Computer Engineering Department

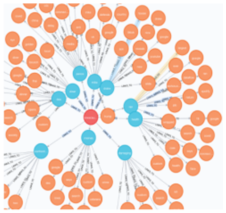
Deep Context Graph for Actionable Business Decision-making

Project Advisor: Dr. Arsanjani

Ma, Jia (MS Software Engineering)
Lai, Kevin (MS Computer Engineering)
Liu, Ying (MS Software Engineering)
Tan, Junteng (MS Software Engineering)

Introduction

Existing graph models associated with machine learning have two primary limitations. Firstly, they do not work on unseen graphs since it is mandatory for these models to train embedded graphs. Secondly, some of these models have to convert the graphs into tables and discard some of the original structures. In this master project, we are planning to explore the research in the graph deep neural network field, and aim to remove these restrictions.

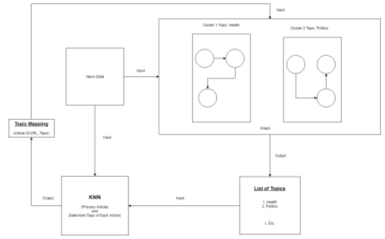


We are exploring various graph embedding models for knowledge graphs in this master project, and compare the training accuracy and performance measurement among the graph embedding models of TransE, RESCAL, and DistMult, and ComplEx.

Methodology

KNN

K-nearest neighbors (KNN) is a machine learning algorithm that performs classification of an object based on the distance between the object and other objects. In our project, we used KNN as one of the methods to predict the classification of new news articles based on the existing clusters of news articles already in our graph. For the KNN classification, we used factors such as article timestamps, article sources, and etc. Based on the news article's information, we classified the article according to how similar the article is to other articles in the graph. This similarity is represented by the computed Euclidean distance between the article and other articles. The smaller the distance the more similar the article is to the other article. The more similar an article is the more confident we are that the article belongs in a specific cluster in our graph.



Methodology

Graph Link Topic Predictor

By introducing timeline based events from news data, all the topics and articles can be sorted in a timeline based structure and grow to a graph based timeline events. We are using stellar graphs to store the events graph and map timeline and topics as source and target. All the articles that related to the topics will be converted into documents onehot encodings as node features to the graph. The graph embeddings are fed into Graph Convolutional Network and trained 50 epochs.

```
StellarGraph: Undirected multigraph
 Nodes: 91, Edges: 120

 Node types:
  news: [91]
    Features: float32 vector, length 1433
    Edge types: news-topic->news

 Edge types:
    news-topic->news: [120]
        Weights: all 1 (default)
```

NeuralCoref and AmpliGraph

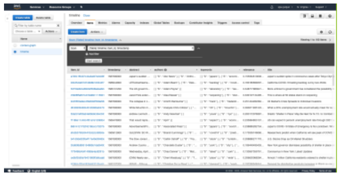
We used the scraped news data and converted it into a dataset that is made up of subject, relationship, object triplet by using SpaCy and NeuralCoref NLP libraries. Then trained a model with AmpliGraph, which is an open source graph embedding network. After our topic generator predicts a future news topic (Tracy's part), we feed the predicted topic into this AmpliGraph model and get the likelihood probability and score. The higher the number, the more likely the topic is closer to the real event.

Statement	Rank	Score	Probability
Trump supports trade war	2	0.87	0.9

Table 3: AmpliGraph Output Result

Database

We used DynamoDB and neo4j to collect, store and plot the graph.



Polynomial Equation

We parsed articles URL as the data we need for our need, including content text, title, authors, and source. It returns an output as a number between 0 - 1, the high number the more accurate. We then store these information into DynamoDB and use them to generate graphs

Here is the list for weight of each features

Feature 1.	Sentiment Analysis - 0.84
Feature 2.	LDA Topic Modelling - 0.56
Feature 3.	Sensationalism - 0.95
Feature 4.	Political Affiliation - 0.35
Feature 5.	Clickbait - 0.1
Feature 6.	Spam - 0.54
Feature 7.	Author Credibility - 0.98
Feature 8.	Source Reputation - 0.71
Feature 9.	Content Length - 0.6
Feature 10.	Word Frequency - 1
Feature 11.	Bias - 0.05

we use the youden's index equation which is shown below to calculate the weight for each feature.

$$normalized\ accuracy = \frac{TruePositive}{(TruePositive+FalseNegative)} + \frac{TrueNegative}{(TrueNegative+FalsePositive)} - 1$$

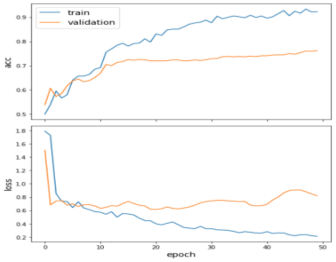
Analysis and Results

We compared four graph embedding network models, including TransE, RESCAL, DistMult, and ComlEX. Average loss, MRR score, and Hits-at-n-score, are returned for each model. MRR stands for mean reciprocal rank, which is a measure to evaluate systems that return a ranked list of answers to queries. If there is no correct answer from the query, the MRR score would be zero. The higher the MRR score is, the more accurate the model is.

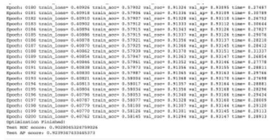
	ComlE	TransE	DistMult	RESCAL
MRR	0.30	0.14	0.27	0.40
Hits@10	0.41	0.26	0.38	0.53
Hits@3	0.33	0.16	0.29	0.45
Hits@1	0.24	0.08	0.21	0.33
Average Loss	0.021658	0.025696	0.021388	0.090501

Table 4: Embedding Network Models Scores

Below is showing accuracy and loss performance measurement of the Graph Link Topic Predictor Model. The project has achieved training accuracy of above ninety percent.



In the training step, the model is trained on Colab GPU using tensorflow 1.x. To optimize the model, we are using learning rate 0.01, and there are a total 200 epochs, weight decay and drop out are applied too.



```
Epoch: 0180 train_loss= 0.40926 train_acc= 0.57902 val_roc= 0.91324 val_ap= 0.93095 time= 0.27487
Epoch: 0181 train_loss= 0.40918 train_acc= 0.57906 val_roc= 0.91336 val_ap= 0.93109 time= 0.30789
Epoch: 0182 train_loss= 0.40910 train_acc= 0.57907 val_roc= 0.91328 val_ap= 0.93110 time= 0.26752
Epoch: 0183 train_loss= 0.40902 train_acc= 0.57912 val_roc= 0.91333 val_ap= 0.93112 time= 0.30644
Epoch: 0184 train_loss= 0.40894 train_acc= 0.57915 val_roc= 0.91343 val_ap= 0.93126 time= 0.27827
Epoch: 0185 train_loss= 0.40886 train_acc= 0.57915 val_roc= 0.91337 val_ap= 0.93126 time= 0.29076
Epoch: 0186 train_loss= 0.40878 train_acc= 0.57921 val_roc= 0.91356 val_ap= 0.93137 time= 0.28210
Epoch: 0187 train_loss= 0.40870 train_acc= 0.57925 val_roc= 0.91366 val_ap= 0.93145 time= 0.28412
Epoch: 0188 train_loss= 0.40862 train_acc= 0.57939 val_roc= 0.91378 val_ap= 0.93151 time= 0.31337
Epoch: 0189 train_loss= 0.40854 train_acc= 0.57947 val_roc= 0.91363 val_ap= 0.93147 time= 0.26854
Epoch: 0190 train_loss= 0.40846 train_acc= 0.57961 val_roc= 0.91352 val_ap= 0.93146 time= 0.27778
Epoch: 0191 train_loss= 0.40838 train_acc= 0.57973 val_roc= 0.91356 val_ap= 0.93155 time= 0.28811
Epoch: 0192 train_loss= 0.40830 train_acc= 0.57987 val_roc= 0.91365 val_ap= 0.93163 time= 0.29766
Epoch: 0193 train_loss= 0.40821 train_acc= 0.58004 val_roc= 0.91368 val_ap= 0.93170 time= 0.27698
Epoch: 0194 train_loss= 0.40813 train_acc= 0.58017 val_roc= 0.91357 val_ap= 0.93164 time= 0.28881
Epoch: 0195 train_loss= 0.40804 train_acc= 0.58034 val_roc= 0.91356 val_ap= 0.93168 time= 0.28296
Epoch: 0196 train_loss= 0.40796 train_acc= 0.58053 val_roc= 0.91348 val_ap= 0.93166 time= 0.29434
Epoch: 0197 train_loss= 0.40787 train_acc= 0.58077 val_roc= 0.91328 val_ap= 0.93168 time= 0.28009
Epoch: 0198 train_loss= 0.40779 train_acc= 0.58100 val_roc= 0.91307 val_ap= 0.93144 time= 0.29120
Epoch: 0199 train_loss= 0.40770 train_acc= 0.58126 val_roc= 0.91301 val_ap= 0.93145 time= 0.29113
Epoch: 0200 train_loss= 0.40762 train_acc= 0.58145 val_roc= 0.91294 val_ap= 0.93147 time= 0.28913
Optimization Finished!
Test ROC score: 0.9028045324759026
Test AP score: 0.9239347433445373
```

Summary/Conclusions

The goal of this project is to use existing news context to predict future events and news topics by growing a context graph and train graph convolutional networks. By building the graph, we are able to have a high level view of relationships between news topics. It's like a news search engine which can explore articles that are related to one topic and grow new articles and topics via related topics. The real world relationship will be distilled and concentrated into topics and relationships.

Key References

[1] Arsanjani, A. (2019, September 2). Deep context graphs: Enriched context for business decision-making using knowledge graphs and deep reinforcement learning.

[2] Coding Tech. (2018, June 28). Knowledge graphs & deep learning at YouTube.

[3] Alphabet Inc. (2012, May 16). Introducing the knowledge graph: things, not strings.

[4] KDD2019. (2019, July 2). Learning dynamic context graphs for predicting social events.

[5] Hamilton, W. L., Bajaj, P., Zitnik, M., Jurafsky, D., & Leskovec, J. (n.d.). Embedding logical queries on knowledge graphs.

[6] Hodler, A. E. (2019, July 29). AI and graph technology: 4 ways graphs add context

Acknowledgements

We would like to express our thanks to Dr. Arsanjani, our masters project advisor. He guided us on multiple subjects and helped to answer and resolve numerous questions/issues. He provided us with lots of related research papers and articles and communicated with us regularly and effectively. We also appreciate the teamwork between team members.