AcademicRank: Rank Academic Works on Specific Field of Study

Haozhe Si (NetID: haozhes3)

May 2021

1 Introduction

Millions ans billions of academic works are there on the internet. To find the most classical reference about a specific keyword, people would usually search for that keyword on websites like Google Scholar. This searching strategy seems to be reasonable, however, this may not be the case. Searching engines like Google Scholar performs "title search"[1], that is to say they would prefer to rank the academic works whose titles contain the keywords we are searching for higher, but the field of study of an academic work may not appears in its title. This motivated us to build a more precise ranking algorithm that can take the field of study into consider when ranking the academic works given a keyword instead of solely rank the papers filtered out by titles. In addition, we made the assumption that a classical academic work would not only relate to the its citation number, but also how important the citing sources are. Thus, this becomes very intuitive for us building our model on the PageRank algorithm[1]. To sum up, in this project, we are going to build a directed graph of citation where nodes are academic works and the edges represents the citation relations, and perform a PageRank calculation on that graph. To simplify our task, we limited the academic works in Computer Science domain and so does the keywords. The specific approaches are discussed in the following sections.

2 Approaches

2.1 Dataset Selection

To achieve the goal we mentioned above, we need to acquire the citation relationship between the Computer Science papers. Fortunately, there are two datasets, **unarXive**[2] and **Microsoft Academic Graph(MAG)**[4], that provides this information and we tried to implement our task on both of the two datasets. We modified our implementation base on the data provided by the two datasets slightly. The details are discussed below.

2.1.1 unarXive

unarXive is a data set based on all publications from all scientific disciplines available on arXiv.org[2]. It provides the papers' plain text and in-text citations annotations, which essentially provides the citation relationship we need and the context (the sentence containing the citation mark and the sentences before and after that sentence) of each citation. We take the context of citation as anchor text, and tried to extract Computer Science keywords from that context to assign field of study of the cited paper. After processing the large datasets and taking out what we need, we have two major datasets, one keeps the citing and cited paper relationship, as well as counting how many time such pair exists; another keeps the fields of studies of each paper. We planned to implement a Weighted PageRank algorithm by first filtering out all the directed edges whose destination(cited paper) contains the target keyword in its fields of study. Then, we assign the weight base on how many times such citing-cited relationship occurs and perform the ranking algorithm. This seems to be an feasible solution. However, the unarXive dataset has a serious issue, that is arXiv.org only contains a small number of papers comparing to all of the Computer Science works. Therefore, given all of the citing papers are on arXiv.org, only a small part of the cited papers is on arXiv.org. The authors of the dataset also noticed this problem, thus, they connected the cited papers to MAG, which is way larger than arXiv. This missing data caused our graph being incomplete and having a lot of dangling nodes, which seriously affected the performance of ranking algorithm. Hence, although it is precious that unarXive provides context information for us so that we can use anchor text to find the field of studies, we had to give this dataset up due to the limitation of dataset size.

2.1.2 Microsoft Academic Graph(MAG)

The **Microsoft Academic Graph** is a heterogeneous graph containing scientific publication records, citation relationships between those publications, as well as authors, institutions, journals, conferences, and fields of study[4]. It is a much larger datasets with sufficient information provided. Among the datasets, we specifically choose three of them: (a) FieldsOfStudy, which lists the information for fields of study; (b) PaperFieldsOfStudy, which saves the fields of study of each papers, notice that each paper may have multiple fields of study; (c) PaperReferences, which contains the citation relationships of the papers. Unlike arXiv, MAG is a collection of all the scientific papers instead of Computer Science only, and thus, there are more than one billion items in the second and third datasets. It is necessary for us to filter out the Computer Science keywords and prune out papers that are not related to Computer Science. The details for pruning will be discussed in following sections. Another difference of implementation detail to using the unarXive dataset is that we no longer have reliable citation context information. That means we can only use the field of study information MAG provides us, instead of using anchor text as we used in unarXiv. Considering the strong connection and abundant information provided by MAG, we finally chose this dataset.

2.2 Dataset Pre-process

As we mentioned, some data in MAG dataset is redundant for our task of ranking Computer Science academic works. Thus, we need to prune the datasets first. We perform the pruning in three steps:

- **Prune Fields of Study**: The FieldsOfStudy dataset has 741K items. To filter out the Computer Science keywords, we selected a computer science keyword dataset with 83K items collected from Springer by Y. Peng in the FORWARD Data Lab. We first try to check if the field of study is in the CS keyword dataset, if so, we will keep that keyword. The resulting pruned dataset has 25K items. However, when we inspecting the new field of study dataset, we found that some of the keywords that were originally in the CS keyword dataset are not actually CS keywords, such as "medicine" or "crystal". This is possible because CS techniques can be used in other disciplines, which make keywords from other disciplines being collected and considered as CS keywords. However, keeping such keywords will resulting inefficient pruning of papers and edges, since a lot of papers will have "medicine" as their field of study, but they may not relate to Computer Science at all. Thus we made an assumption that in the CS keyword dataset, the keywords with multiple words will be more descriptive while the single word keyword, being general, can be also pruned; meanwhile, the keywords from other disciplines will mainly be the general keywords since they have higher chance to exist in CS papers and thus can be collected into the initial dataset. The resulting dataset has 21K items.
- **Prune Papers' Fields of Study**: Having the pruned field of study set, we than iterate through the PaperFieldsOfStudy dataset and check if the paper's field of study is about Computer Science. After pruning out the fields of studies that are unrelated to Computer Science, we reduced the number of items of the dataset from 1.4 billion t0 356M.
- **Prune Paper References** From the pruned papers' field of study dataset, we can generate a set of papers that are containing CS fields of study. This set of papers about CS can be used to prune out the papers' citation relationship: if either the citing work or the cited work is not in the CS paper set, we are going to discard the item. Building such citation relation dataset only for CS leads to a drop of items from 1.67 billion to 1.09 billion.

Such pruning for dataset takes about two hours to be finished. Fortunately, we only need to perform this pruning once and it will save much time in ranking calculation.

2.3 Ranking Algorithm Implementation

Since the PageRank[1] algorithm being introduced in 1998 by Larry Page, it has been one of the most classical ranking algorithm. PageRank algorithm, as well as Personalized PageRank, are wildly used in ranking problems for web pages, where each page is considered equally important. Meanwhile, algorithm for calculating PageRank for weighted items was also purposed[5] by , which, as indicated, can calculated the rank where weights are involved. As stated before, our project is correlated with the weighted ranking algorithm. Before actually applying the ranking algorithm, we need to translate the datasets into a weighted Personalized PageRank setting first.

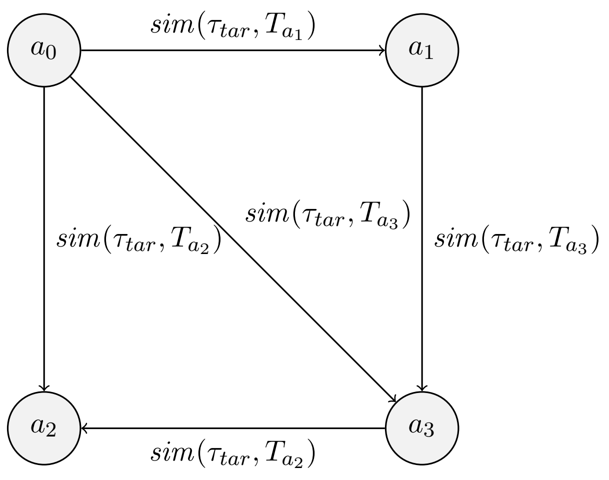
a_0
a_1
a_2
a_3
$sim(\tau_{tar}, T_{a_1})$
$sim(\tau_{tar}, T_{a_2})$
$sim(\tau_{tar}, T_{a_3})$
$sim(\tau_{tar}, T_{a_3})$
$sim(\tau_{tar}, T_{a_2})$

Figure 1: An example for a established citation net. a_i means an arbitrary article. The directed edges interpret that the source articles are citing the destination articles. τ means a single keyword. τ_{tar} means the keyword we are searching for and $T_{a_i} = \{\tau_0, \cdots, \tau_n\}$ is the set of fields of study for article a_i. $sim(\tau_{tar}, T_{a_i})$ is calculated by $\max_{\tau \in T_{a_i}} \texttt{similarity}(\tau_{tar}, \tau)$ and the word-wise similarity is calculated using *word2vec* model.

2.3.1 Setup

Algorithm 1 shows the pseudo code for calculating the Personalized PageRank[1]. In the original Personalized PageRank algorithm, we perform the power iteration, as **Line 4** shows, until the rank is converge. Here, we would like to further inspect what does this power iteration do to each entry in the rank.

Algorithm 1 Personalize PageRank

```
1: $R_0 \leftarrow E$
2: $\delta \leftarrow +\infty$
3: while $\delta > \varepsilon$ do
4:     $R_{i+1} \leftarrow AR_{i+1}$
5:     $d \leftarrow \|R_i\|_1 - \|R_{i+1}\|_1$
6:     $R_{i+1} \leftarrow R_{i+1} + dE$
7:     $\delta \leftarrow \|R_{i+1} - R_i\|_1$
8: end while
9: return $R$
```

Given the update rule for rank $R_{t+1} \leftarrow AR_{t+1}$, we can write out how the i-th entry a_i is updated:

$$R_{t+1}[a_i] = \sum_{a_j \in p(a_i)} A[a_j, a_i] \cdot R_t[a_j] \tag{1}$$

Where R_t is the rank in current round of update and R_{t+1} is the rank we are calculating; A represents the citation relationships as well as the weight information; $p(a_i)$ means all the parents node of a_i. Therefore, we can conclude that, a rank of a given node is the sum of a discounted rank of its parents. The next step is how to assign the discounted coefficient. That is to say, given we are at a random node, how should we distribute the rank of the node to its children. In our implementation, we chose to assign the weight using the relativity of our target keyword(the keyword we are searching) and the paper. To be specific, the relativity is defined as the highest similarity between the target keyword and all the fields of study of a given paper, where the similarity is calculated using the *word2vec* model. Figure 1 will be an example for a weighted directed graph for representing the citation network. In such setting, we will re-write our update equation in the way that:

$$R_{t+1}[a_i] = \sum_{a_j \in p(a_i)} \frac{sim(\tau_{tar}, T_{a_i})}{\sum_{a_k \in c(a_j)} sim(\tau_{tar}, T_{a_k})} \cdot R_t[a_j] \tag{2}$$

Where $sim(\tau_{tar}, T_{a_i}) = \max_{\tau \in T_{a_i}} \texttt{similarity}(\tau_{tar}, \tau)$ and $c(a_j)$ means all the children of a_j.

2.3.2 Rank Initialization

Notice the **Line 1** of **Algorithm 1**, we initialized R_0 using E. The E vector in our setting can intuitively corresponds to the relativity of the paper and the target keyword. In other words, we would assume the papers that are more relative to our target keyword will have higher initial rank. Thus, we calculated the relativity of each papers to the keyword as we did before:

$$R_0[a_i] = sim(\tau_{tar}, T_{a_i}),\ \forall a_i \tag{3}$$

Such initialization of R also implies that instead of treating rank as probability as in [1], we would consider the rank as the relativity, and what we are calculating for using the ranking algorithm is the most relative paper to the target keyword.

2.3.3 Relativity Redistribution Assumption

In the web page setting of Personalized PageRank, we need to repeat the power iteration until the rank is converge. Such iteration is mathematically reasonable because, as stated before, the rank in [1] is considered as probability. Probability for the ranks always sum to one after normalization, which ensures the rank value will not explode after rounds of iteration. However, it is a different case in the academic rank setting. The rank in our task is considered as relativity and normalizing such relativity has no mathematical meaning. In addition, after one iteration of update, the mathematical meaning of the rank would change a concrete idea of similarity between keywords to an abstract idea of re-distributing such similarities. Thus, unlike the original PageRank algorithm, we cannot normalize the rank nor can we treat the the rank before and after a iteration the same. To circumvent the conceptual confusion, we made an assumption that the rank algorithm in our task can only be done once and then we can get a reasonable result. With this assumption, we can remove the normalization steps for the rank R since normalization for relativity does not make sense. The Personalize PageRank algorithm with our assumption now is essentially distribute the initial rank of a node to all its children according to how similar its fields of the study and the target keywords are; while each node accumulates such re-distributed rank, new rank would be calculated. Thus, the new update equation will be

$$R[a_i] = \sum_{a_j \in p(a_i)} \frac{sim(\tau_{tar}, T_{a_i})}{\sum_{a_k \in c(a_j)} sim(\tau_{tar}, T_{a_k})} \cdot R_0[a_j] \tag{4}$$

Plug (3) into (4), we can have

$$R[a_i] = \sum_{a_j \in p(a_i)} \frac{sim(\tau_{tar}, T_{a_i})}{\sum_{a_k \in c(a_j)} sim(\tau_{tar}, T_{a_k})} \cdot sim(\tau_{tar}, T_{a_i}) \tag{5}$$

Therefore, we can calculate the rank for the papers purely using the similarity between their fields of study and the target keyword. That makes the algorithm highly rely on the performance of the *word2vec* model.

From (5) we can see that all the information we need to calculate the rank is the similarity between their fields of study and the target keyword. Since we already have the Pruned Field of Study dataset, we can pre-calculate all the `similarity`(τ_{tar}, τ) for τ in the dataset. Then, given the Pruned Papers' Fields of Study dataset, we can find the $\max_{\tau \in T}$ `similarity`(τ_{tar}, τ) for each paper given its set of fields of study T. Then, we can build a lookup table where given the paper a_i, we can find $sim(\tau_{tar}, T_{a_i})$ in $O(1)$. These steps above needs to be done for each target keyword.

2.3.4 Linear-time Implementation

In past works, people would build matrices and vectors for PageRank calculation. This is reasonable because the adjacency matrices A describing the networks are usually sparse matrices, which is fast to calculate, and the matrices calculation can be further speed up using various tricks or implementations[3, 6]. However, building the adjacency matrices, which requires going through all the edge information of the graph, take $O(|E|)$ time. Since we are only calculating the rank for one time, we came up with an algorithm that will accumulate the rank for each cited paper as the edge data being read in. In our algorithm, each edge will be accessed twice and thus the runtime for the ranking algorithm will be $O(2|E|) = O(|E|)$.

Algorithm 2 is our purposed algorithm. It is easy to observe that each (a_j, a_i) pair is visited exactly twice. In the pseudo code, **Line 3-6** is preparing for calculating the normalization and **Line 7-9** is performing one addition in (5). $sim[a_i]$ is the dictionary that contains the max similarity between the target keyword and the fields of study of a_i.

The outer loop of **Algoritm 2** is iterating all the source nodes in the citation relationship graph, while given a source node we will iterate through all its children nodes. This order of iteration

Algorithm 2 AcademicRank

```
1: $R \leftarrow dict()$
2: for $a_j$ in $PaperSet$ do
3:     $sum\_child \leftarrow 0$
4:     for $a_i$ in $child(a_j)$ do
5:         $sum\_child \leftarrow sum\_child + sim[a_i]$
6:     end for
7:     for $a_i$ in $child(a_j)$ do
8:         $R[a_i] \leftarrow R[a_i] + \frac{sim[a_i]}{sum\_child} \cdot sim[a_j]$
9:     end for
10: end for
11: return $R$
```

surprisingly matches how the data is ordered in the PaperReferences dataset, where data is ordered by the Id of the citing papers. Taking the the advantage of the data order, we can load in the dataset line by line and check if the source node has been changed, if not, we will perform **Line 3-6** and if so, we will perform **Line 7-9** for the old source in addition, Also, in real implementation, if we only want to rank the papers that contain the target keyword in their fields of study, we can only accumulate the ranks for these papers and the running time and memory usage can be further improved. **Algorithm 3** is our resulting implementation.

Algorithm 3 AcademicRank on MAG

```
1: $R \leftarrow dict()$
2: $old\_src \leftarrow None$
3: for $src, dst$ in $PaperReferences$ do
4:     if $src \neq old\_src$ and $old\_src \neq None$ then
5:         for $old\_dst$ in $child(old\_src)$ do
6:             if $\tau_{tar}$ in $T_{old\_dst}$ then
7:                 $R[old\_dst] \leftarrow R[old\_dst] + \frac{sim[old\_dst]}{sum\_child} \cdot sim[old\_src]$
8:             end if
9:         end for
10:     end if
11:     $sum\_child \leftarrow 0$
12:     $old\_src \leftarrow src$
13:     $sum\_child \leftarrow sum\_child + sim[dst]$
14: end for
15: return $R$
```

3 Result

3.1 Data Pruning Result

Table 1 shows the resulting dataset size after being pruned. We can see from the table that using multi-word keywords only will lead to a drop of items in the FieldsOfStudy dataset and even larger drops in PaperFieldsOfStudy and PaperReferencecs datasets, which proves the effectiveness of pruning. Notice that in addition to the **Springer(83K)** CS keyword set, we also performed the pruning using **Aniner-MAG(82K)** set. This comparison will be explained in section 4.1.

Dataset Size Comparison

CS Keyword Set		FieldsOfStudy	PaperFieldsOfStudy	PaperReferences	Effective
		741K	1.40G	1.67G	
Aminer-MAG	*all*	52K	1.15G	1.61G	-
(82K)	*multi*	43K	564M	1.46G	3839
Springer	*all*	25K	777M	1.66G	-
(83K)	*multi*	21K	356M	1.09G	3996

Table 1: The size of each datasets after being pruned using the selected CS keyword set. The CS keyword set are collected from Aminer-MAG and Springer respectively, and *all* mean using all the keywords while *multi* means using only keywords consists of multiple words. The numbers mean the number of items in the dataset. The Effective column implies how many keywords in the resulting FieldsOfStudy dataset is also in the vocabulary of the word2vec model, and thus can effectively calculate the similarity.

3.2 Ranking Result

We tried to use our implementation to find the top 10 academic works that are relate to the keywords "data mining", "information retrieval", "computer architecture" and "machine learning". We are only keeping the papers that actually containing these keywords as their fields of study. For the fields of study that are not in the vocabulary of the *word2vec* model and thus similarity cannot be calculated, we arbitrarily assigning similarity as 0.001. This arbitrary assigning of weight makes this experiment not guaranteed as we hope, but its comparison with simply ranking using citation numbers shows that our implementation is not trivial.

AcademicRank	Citation Count
Fuzzy sets	**Latent dirichlet allocation**
Data Mining: Concepts and Techniques	Data Mining: Practical Machine Learning Tools and Techniques
Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information	**Statistical Analysis with Missing Data**
Mining association rules between sets of items in large databases	PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses
Latent dirichlet allocation	Visualizing Data using t-SNE
Data Mining: Practical Machine Learning Tools and Techniques	Community structure in social and biological networks
A manual for repertory grid technique	RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies.
Querying and mining of time series data: experimental comparison of representations and distance measures	Fast Algorithms for Mining Association Rules in Large Databases
Statistical Analysis with Missing Data	Categorical Data Analysis
Data Mining	Cluster Analysis

Table 2: Top 10 ranks for *Data Mining* using our algorithm and direct citation count. Bold titles shows the academic works that exist in both ranking results.

AcademicRank	Citation Count
The anatomy of a large-scale hypertextual Web search engine	**Indexing by Latent Semantic Analysis**
ImageNet: A large-scale hierarchical image database	**WordNet : an electronic lexical database**
WordNet : an electronic lexical database	The scree test for the number of factors
Image Analysis and Mathematical Morphology	Qualitative data analysis: a sourcebook of new methods
Indexing by Latent Semantic Analysis	DAVID: Database for Annotation, Visualization, and Integrated Discovery
Introduction to Modern Information Retrieval	The RAST Server: Rapid Annotations using Subsystems Technology
Data clustering: a review	DBpedia: a nucleus for a web of open data
Three Approaches to Qualitative Content Analysis	Detecting influenza epidemics using search engine query data
Modern Information Retrieval	Dali server: conservation mapping in 3D
The Sequence Alignment/Map format and SAMtools	GroupLens: applying collaborative filtering to Usenet news

Table 3: Top 10 ranks for *Information Retrieval* using our algorithm and direct citation count. Bold titles shows the academic works that exist in both ranking results.

Table 2,3,4,5 show the top 10 ranks for the four selected keywords. We bold the titles of common academic works in the two ranking methods to show that the resulting paper and order of our algorithm is largely different with the result got from counting citation. Therefore, the works with high citation numbers will not always be the most classical one, we expects the works being cited by more related works having higher ranks. Also, we can see that the title of the highly ranked works do not necessarily containing the keyword we are looking for. This is what we our desired result since the ranking is for what the works are about instead of whether the keywords are contained in their title.

AcademicRank	Citation Count
RAID: high-performance, reliable secondary storage	**Introduction to VLSI systems**
Overview of the High Efficiency Video Coding (HEVC) Standard	CMOS VLSI Design: A Circuits and Systems Perspective
Caffe: Convolutional Architecture for Fast Feature Embedding	Computer Architecture: A Quantitative Approach, 2nd Edition
Overview of the Scalable Video Coding Extension of the H.264/AVC Standard	Introduction to the cell multiprocessor
Cognitive Radio An Integrated Agent Architecture for Software Defined Radio	Computer architecture (2nd ed.): a quantitative approach
Introduction to VLSI systems	The structure of the “THE”-multiprogramming system
Why systolic architectures	Garp: a MIPS processor with a reconfigurable coprocessor
The gem5 simulator	DIVA: a reliable substrate for deep submicron microarchitecture design
SimpleScalar: an infrastructure for computer system modeling	The case for a single-chip multiprocessor
Reconfigurable computing: a survey of systems and software	ANGSD: Analysis of Next Generation Sequencing Data

Table 4: Top 10 ranks for *Computer Architecture* using our algorithm and direct citation count. Bold titles shows the academic works that exist in both ranking results.

AcademicRank	Citation Count
Genetic algorithms in search, optimization, and machine learning	**The Nature of Statistical Learning Theory**
Distinctive Image Features from Scale-Invariant Keypoints	An introduction to ROC analysis
The Nature of Statistical Learning Theory	Visualizing and Understanding Convolutional Networks
Particle swarm optimization	A comparison of methods for multiclass support vector machines
Neural Networks: A Comprehensive Foundation	Neural network design
LIBSVM: A library for support vector machines	Statistical Inference
Understanding the difficulty of training deep feedforward neural networks	Natural Language Processing (Almost) from Scratch
Deep Residual Learning for Image Recognition	Instance-Based Learning Algorithms
ImageNet Classification with Deep Convolutional Neural Networks	Comparison of Multiobjective Evolutionary Algorithms: Empirical Results
Statistical learning theory	Fast learning in networks of locally-tuned processing units

Table 5: Top 10 ranks for *Machine Learning* using our algorithm and direct citation count. Bold titles shows the academic works that exist in both ranking results.

One thing to notice that, since MAG dataset also contains the information for book and reviews, therefore, strictly speaking, we are calculating the ranking for CS works instead of papers only. One can also filter out the papers at the **Line 6** of **Algorithm 3** to ensure that only papers will be in the ranking result.

4 Discussion

4.1 Computer Science Keyword Dataset Selection

To filter out the CS keywords from the FieldsOfStudy dataset, we need to select a CS keyword set first. There are two such set provided: Aminer-MAG CS keyword set and Springer CS keyword set. Aminer-MAG CS keyword set contains the keywords collected from the fields of study in MAG, therefore, theoretically, it should be a more suitable keyword set in our task, since we are running our ranking algorithm on MAG datasets. This can be seen from using Aminer-MAG keyword set will result in a larger pruned FieldsOfStudy dataset than using Springer keyword set as shown in Table 4.1. However, If we see the **Effective** column, we can see that the Aminer-MAG case has fewer effective keywords than the Springer case. Given the **Effective** column records the number of keywords that can be calculated the similarity using the *word2vec* model, the larger number of effective keywords in each pruned FieldsOfStudy is more important. Therefore, we chose the Springer CS keyword set due to its slightly more effective keywords.

4.2 *word2vec* Model

Although we chose the Springer CS keyword set for now, the 3996 effective keywords is far from sufficient given the 21K keywords in the FieldOfStudy dataset. This insufficient effective keywords leads to the incapability of calculating word similarity and thus the weights for the citation graph are not complete in when calculating the examples above. The insufficient effective keywords in due to the mismatch of the *word2vec* model's vocabulary and our CS keyword set. The *word2vec* model we currently use is trained using the abstracts of papers in arXiv, and results in a model with 42K vocabularies. Meanwhile, our task runs on MAG dataset. Such mismatch of training and task domain may leads to the insufficient effective keywords. This problem can be solved by training on larger corpus(e.g. full-text on arXiv) or abstracts from MAG dataset.

4.3 Overall Assessment

Our AcademicRank algorithm is build on a definition of rank and an assumption about that: we defined the rank to be the relativity of the paper to the target keyword, where the relativity is calculated by measuring the similarity of the paper's fields of study and the target keyword; we assume such definition of relativity prevent the algorithm to be run multiple rounds until converge and take the result after performing power iteration one time as our final result. However, this assumption is made base on the concept of relativity while the concept of relativity in this task is also defined by us. Thus, more experiments are needed to show if the one-shot algorithm in calculating the rank is enough and if not so, how can we interpret the rank to make the power iteration mathematically reasonable. In addition to providing the conceptual build up for the ranking task, our progress in this project is more about building pipelines for calculating rank on MAG dataset, which including translating the task in to a graph, pruning the datasets and running the ranking algorithm. One of the major challenges in this task is the large dataset leads to slow calculation. Our purposed ranking algorithm runs in linear time and can be run multiple times as long as we accumulate all the ranks and use them as the initial rank in the next round, which still can be done in linear time. The result we get right now is not guaranteed and can be improved after we have a better *word2vec* model and verify our assumptions.

4.4 Future Plan

First of all, we need a *word2vec* model with larger vocabulary to increase the number of effective keywords and build a more complete graph. Second, we need to verify the relativity re-distribution assumption, by checking whether running our algorithm once is sufficient to calculate the rank. If it is not sufficient, the definition of rank in this task is also needed to be re-considered. Our implementation present in this report inclines to solve the ranking problem in a shorter time. Since the running time to solve the ranking for one keyword still takes about 2 hours right now in our setting, coming up with faster implementations is always needed in this task. Finally, given our current result of ranking is not guaranteed, we did not define a metric to measure the performance of the algorithm, and this may needs to be done in future if more stable implementation is proposed.

5 Reflection

The most important thing I learnt in this project is that, as engineers, we need to convert the problems from natural language to math model, than we can solve the challenges systematically and explain our solution convincingly. Also, I was used to finding inspirations from reading papers, and such methodologies in conducting research is more important than any concrete skills or knowledge I leaned during this research like MySQL or Personalized PageRank. I like way that how our group meeting is held, since work in the group of four provides me information about what other people are doing, those are all new knowledge to me and sometimes can also inspire me in my own task as well. Also, requiring us reading a paper each week is also helping us forming the correct researching methodology as I mentioned before.

References

[1] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[2] Tarek Saier and Michael Färber. unarxive: a large scholarly data set with publications' full-text, annotated in-text citations, and links to metadata. *Scientometrics*, 125:3085–3108, 03 2020.

[3] Jieming Shi, Renchi Yang, Tianyuan Jin, Xiaokui Xiao, and Yin Yang. Realtime top-k personalized pagerank over large graphs on gpus. *Proc. VLDB Endow.*, 13(1):15–28, September 2019.

[4] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 243–246, New York, NY, USA, 2015. Association for Computing Machinery.

[5] W. Xing and A. Ghorbani. Weighted pagerank algorithm. In *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.*, pages 305–314, 2004.

[6] Zhibo Zhu, Qinke Peng, Zhi Li, Xinyu Guan, and Owais Muhammad. Fast pagerank computation based on network decomposition and dag structure. *IEEE Access*, 6:41760–41770, 2018.