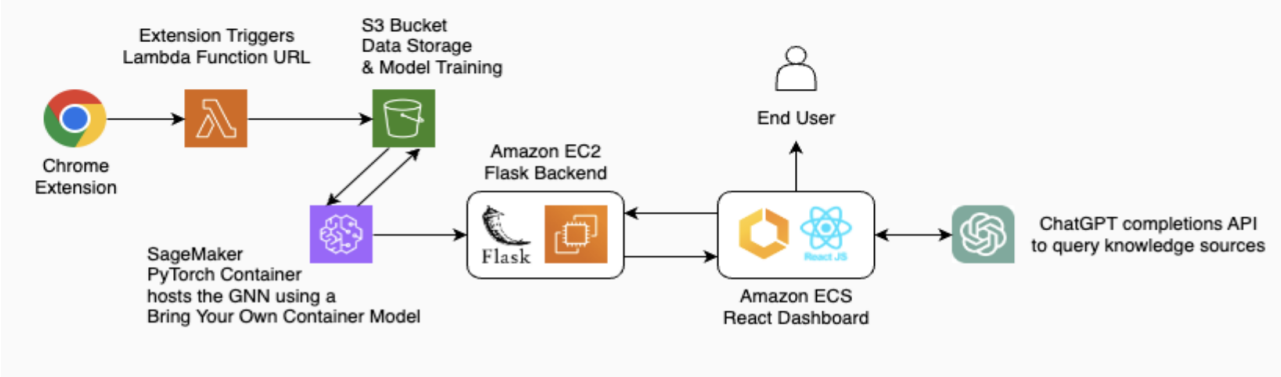

Figure 1: Complete Architecture

3.2 Chrome Extension

The extraction of data from LinkedIn feed using the Chrome extension is done with `MutationObserver` concept. In detail, the `MutationObserver` is a built-in browser API that allows the observation of changes in the DOM tree. It is commonly used in web development to detect and respond to changes in web page content without resorting to continuous polling. In this case, the `MutationObserver` is utilized to monitor changes in the LinkedIn feed content and structure, and it initiates the data scraping process when new content is detected. By using the `MutationObserver`, the data extraction process is automated, and the extracted data is more reliable and consistent than manually collected data.

The extracted data is then stored in an S3 bucket using an AWS Lambda function. This integration provides a scalable and cost-effective solution for processing and storing the scraped data. The Lambda function is triggered automatically when the Chrome extension pushes new data to the AWS S3 bucket.

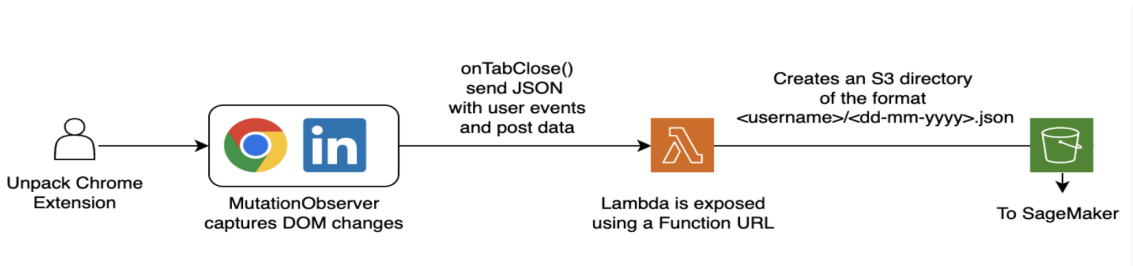

Figure 2: Chrome Extension

3.3 GCN Modelling and Architecture

- The model uses a Graph Neural Network (GNN) with input features: author name, text, user reaction, and time viewed by the user.
- The author features are one-hot encoded representations of the authors of the posts.
- The text features are embeddings of the post text generated using a pre-trained Transformer model (BERT*).
- The user features are one-hot encoded representations of the users for whom the model generates recommendations.
- The model takes in the concatenated author features, text features, and user features, and learns the relationships between the nodes (posts) in the graph using a GNN.
- The output of the model is the engagement score (number of likes, comments, etc.) of each post.
- User-post graph - Edge weights have decreased sensitivity to number of reposts, reactions and comments in that order. If a user has not reacted to the post, the edge value is pushed down to 0.
- Edge-weight - Product of below three metricsl; user-reaction - 0 or 1: reactions pow(2): comments : reposts pow(3)
- The code is built using references and architectural considerations from - https://arxiv.org/pdf/2002.02126.pdf (LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation)

*BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained Transformer model that is trained on a large corpus of text. It learns to represent words and phrases as dense vectors, which can capture semantic similarities between them. By using BERT to generate text embeddings, we can incorporate this semantic information into the GNN model. In particular, using BERT allows us to capture more nuanced relationships between posts based on the content of the text. This can help the GNN model to learn more accurate representations of the posts and to better capture the interactions between them. By incorporating BERT into the model architecture, we can potentially improve the quality of the recommendations generated by the model.

3.4 Topic Modelling

- BERT is used to generate topics and tf-idf routine reduction to ensure the topic sizes are low and relevant. The generated topics are used to label and extract top words in LinkedIn posts. Topic modeling helps find a representative set of words for each document.

3.5 Code

- The model is implemented as a class called 'LightGCN'.
- The 'init' function initializes the model with the user-embedding and message passing layers
- The 'train' function trains the GNN model on new data and updates the model's internal data with the new data.
- The 'evaluate' function evaluates the model's performance on a validation set.
- The model uses BPR-Bayesian Personalised Ranking. BPR encourages observed positive user-item predictions to have increasingly higher values than unobserved negative ones. L2 higher the penalty, prevents overfit of the model to the training data and helps generalize well.

3.6 Model Evaluation

- We utilize precision and recall. We are interested in the top 10 test precision when evaluating model performance because top 10 precision tells us "If our model were to recommend 10 LinkedIn posts to our user, what would be the average number of posts our user would like, out of the top 10?"

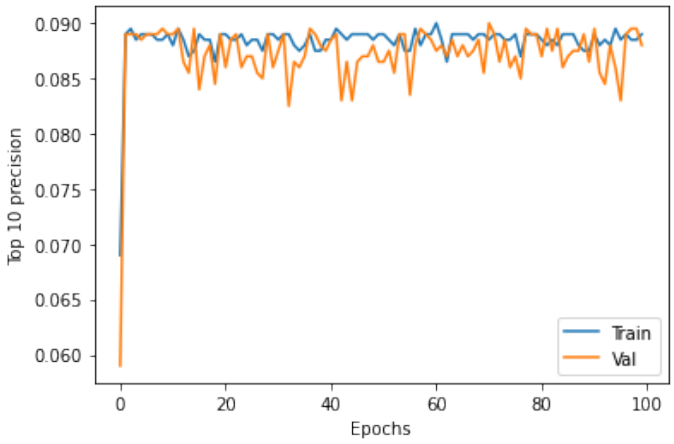

Figure 3: Precision

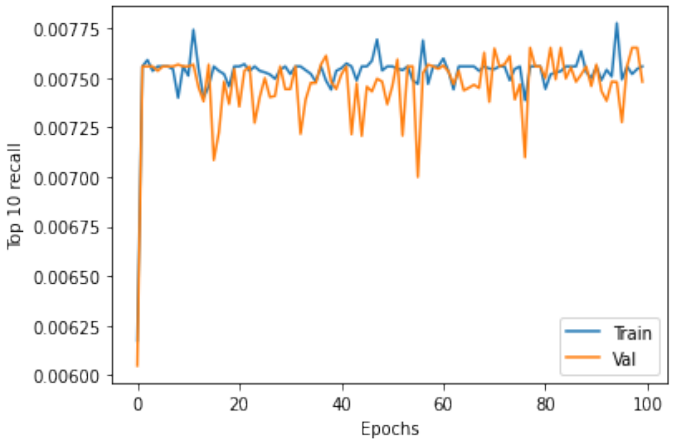

Figure 4: Recall

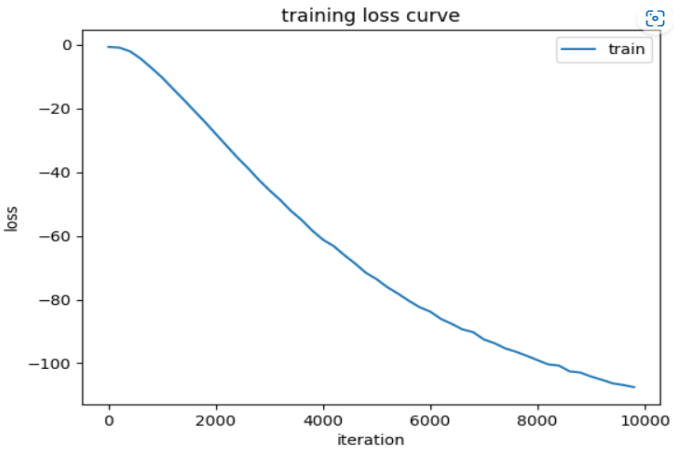

Figure 5: Traninig Loss

3.7 Inputs

- The model takes in data for each post, including author features - text, author image, author URL, text features, user features such as reaction, time that the user viewed the post.

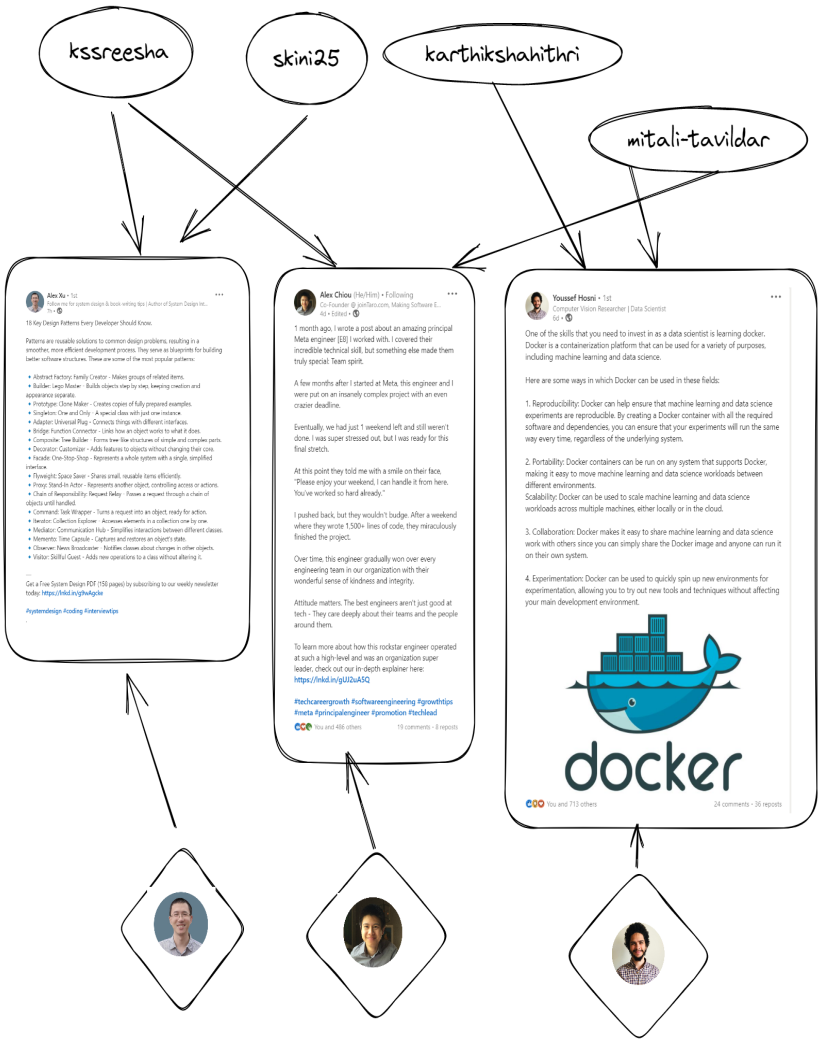

Figure 6: User Post Author graph

3.8 Ouputs

- The model learns user-post embeddings based on the reaction and post reach - comments, reactions to posts, and repost count.

3.9 Explanation

- The model uses a GCN to learn relationships between post and user interaction and generate recommendations for a given user.
- The BERT model is used to generate text embeddings, which are used as input to the GNN.
- The model can be trained on new data using the 'train' function, and can generate recommendations for a given user.
- Computationally efficient and Easier to train- LightGCN uses only neighborhood aggregation and uses multi-scale diffusion.

Figure 7: Generated topic distributions

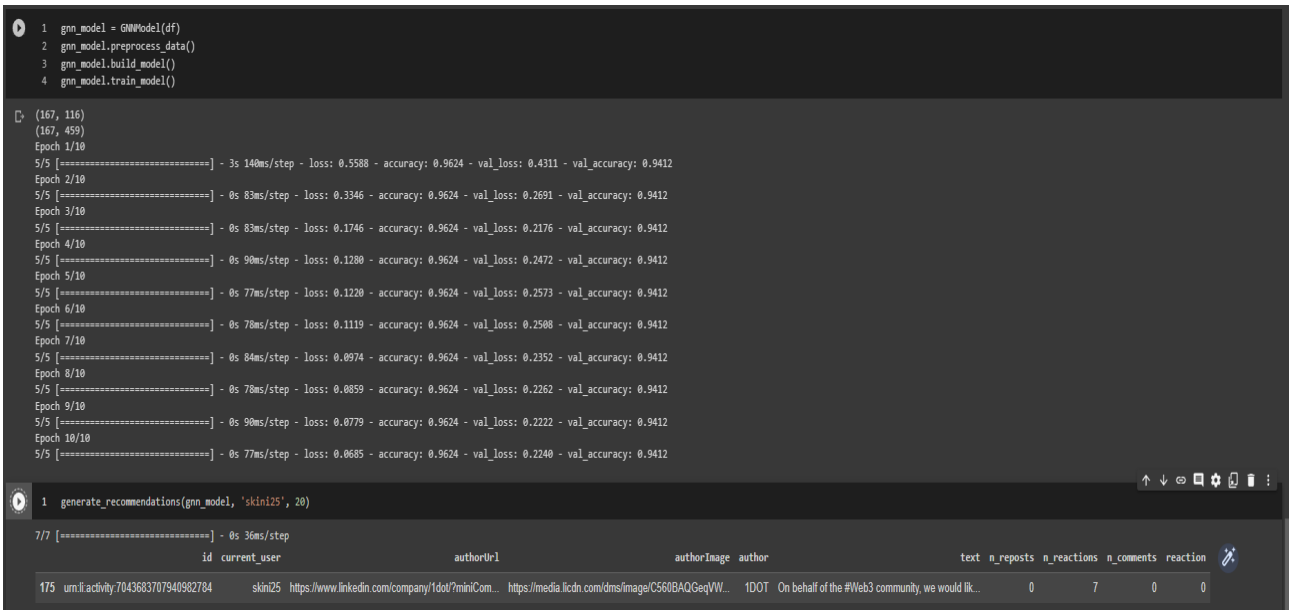

Figure 8: Recommendations

3.10 Training

- Intra-layer aggregation - Within each layer, for each user in the graph, we compute its updated embedding as the weighted sum of embeddings from all its neighboring post items
- Inter-layer aggregation -multi-scale diffusion - Instead of taking embedding of the final layer, LightGCN computes a weighted sum of the embeddings are different layers.
- Loss Aggregation and Metrics - BPR loss, Discounted cumulative gain, and NDCG
- Computationally efficient and Easier to train- LightGCN uses only neighborhood aggregation and uses multi-scale diffusion.
- Embedding layer Embedding dimension - 64, Message passing layer K = 20.
- User embedding layer Number of users, 64
- Linkedin post embedding layer Number of posts, 64

3.11 User Dashboard

The dashboard is a React application deployed on Amazon ECS that leverages a SageMaker inference endpoint deployed on Amazon EC2. The application queries the SageMaker endpoint to generate a response, which is then used to query OpenAI using the `chatCompletion` API. The OpenAI API provides knowledge sources related to the topics present in the response generated by SageMaker. The collected knowledge sources include relevant videos, research papers, blogs, and GitHub repositories, which can be utilized by users to improve their knowledge related to the topics at hand.

This system facilitates efficient and effective knowledge retrieval and provides users with a comprehensive set of resources to improve their understanding of relevant topics. The integration of multiple AWS services and the OpenAI API offers a scalable and cost-effective solution for knowledge discovery and information retrieval.

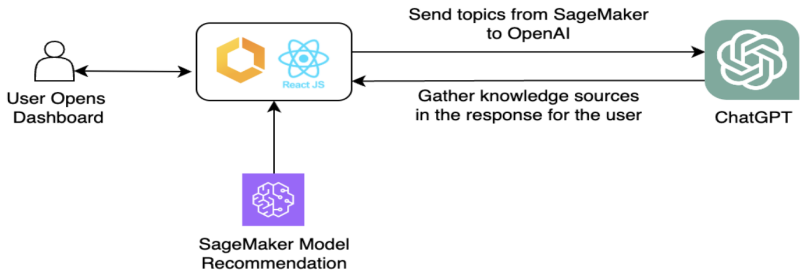

Figure 9: User Dashboard

4 Team Members & Workload Allocation

Member	Focus
Sreesha	Feature Engineering, Model Design, Data labeling, Model Resource Management, Model Delivery
Srinivas	Data Acquisition (Scraping and S3 storage), User Dashboard, Open AI recommendations
Karan	User-Dashboard

5 Important Links

- Source Code
- Video Demo
- Slides

6 References

- https://arxiv.org/pdf/2002.02126.pdf (LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation)
- https://pytorch-geometric.readthedocs.io/en/latest/index.html
- Bertopic-Pre-trained Bert model- retrained on the Linkedin posts dataset - https://pypi.org/project/bertopic