CS510 Final Report

Ti Chen, Yuyuan Tang, Qinghui Zhou

{tichen3, yuyuant2, qinghui2}@university.edu

Demo Link: https://mediaspace.illinois.edu/media/t/1_9gybbmuf
Github: https://github.com/tichen47/CS510-Project
Host: http://18.190.124.145/

1 Description

Our team aim at establishing a Literature Search Engine system integrated with recommendation systems. We all know that it is important for researchers to jump out of the circle to explore new ideas in a relevant topic. Thus, we design a search engine not only can enable the paper search by queries but also can provide a chance for users to explore relevant sources based on his/her current search.

2 Uniqueness

The unique features that our search engine provides is that it could recommend not only papers that the client might be interested in but also the venus or the authors the user may want to explore. Specifically, given an input that is either the name of the paper, or an author, the recommendation system would recommend based on the cosine similarity of node embeddings of either venues or authors and output list of nodes that lies in the affinity of the node embedding of the search query. Thus the user could explore the papers or venues based on this chained relationship. Which is better than homogeneous recommendation such as only display the search query result, which consists of only papers.

3 Algorithms

3.1 Search Algorithm

To deal with the search results ranking problem, the team first chose to use the BM25 in the pre-project stage to approximate the relevant score of a document. The default BM25 algorithm incorporates term frequency, inverse document frequency and document length normalization which penalizes longer documents:

$$\frac{(k_3+1)q}{(k_3+q)} \cdot \frac{(k_1+1)f}{(K+f)} \cdot \log \frac{(r+0.5)(N-n-R+r+0.5)}{(n-r+0.5)(R-r+0.5)}$$

where R is the number of document known to be relevant to a specific topic and r is the number of relevant documents containing the term. Since R and r are not provided, we set them as zero.

To improve the performance of the baseline ranking function, the team update the BM25 model to BM25L model [1]. BM25L proposed by Lv&Zhai[] solve the problem of BM25 that the document length normalization L_d/L_{avg} prefers shorter documents to longer ones. They first re-arange the formula of BM25 to

$$rev_q = \sum_{t \in q} log(\frac{N+1}{df_t + 0.5} \cdot \frac{(k_1+1)c_{td}}{k_1 + c_{td}})$$

where

$$c_{td} = \frac{tf_{td}}{1 - b + b \cdot (\frac{L_d}{L_{avg}})}$$

and add a positive constant, δ, to it:

$$rev_q = \sum_{t \in q} log(\frac{N+1}{df_t + 0.5} \cdot \frac{(k_1+1)(c_{td}+\delta)}{k_1 + (c_{td}+\delta)})$$

The team found that BM25L performs better than the BM25 model.

3.2 Relevant Recommendations

Our search engine is also able to provide exploration to users based on their search results. More specifically, the search engine can make recommendations based on searched papers on relevant papers as well as authors who also write papers in the similar area. We treat this recommendation problem as a graph representation task to solve by constructing and embedding a heterogeneous networks of authors and papers. In this section, we will first introduce how we construct an academic graph. And then, we will discuss how we compute the representation of authors and papers based on the graph, and the way to use the similarity of their representations to get the final recommendations.

We use the AMiner dataset [2] to construct our heterogeneous graph. In details, we extract the conference and authors of several million papers in this dataset. Each conference, author, and paper is treated as a node in the heterogeneous graph, and the relationship between a paper and its author, or a paper and its conference is mapped to this graph as edges.

Then we implement a scalable representation learning, metapath2vec [3], for our heterogeneous network. Although the node in our heterogeneous graph represent different entities, this graph learning algorithm will help us to map these different entities to the same low-dimensional latent space, which will further benefit us in the following stage of recommendation. This metapath2vec algorithm is derived from a traditional model in Neural Language Processing, named word2vec [4, 5], which is used to learn the representations of words given a corpus. Based on similar technique, the metapath2vec use the skip-gram model to obtain the node representation from random walks generated using the given graph. Usually in a homogeneous network, the objective is to maximize the probability of local structures of networks:

$$argmax_\theta \prod_{v \in V} \prod_{c \in N(v)} p(c|v;\theta)$$

, where $N(v)$ is the set of neighbor nodes of node v and $p(c|v;\theta)$ measures the conditional probability of context node c given a node v. And when consider the heterogenous nodes in the graph, the metapath2vec extend the above formula to maximize the probability of heterogeneous context $N_t(v)$ given the node v where t is one type of the context T_V:

$$argmax_\theta \prod_{v \in V} \prod_{t \in T_V} \prod_{c_t \in N_t(v)} p(c_t|v;\theta)$$

where $N_T(v)$ is the t type of neighbors of node v, and $p(c_t|v;\theta)$ is conditional probability of having a t type of context node c given v. And $p(c_t|v;\theta)$ is usually defined as a softmax function

$$\frac{e^{Xc_t \cdot X_v}}{\sum u \in V e^{X_u \cdot X_v}}$$

where X_v is the embedding vector for node v.

After learning all embedding vector for every node, we use the cosine similarity to compute the distance between each pair of nodes. Thus, for recommendation, we can determine the distance between one paper and all other paper as well as users. We treat such similarity as recommendation scores and sort those scores to get the top 5 relevant papers and authors.

4 backend implementation

4.1 Data Pre-processing

Converted data to be readable by MetaPy line by line. Then basic configurations were set up and inverted indexes were made. For the Metapath2Vec training, the input Aminer data were first parsed and split into files of heterogeneous pairs of (authorId, author), (conferenceId, conference), (paperId, paper), (paperId, authorId), (paperId, conferenceId) pairs.

4.2 Framework setup

The static backend was generated with Flask framework and the frontend is implemented with basic HTML and CSS. The website was deployed to AWS lightsail for demo purpose.

5 System tutorial

5.1 Requirement

- Python version: 2.7 and 3.6
- Install required package: 'pip install -r requirement.txt'
- Required data set:
  - Link: https://www.openacademic.ai/oag/
  - Download aminer_paper_0 and unzip file

  - Rename unzipped folder to 'dataset'
  - Place under 'data' folder

- Create 'net_aminer' under 'data' folder

5.2 Prepossess Data

- Under 'data' folder
- Run 'preprocess.py'
- New data should under the 'net_aminer' folder
- Run 'python py4genMetaPaths.py 1000 100 net_aminer output.aminer.w1000.l100.txt' (this should use python2.7)

5.3 Run the program

- Under root path, run 'python app.py'

References

[1] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, page 58. ACM, 2014.

[2] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[3] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144. ACM, 2017.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.