BIG DATA ANALYTICS REPORT (ILS – Z604)

MICROSOFT ACADEMIC GRAPH

-By

ABHISHEK ANAND SINGH *(aasingh@indiana.edu)*

MILIND GOKHALE *(mgokhale@indiana.edu)*

RENUKA DESHMUKH *(renudesh@indiana.edu)*

SANTHOSH SUNDARAJAN *(soundars@indiana.edu)*

RITESH AGARWAL *(riteagar@indiana.edu)*

Contents

Problem Statement:

Microsoft has provided a comprehensive dataset of more than 120 million papers from 119 million authors with several other attributes including – keywords, conferences, journals, affiliations, etc. We intend to create a heterogeneous graph from this dataset and perform graph mining to achieve two tasks

1. Citation Recommendation for paper
2. Keyword Recommendation for paper

In task 1, given a paper id, we tried to predict Paper References (citation) based on the paper keywords, authors and other such data. This is an important problem as it can be used to recommend other reading material for researchers similar to the paper they are interested in.

In task 2, given a paper id, we try to predict key words that the paper can be tagged with.

Data Processing:

Dataset Description:

The Microsoft Academic Graph is a heterogeneous graph containing scientific publication records, citation relationships between those publications, as well as authors, institutions, journals and conference "venues" and fields of study.

Collection	*Records*	*Size*
Affiliations	**19,849**	**753 KB**
Authors	**119,892,201**	**2 GB**
ConferenceInstances	**50,202**	**10 MB**
Conferences	**1,275**	**78 KB**
FieldsOfStudy	**53,834**	**1.5 MB**
Journals	**23,568**	**1 MB**
Papers	**120,887,833**	**26 GB**

PaperAuthorAffiliations	**312,274,259**	**17.5 GB**
PaperKeywords	**157,052,442**	**5 GB**
PaperReferences	**952,364,264**	**17.6 GB**
PaperUrls	**349,947,403**	**26 GB**

Data Pre-processing and Indexation:

In order to tackle the storage and computational restrictions imposed by the scale of the entire data (which tops 100GB uncompressed), we imported the dataset into a MongoDB database. Indexing was required to optimize the search queries as non-indexed search took very long time. We indexed the data using MongoDB's B-Tree index. As per our requirement, we indexed the tables on the columns with frequent queries and/or joins.

Snowball Sampling:

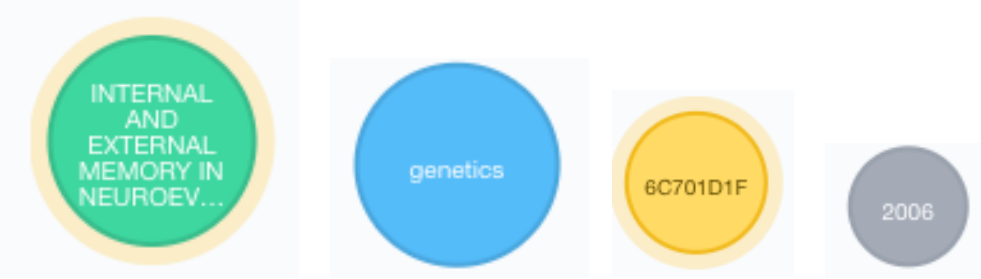
We implemented sampling methodology inspired by snowball sampling and fetched 100,000 papers and all the corresponding attributes related to these papers. We started with a random sample of 1,000 papers and fetched corresponding data for these papers from other tables. This was done to ensure a diversity of the paper-set, while also maintaining the completeness of the sample dataset. After initial analysis and study of this sample dataset, we increased the sample size to 10,000 and 100,000 papers respectively.

For the final set of 100,000 papers, the total number of nodes and relationships in the graph after this sampling are 1.3M and 1.8M respectively.

Graph Structure:

Nodes-

We have 5 type of nodes- Paper, Keyword, Author, Reference, and Published Year



Paper Node *Keyword* *Author* *Published Year*

Each node has following ***properties****:*

Paper *[id, rank, paperID, title, year, isJrnl]*

Keyword *[id, keyword]*

Author *[id, authorID, paperID]*

PublishedYear *[id, year, paperID]*

Relationships-

We have multiple relationships between above nodes as follows-

Paper → ***HAS_AUTHOR***

Paper → ***HAS_KEYWORD***

Paper → ***REFERS2***

Paper → ***PUBLISHED_IN***

Task 1: Citation Recommendation Problem

Methodology:

The task of recommendation has been previously approached in several works through sophisticated information retrieval, machine learning and graph mining techniques. We focus primarily on Graph Mining and provide recommendation based on traversing the nodes and edges in the graph. Following is a description of our take on citation recommendation: -

- A hybrid data storage and retrieval approach has been utilized to counteract the restrictions imposed by graph and noSQL databases.

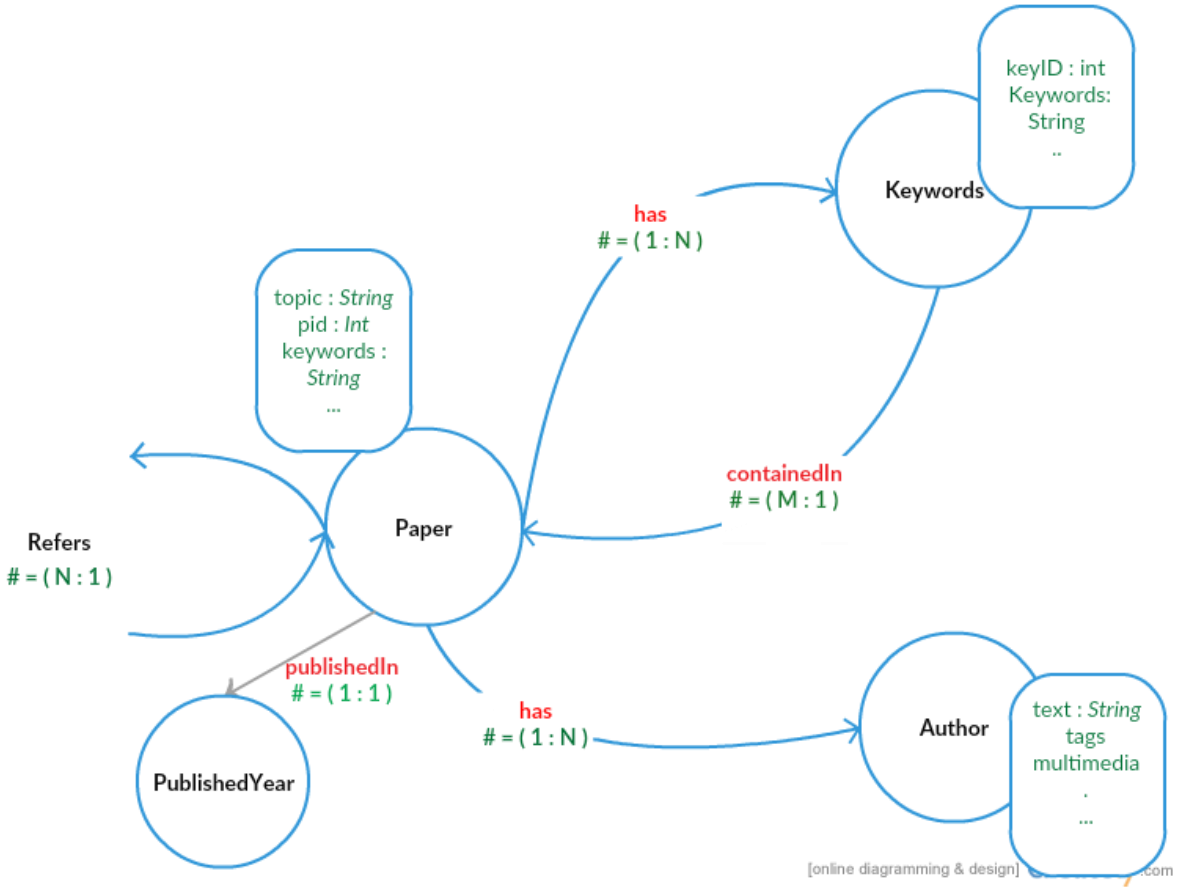
- A hierarchical sampling approach has been followed to completely exploit the various relationships which exist in the data.
- The attributes of paper, such as keyword, paper rank, author, and published year are used to mine the graph & predict the citations.
- A ranking algorithm has been devised to perform a weighted evaluation of every attribute and publish the results in the order of relevance.

Algorithm:

In order to accomplish our task successfully, we used a hybrid approach of using a graph database (Neo4J) for graph mining & recommendation, and MongoDB to rank & evaluate the recommendations. This synergy led to a complex yet highly effective recommendation and evaluation system.

MongoDB: The following steps are for preparing the data for loading into Neo4J

1. Loading CSV files in Microsoft Academic Dataset into MongoDB
2. Indexing tables in MongoDB
3. Create a sample from this dataset using a sampling algorithm.
4. Divide this sample dataset into respective tables in CSV format to be able to import this data into Neo4J in form of nodes and relationships.



JSON object sample:

{

"_id":{"$oid":"571a48d1f6d58545f853a4cb"},

"PId":"5853E23C",

"PublYear":2008,

"PRank":19485,

"IsConf":true,

"IsJrnl":false,

"FOSIds":"083736DA|0304C748|04EDB532|8.782E12|7868074|097C6C78|06ADBDFF|

","KW":"optimization problem|artificial neural network|external memory|genetic

algorithm|objective function|evolutionary algorithm|explicit memory|",

"PRefIds":"5D82C5E4|587A64D7|5A4CB248|0233D103|5B74E9C2|5BD9C7F1|137A2F83|6D46A002|6840DDAB|6DC727F2|707B3A51|5A765E23|5E43E5A4|59B0F480|0C33AD48|5D193415|0C6C6BDB|6DA73FA5|3.75E173|63E30B04|",
"AuthIds":"0E2CDBF9|49CC53F2|700180FB|"
}

Neo4J: The following steps will create a connected graph of the sampled data

1- Load the csv data created in above step as normalized tables from Microsoft Academic graph dataset in Neo4j. This will create Paper node with it's properties.

2- Create index and unique constraint on paperID, ensuring uniqueness of paperIDs.

3- Now, load the keyword table based on matching paperID, thus creating Keyword node. Merge the keyword node to maintain distinction of each keyword.

4- Create [HAS_KEYWORD] relationship between keyword node and paper node.

5- Repeat step 3, 4 for every other table/node and relationship i.e. Author [HAS_AUTHOR], PublishedYear [PUBLISHED_IN], Reference [REFERS2].

Representation of Nodes and their corresponding Relationships

Paper Recommendation Algorithm

1- We start with a Paper node, P1.
2- Given this paper P1, we retrieve all the keywords related to this paper.
3- We fetch all the Paper nodes that share these keywords i.e. Papers which have one or more of these keywords and Publish_year earlier than P1.
4- We provide these papers as recommended citations to our initial paper, P1.
5- In order to filter and rank the most relevant papers, we perform our proprietary ranking algorithm and display the results in an ordered fashion.

Ranking Algorithm

Ranking algorithm is used to extract relevant documents, based on their relevance. In task 1, we designed and implemented a Ranking Algorithm to rank the papers, so that our system could recommend the best papers for citation. We started with Paper Rank and added other parameters, one at a time, to derive the best ranking algorithm. We also assigned weights to each parameters and varied the weights to test for best accuracy. We describe the final ranking function below:

1- We considered 4 factors that we believe would provide us with an optimal result, namely Author_Popularity, Paper_Publish_Year, Paper_Rank and Paper_Keywords.
2- Author_Popularity is finding the most popular author from all authors of a paper. We believe that the most popular authors have a high probability of being cited. To do that we only consider the author who has written the maximum number of papers and save the count of papers published to be consumed later in the ranking function.
3- We found that the most recent papers of a field have more chances of being referenced than the older ones, hence we used Paper_Publish_Year for ranking.
4- Keywords are an important factor. We believe that the more keywords match between 2 papers, there is a very high probability for such papers to be referenced.
5- Lastly, we also believe using Paper_Rank can be a good feature but we give it a low weightage.
6- We normalized all the 4 features so that the values lie between 0 and 1. This was done so that a high spike in any one of the values does not affect the output. [4]

7- We assigned weights to each of these features based on their relevance and importance and calculated the final score for each paper.

8- Then we considered the top ranked documents as final recommendation.

Ranking function:

F(x) = 0.2 * Author_Popularity_Normalized + 0.5 * Paper_Keywords_Normalized + 0.2 * Paper_Publish_Year_Normalized + 0.1 * Paper_Rank_Normalized

Task -2: Predict Papers with most relevant Keywords

Problem Statement:

In this task, given a set of keywords belonging to a certain paper we want to compute the most relevant keywords with maximum relevance.

Methodology:

This is a **Page Rank** model we implemented on a subset of data (described below) that will compute the weight of each paper based on **KeyWord Similarity Match, PRank and Author Relevance**. This will help the user to *search for academic papers* from this graph database that will precisely fall under the target category with the help of this ranking function. We have used python language to compute these Page Weights taking the subset data in *JSON* format.

We considered this task to be an information retrieval task and hence decided to use Page Rank method to compute papers based on the weights discussed in the below algorithm. We assign more weight [0.6] to the **Keyword Similarity Match** feature for the purpose of serving our need of recommending papers with most accurate match for a given paper.

Algorithm:

Step 1: Load the data from *PaperCollection.json* file and use *python re* (regex) to fetch the values for *{PId, PublYear, Prank, KW (keywords), PRefIds }.*

Step 2: Filter out papersIds published after target paper's *PublYear* year.

Step 3: Our weight computation function is of form: $\frac{\max\_weight * currentPid\_weight}{\max\_paper\_count}$ *where max_weight = 0.2 in the case of* **Author Relevance Weight.**

Step 4: Compute **Author Relevance Weight** with the range of [0,0.2] and assign it to Paper Weight Dictionary.

Step 5: Compute **Keyword Similarity Weight** with the range of [0,0.6] and add it to the already computed weights Paper Weight Dictionary.

Step 6: Compute relative **PRank** based weight with the range of [0,0.2] and add with the overall Paper Weight Dictionary.

Step 7: Now we get an input from the user for the number of papers to be shown per IR task and return the specified number of papers.

Step 8: We finally iterate over these papers for fetch the relevant keywords predicted by the model.

Evaluation Metrics:

For both tasks we have considered Precision and Recall as the evaluation metrics. Since in both tasks predict citations and keywords respectively, precision and recall would be apt to evaluate correctness of this model.

Recall:

Recall was considered to check how relevant the predictions of our model was. A higher recall would indicate that the model has predicted most of the relevant categories for each business.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Precision:

Precision was considered to how accurate our model performed while predicting. A higher precision would indicate that the model has predicted less irrelevant categories for each business.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Experiment and Evaluation:

Task 1:

Experiment:

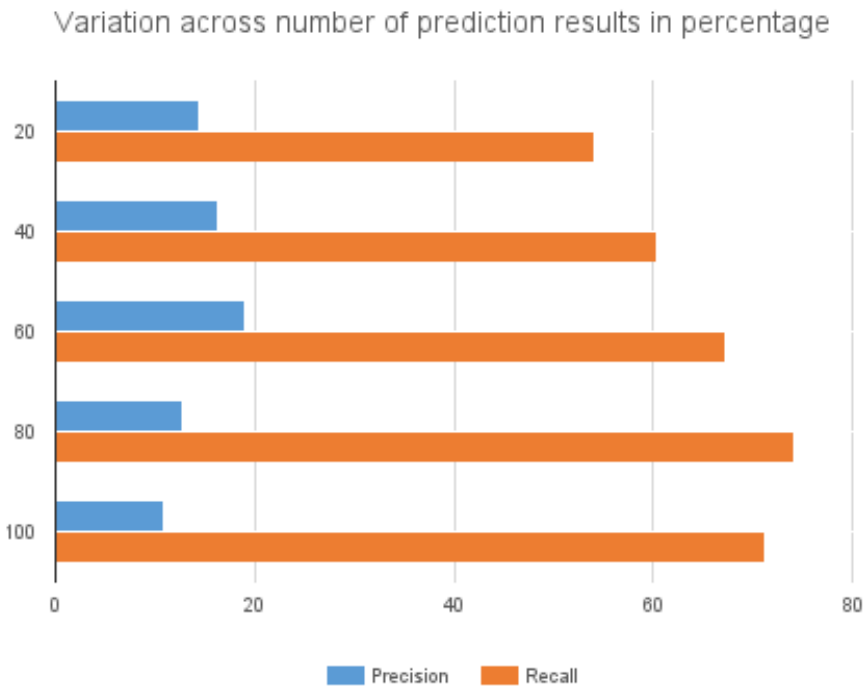
In the initial proposal for the project, we proposed citation recommendation using text analysis of paper text or abstract. However, due to the lack of such textual data, we could not do text analysis. In our current approach, we varied the dataset size from 1000 papers to 100,000 papers to test our proposed model with varied input size. We also varied the number of citation recommendations

from 20 to 100 with an interval of 20, to analyze the variation in precision and recall for these recommendations.

Evaluation:

When testing for dataset of different sizes we found that we get the best results from the largest dataset which has details about 100,000 papers and all its references. Increasing the dataset size from 1000 to 100,000 papers saw an increase in the overall precision and recall.

Hence, we decided to primarily work with the largest sampled dataset. With this dataset, we varied the number of prediction size from 20 to 100 in steps of 20. We found that we got the best precision with 60 predictions, with the value of 18%.



A sample of ground truth and our prediction

Ground Truth:

0404A0F1:58959A0F,5BDF59A2,5D8AB7F3,06638D0B,399E2684,5AA7A0D1,5C0A24F6,59AD7AF0,625658B0,0B7FFE1E,6B891BF1,69DF6B78,08424B9C,6651E41C,6198FB09,61BCF31A,064A9

D02,67AEAFCD,016606F2,0330DA1A,0ADD9FFE,5D848A5D,694C89B2,0F129181,02F60458,593E5200,5C70C5D0,5FBACA0F,6B2EA82B

Prediction:

0404A0F1:0330DA1A,5C8070CA,58959A0F,0B7FFE1E,0D734EE0,590FD475,5D1B2007,08424B9C,5AC592B3,6FFDAA4F,6CC24931,62B9FD6D,601FDDCF,5FEA8C3B,6BC60F56,5ADD85C1,691D37E4,0ADD9FFE,694C89B2,6CC24932

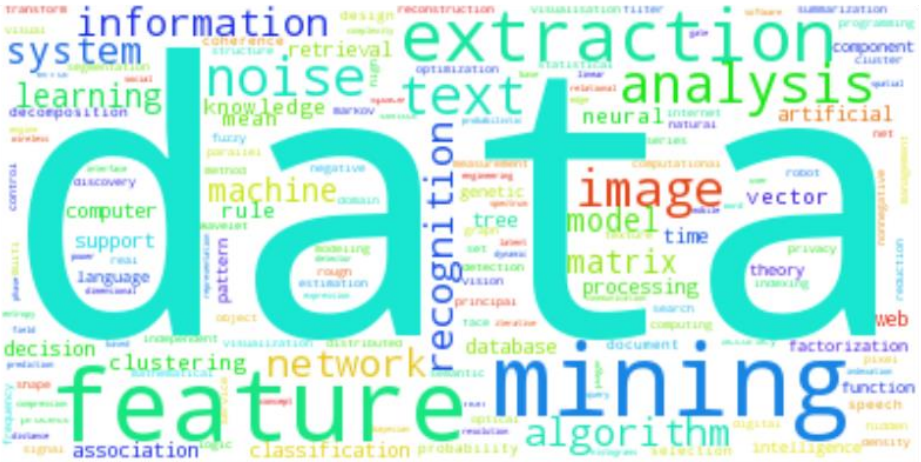
Highlighted were exact match.

Task 2:

We computed the precision recall measures for this task and found some unstable results since the data under consideration is a subset and building a ground truth on this sub-set either gave us exact match of 100%, 50% or no match in most of the cases since we found only 1 or 2 PRefIds for the ground truth which can either be present in the predicted list or not be present.

For a given input *{Pid: "5D03E53B" , KW: "matrix decomposition, nonnegative matrix factorization, data mining, noise, multi document summarization, k means, coherence, feature extraction, text analysis "}*

The Ranking function returns top 50 papers with the following KeyWords visualized in the python WordCloud plot.



We can see the visualized keywords above that correlate with the target set of keyword.

Tool-kit and API

- Mongo DB
- Neo4J
- Java
- R
- Unix Bash

Analysis and Conclusion:

We successfully completed the task of paper citation recommendation. We got the best accuracy of 18% when we varied the number of recommended citations to 60. We believe this result can be improved by considered additional features provided in the dataset.

For task 2, we believe we need some more research. We also need to identify some reliable way for calculating precision and recall which would in turn help us improve the performance of the algorithm.

Future Work:

Possibly to further improve accuracy, we can implement a better ranking function by considering more features like conference locations, journals, Fields of Study, etc. On analysis of data we found that some paper do not have complete and exhaustive keyword tagging. This could be handled by studying similarity between different keywords, which has been left as future scope. Also, task 2 can be used for predicting keywords for a given paper and those keywords can be input to citation recommendation algorithm.

For task 2, we would like to extend the JSON file to contain the entire papers data given by Microsoft to see fruitful results in the form of precision recall.

References:

1. "Precision and Recall - Wikipedia the Free Encyclopaedia." Wikipedia. Accessed May 2, 2016. https://en.wikipedia.org/wiki/Precision_and_recall.
2. He, Qi, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. "Context-aware Citation Recommendation." *Proceedings of the 19th International Conference on World Wide Web - WWW '10*, 2010.
3. Liu, Xiaozhong, Yingying Yu, Chun Guo, Yizhou Sun, and Liangcai Gao. "Full-text Based Context-rich Heterogeneous Network Mining Approach for Citation Recommendation." *IEEE/ACM Joint Conference on Digital Libraries*, 2014.
4. "Feature Scaling." Wikipedia. Accessed May 03, 2016. https://en.wikipedia.org/wiki/Feature_scaling.
5. Deshmukh, Renuka, Milind Gokhale, Abhishek Singh, Sathosh Soundarajan, and Ritesh Agarwal. "Github Repository for Microsoft Academic Data Big Data Project." Accessed May 3, 2016. https://github.com/milindhg/Microsoft_Academic_Graph.